Cat. No. W340-E1-16

SYSMAC CS Series

CS1G/H-CPU□□-EV1
CS1G/H-CPU□□H
CS1D-CPU□□H
CS1D-CPU□□S

SYSMAC CJ Series

CJ1H-CPU□□H-R
CJ1G-CPU□□
CJ1G/H-CPU□□H
CJ1G-CPU□□P
CJ1M-CPU□□

SYSMAC One NSJ Series

Programmable Controllers

INSTRUCTIONS REFERENCE MANUAL

OMRON

SYSMAC CS Series

CS1G/H-CPU□□-EV1
CS1G/H-CPU□□H
CS1D-CPU□□H
CS1D-CPU□□S

SYSMAC CJ Series

CJ1H-CPU□□H-R
CJ1G-CPU□□
CJ1G/H-CPU□□H
CJ1G-CPU□□P
CJ1M-CPU□□

SYSMAC One NSJ Series

Programmable Controllers

Instructions Reference Manual

Revised August 2008

Notice:

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

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

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

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

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

OMRON Product References

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

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

The abbreviation “PLC” means Programmable Controller. “PC” is used, however, in some Programming Device displays to mean Programmable Controller.

Visual Aids

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

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

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

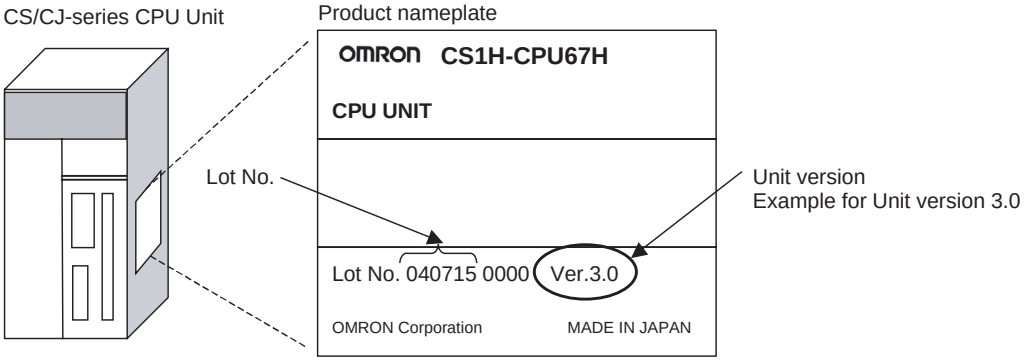
Unit Versions of CS/CJ-series CPU Units

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. This applies to the CS1-H, CJ1-H, CJ1M, and CS1D CPU Units.

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.



- CS1-H, CJ1-H, and CJ1M CPU Units manufactured on or before November 4, 2003 do not have a unit version given on the CPU Unit (i.e., the location for the unit version shown above is blank).
- The unit version of the CJ1-H-R CPU Units begins at version 4.0.
- The unit version of the CS1-H, CJ1-H, and CJ1M CPU Units, as well as the CS1D CPU Units for Single-CPU Systems, begins at version 2.0.
- The unit version of the CS1D CPU Units for Duplex-CPU Systems, begins at version 1.1.
- CPU Units for which a unit version is not given are called *Pre-Ver. □.□ CPU Units, such as Pre-Ver. 2.0 CPU Units and Pre-Ver. 1.1 CPU Units*.

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.

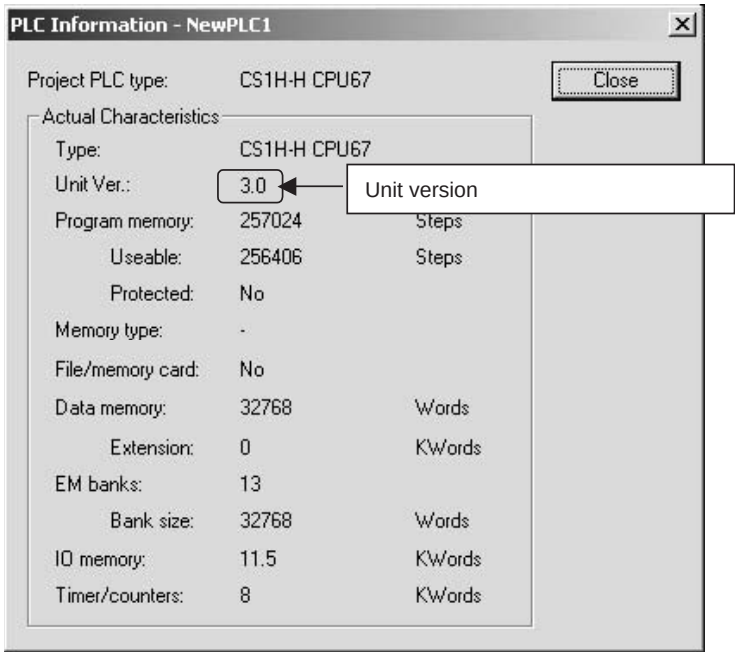
- Using the ***PLC Information***
- Using the ***Unit Manufacturing Information*** (This method can be used for Special I/O Units and CPU Bus Units as well.)

Note CX-Programmer version 3.3 or lower cannot be used to confirm unit versions.

PLC Information

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

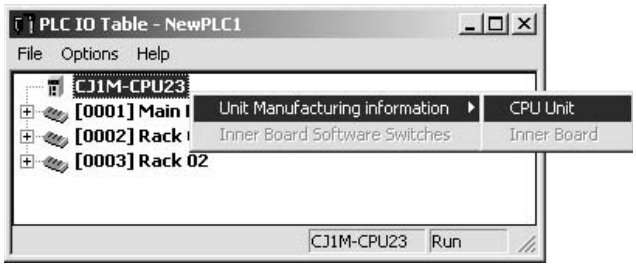
In either case, the following *PLC Information* Dialog Box will be displayed.



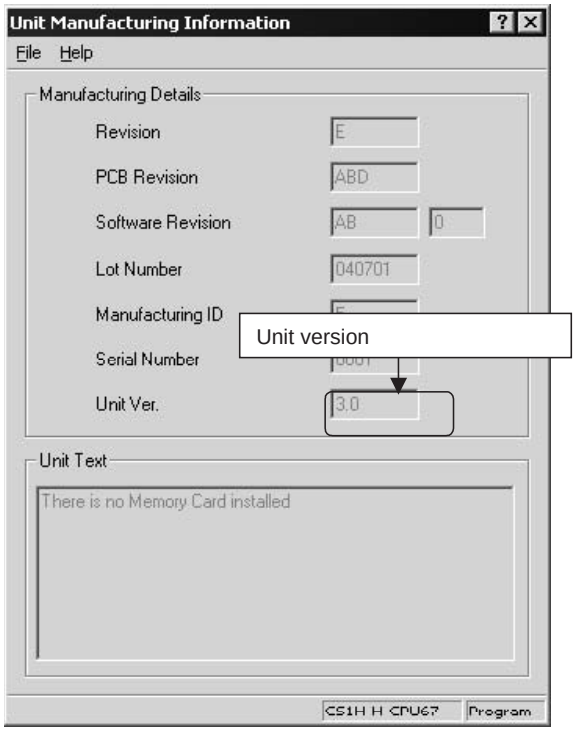
Use the above display to confirm the unit version of the CPU Unit.

Unit Manufacturing Information

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



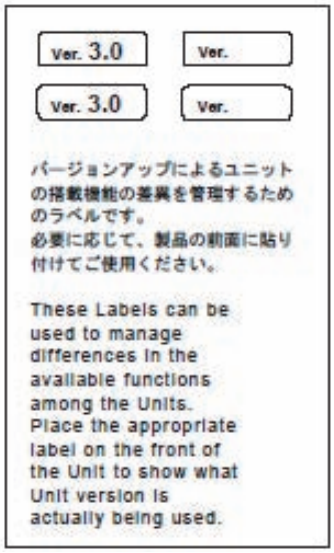
The following *Unit Manufacturing information* Dialog Box will be displayed.



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

Using the Unit Version Labels

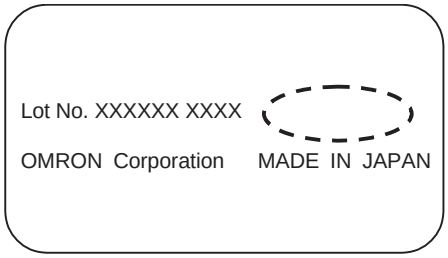
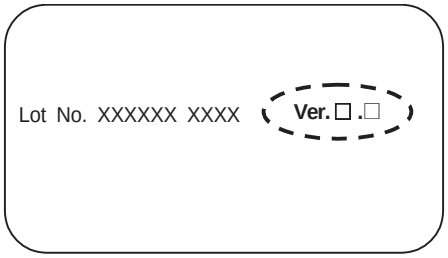
The following unit version labels are provided with the CPU Unit.



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

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 4.2
		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
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 Ver. 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 Ver. 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 Ver. 2.0	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 Ver. 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 Ver.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 added for unit version 4.0 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. CX-Programmer version 7.0 or higher is required to use unit version 4.2 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

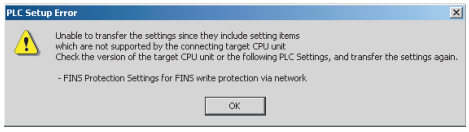
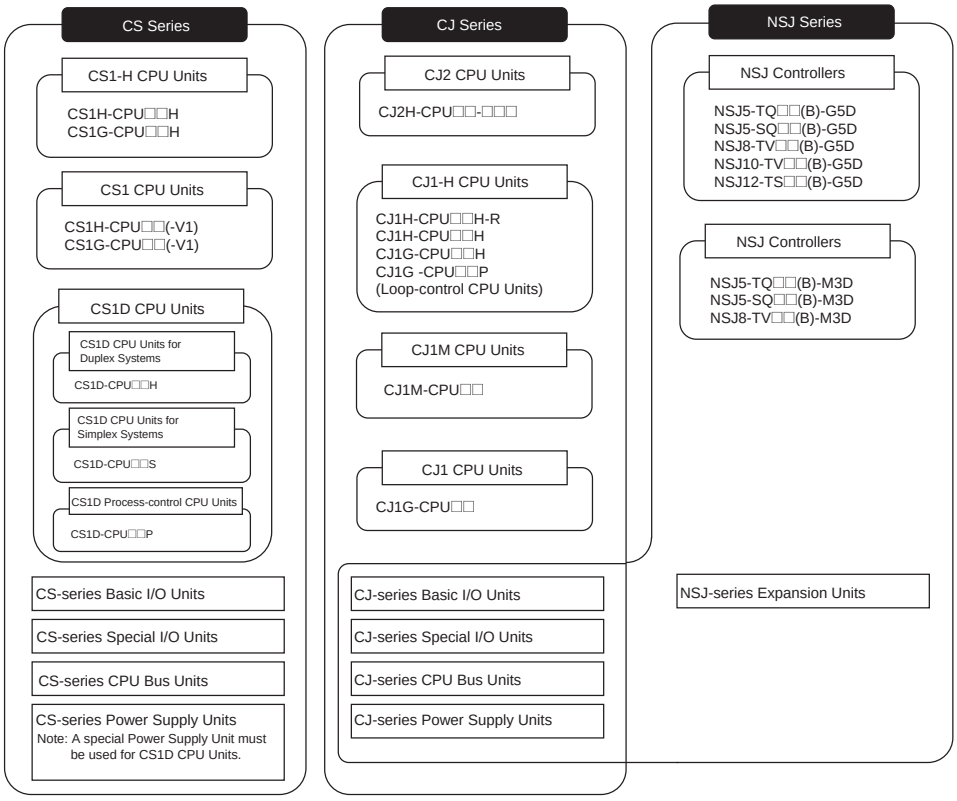
Problem	Cause	Solution
 After the above message is displayed, a compiling error will be displayed on the *Compile* Tab Page in the Output Window.	An attempt was made to download a program containing instructions supported only by later unit versions or a CPU Unit to a previous unit version.	Check the program or change to a CPU Unit with a later unit version.
	An attempt was 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.

TABLE OF CONTENTS

TABLE OF CONTENTS

About this Manual:

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



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.

Section 1 introduces the CS/CJ-series PLCs in terms of the instruction set that they support.

Section 2 provides various lists of instructions that can be used for reference.

Section 3 individually describes the instructions in the CS/CJ-series instruction set.

Section 4 provides instruction execution times and the number of steps for each CS/CJ-series instruction.

About this Manual, Continued

Name	Cat. No.	Contents
SYSMAC CS/CJ/NSJ Series CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H, CS1D-CPU□□H, CS1D-CPU□□S, CJ1H-CPU□□H-R, CJ1G-CPU□□, CJ1G/H-CPU□□H, CJ1G-CPU□□P, CJ1M-CPU□□, NSJ□-□□□□(B)-G5D, NSJ□-□□□□(B)-M3D Programmable Controllers Instructions Reference Manual	W340	Describes the ladder diagram programming instructions supported by CS/CJ/NSJ-series PLCs. (This manual)
SYSMAC CS/CJ/NSJ Series CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H, CS1D-CPU□□H, CS1D-CPU□□S, CJ1H-CPU□□H-R, CJ1G-CPU□□, CJ1G/H-CPU□□H, CJ1G-CPU□□P, CJ1M-CPU□□, NSJ□-□□□□(B)-G5D, NSJ□-□□□□(B)-M3D Programmable Controllers Programming Manual	W394	This manual describes programming and other methods to use the functions of the CS/CJ/NSJ-series PLCs.
SYSMAC CS Series CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H Programmable Controllers Operation Manual	W339	Provides an outlines of and describes the design, installation, maintenance, and other basic operations for the CS-series PLCs.
SYSMAC CJ Series CJ1H-CPU□□H-R, CJ1G/H-CPU□□H, CJ1G-CPU□□P, CJ1G-CPU□□, CJ1M-CPU□□ Programmable Controllers Operation Manual	W393	Provides an outlines of and describes the design, installation, maintenance, and other basic operations for the CJ-series PLCs.
SYSMAC CJ Series CJ1M-CPU21/22/23 Built-in I/O Functions Operation Manual	W395	Describes the functions of the built-in I/O for CJ1M CPU Units.
SYSMAC CS Series CS1D-CPU□□H CPU Units CS1D-CPU□□S CPU Units CS1D-DPL1 Duplex Unit CS1D-PA207R 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.
SYSMAC CS/CJ Series CQM1H-PRO01-E, C200H-PRO27-E, CQM1-PRO01-E Programming Consoles Operation Manual	W341	Provides information on how to program and operate CS/CJ-series PLCs using a Programming Console.
SYSMAC CS/CJ/NSJ Series CJ1H-CPU□□H-R, CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H, CS1D-CPU□□H, CS1D-CPU□□S, CJ1M-CPU□□, CJ1G-CPU□□, CJ1G-CPU□□P, CJ1G/H-CPU□□H, CS1W-SCB□□-V1, CS1W-SCU□□-V1, CJ1W-SCU□□-V1, CP1H-X□□□□-□, CP1H-XA□□□□-□, CP1H-Y□□□□-□, 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.

Name	Cat. No.	Contents
NSJ Series NSJ5-TQ□□(B)-G5D, NSJ5-SQ□□(B)-G5D, NSJ8-TV□□(B)-G5D, NSJ10-TV□□(B)-G5D, NSJ12-TS□□(B)-G5D Operation Manual	W452	Provides the following information about 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)
SYSMAC WS02-CX□□-V□ CX-Programmer Operation Manual	W446	Provides information on how to use the CX-Programmer for all functionality except for function blocks.
SYSMAC WS02-CX□□-V□ CX-Programmer Ver. 7.0 Operation Manual Function Blocks (CS1G-CPU□□H, CS1H-CPU□□H, CJ1G-CPU□□H, CJ1H-CPU□□H, CJ1M-CPU□□, CP1H-X□□□□-□, CP1H-XA□□□□-□, CP1H-Y□□□□-□ CPU Units)	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. Functionality that is the same as that of the CX-Programmer is described in W446 (enclosed).
SYSMAC CS/CJ Series CS1W-SCB□□-V1, CS1W-SCU□□-V1, CJ1W-SCU□□-V1 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.
SYSMAC WS02-PSTC1-E 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□□C-V3/AL□□D-V3 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□□C-V3/AL□□D-V3 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.

Read and Understand this Manual

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

Warranty and Limitations of Liability

WARRANTY

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

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

LIMITATIONS OF LIABILITY

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

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

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

Application Considerations

SUITABILITY FOR USE

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

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

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

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

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

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

PROGRAMMABLE PRODUCTS

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

Disclaimers

CHANGE IN SPECIFICATIONS

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

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

DIMENSIONS AND WEIGHTS

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

PERFORMANCE DATA

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

ERRORS AND OMISSIONS

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

PRECAUTIONS

This section provides general precautions for using the CS/CJ-series Programmable Controllers (PLCs) and related devices.

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

1 Intended Audience

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

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

2 General Precautions

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

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

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

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

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

3 Safety Precautions

⚠WARNING The CPU Unit refreshes I/O even when the program is stopped (i.e., even in PROGRAM mode). Confirm safety thoroughly in advance before changing the status of any part of memory allocated to I/O Units, Special I/O Units, or CPU Bus Units. Any changes to the data allocated to any Unit may result in unexpected operation of the loads connected to the Unit. Any of the following operation may result in changes to memory status.

- Transferring I/O memory data to the CPU Unit from a Programming Device.
- Changing present values in memory from a Programming Device.
- Force-setting/-resetting bits from a Programming Device.
- Transferring I/O memory files from a Memory Card or EM file memory to the CPU Unit.
- Transferring I/O memory from a host computer or from another PLC on a network.

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

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

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

⚠**WARNING** Provide safety measures in external circuits (i.e., not in the Programmable Controller), including the following items, to ensure safety in the system if an abnormality occurs due to malfunction of the PLC or another external factor affecting the PLC operation. Not doing so may result in serious accidents.

- Emergency stop circuits, interlock circuits, limit circuits, and similar safety measures must be provided in external control circuits.
- The PLC will turn OFF all outputs when its self-diagnosis function detects any error or when a severe failure alarm (FALS) instruction is executed. As a countermeasure for such errors, external safety measures must be provided to ensure safety in the system.
- The PLC 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.
- When the 24-V-DC output (service power supply to the PLC) is overloaded or short-circuited, the voltage may drop and result in the outputs being turned OFF. As a countermeasure for such problems, external safety measures must be provided to ensure safety in the system.

⚠**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 operation 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** The CS1-H, CJ1-H, CJ1M, and CS1D 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, and HR Areas), however, is not written to flash memory. The DM, EM, and HR Areas 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, and HR Areas are used to control external outputs, prevent inappropriate outputs from being made whenever the Battery Error Flag (A40204) is ON.

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

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

⚠**Caution** Do not touch the Power Supply Unit 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-CLK12/52(-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.

4 Operating Environment Precautions

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

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

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

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

⚠**Caution** The operating environment of the PLC System can have a large effect on the longevity and reliability of the system. Improper operating environments can lead to malfunction, failure, and other unforeseeable problems with the PLC System. 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.

5 Application Precautions

Observe the following precautions when using the PLC System.

- You must use the CX-Programmer (programming software that runs on Windows) if you need to program more than one task. A Programming Console can be used to program only one cyclic task plus interrupt tasks.

A Programming Console can, however, be used to edit multitask programs originally created with the CX-Programmer.

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

- Always connect to a ground of 100 Ω or less when installing the 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.
- 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, Inner Boards, or any other Units.
  - Assembling the Units.
  - Setting DIP switches or rotary switches.
  - Connecting cables or wiring the system.
  - Connecting or disconnecting the connectors.

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

- The user program and parameter area data in the CS1-H, CS1D, CJ1-H, and CJ1M 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.
- When using a CS-series CS1 CPU Unit for the first time, install the CS1W-BAT1 Battery provided with the Unit and clear all memory areas from a Programming Device before starting to program. When using the internal clock, turn ON power after installing the battery and set the clock from a Programming Device or using the DATE(735) instruction. The clock will not start until the time has been set.
- When the CPU Unit is shipped from the factory, the PLC Setup is set so that the CPU Unit will start in the operating mode set on the Programming Console mode switch. When a Programming Console is not connected, a CS-series CS1 CPU Unit will start in PROGRAM mode, but a CS1-H, CS1D, CJ1, CJ1-H, or CJ1M CPU Unit will start in RUN mode and operation will begin immediately. Do not advertently or inadvertently allow operation to start without confirming that it is safe.
- 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.

- 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.
- Fail-safe measures must be taken by the customer to ensure safety in the event that outputs from Output Units remain ON as a result of internal circuit failures, which can occur in relays, transistors, and other elements.
- 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.
- Interlock circuits, limit circuits, and similar safety measures in external circuits (i.e., not in the Programmable Controller) must be provided by the customer.
- Do not turn OFF the power supply to the PLC when data is being transferred. In particular, do not turn OFF the power supply when reading or writing a Memory Card. Also, do not remove the Memory Card when the BUSY indicator is lit. 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.
- 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.)
- The contents of the DM, EM, and HR Areas in the CPU Unit are backed up by a Battery. If the Battery voltage drops, this data may be lost. Provide countermeasures in the program using the Battery Error Flag (A40204) to re-initialize data or take other actions if the Battery voltage drops.
- When supplying power at 200 to 240 V AC with a CS-series PLC, always remove the metal jumper from the voltage selector terminals on the Power Supply Unit (except for Power Supply Units with wide-range specifications). The product will be destroyed if 200 to 240 V AC is supplied while the metal jumper is attached.
- Always use the power supply voltages specified in the operation manuals. An incorrect voltage may result in malfunction or burning.
- Take appropriate measures to ensure that the specified power with the rated voltage and frequency is supplied. Be particularly careful in places where the power supply is unstable. An incorrect power supply may result in malfunction.
- 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 apply voltages to the Input Units in excess of the rated input voltage. Excess voltages may result in burning.
- 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.

- 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.
- Install the Units properly as specified in the operation manuals. Improper installation of the Units may result in malfunction.
- With CS-series PLCs, be sure that all the Unit and Backplane mounting screws are tightened to the torque specified in the relevant manuals. Incorrect tightening torque may result in malfunction.
- Be sure that all terminal screws, and cable connector screws are tightened to the torque specified in the relevant manuals. Incorrect tightening torque may result in malfunction.
- 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.
- Wire all connections correctly.
- Double-check all wiring and switch settings before turning ON the power supply. Incorrect wiring may result in burning.
- Mount Units only after checking terminal blocks and connectors completely.
- Be sure that the terminal blocks, Memory Units, expansion cables, and other items with locking devices are properly locked into place. Improper locking may result in malfunction.
- Check switch settings, the contents of the DM Area, and other preparations before starting operation. Starting operation without the proper settings or data may result in an unexpected operation.
- Check the user program for proper execution before actually running it on the Unit. Not checking the program may result in an unexpected operation.
- Confirm that no adverse effect will occur in the system before attempting any of the following. Not doing so may result in an unexpected operation.
    - Changing the operating mode of the PLC (including the setting of the startup operating mode).
    - Force-setting/force-resetting any bit in memory.
    - Changing the present value of any word or any set value in memory.
- Do not pull on the cables or bend the cables beyond their natural limit. Doing either of these may break the cables.
- Do not place objects on top of the cables 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.

- When replacing parts, be sure to confirm that the rating of a new part is correct. Not doing so may result in malfunction or burning.
- Before touching 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.
- When transporting or storing circuit boards, cover them in antistatic material to protect them from static electricity and maintain the proper storage temperature.
- Do not touch circuit boards or the components mounted to them with your bare hands. There are sharp leads and other parts on the boards that may cause injury if handled improperly.
- 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 otherwise subjected to excessive shock. Batteries that have been subjected to shock may leak if they are used.
- UL standards require that batteries be replaced only by experienced technicians. Do not allow unqualified persons to replace batteries.
- Dispose of the product and batteries according to local ordinances as they apply. Have qualified specialists properly dispose of used batteries as industrial waste.

廢電池請回收

- With a CJ-series PLC, the sliders on the tops and bottoms of the Power Supply Unit, CPU Unit, I/O Units, Special I/O Units, and CPU Bus Units must be completely locked (until they click into place). The Unit may not operate properly if the sliders are not locked in place.
- With a CJ-series PLC, always connect the End Plate to the Unit on the right end of the PLC. The PLC will not operate properly without the End Plate
- 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.
- 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 allowing the CPU Bus Units to be reset.

6 Conformance to EC Directives

6-1 Applicable Directives

- EMC Directives
- Low Voltage Directive

6-2 Concepts

EMC Directives

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

standards in the system used by the customer, however, must be checked by the customer.

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

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

EMS (Electromagnetic Susceptibility):	EN61131-2 (CS-series)/ EN61000-6-2 (CJ-series)
EMI (Electromagnetic Interference):	EN61000-6-4 (Radiated emission: 10-m regulations)

Low Voltage Directive

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

6-3 Conformance to EC Directives

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

1,2,3...

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

6-4 Relay Output Noise Reduction Methods

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

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

Countermeasures

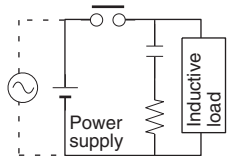
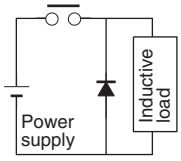
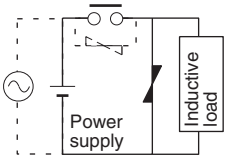
(Refer to EN61000-6-4 for more details.)

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

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

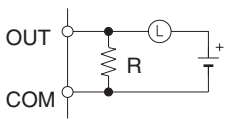
Countermeasure Examples

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

Circuit	Current		Characteristic	Required element
	AC	DC		
CR method 	Yes	Yes	If the load is a relay or solenoid, there is a time lag between the moment the circuit is opened and the moment the load is reset. If the supply voltage is 24 or 48 V, insert the surge protector in parallel with the load. If the supply voltage is 100 to 200 V, insert the surge protector between the contacts.	The capacitance of the capacitor must be 1 to 0.5 μF per contact current of 1 A and resistance of the resistor must be 0.5 to 1 Ω per contact voltage of 1 V. These values, however, vary with the load and the characteristics of the relay. Decide these values from experiments, and take into consideration that the capacitance suppresses spark discharge when the contacts are separated and the resistance limits the current that flows into the load when the circuit is closed again. The dielectric strength of the capacitor must be 200 to 300 V. If the circuit is an AC circuit, use a capacitor with no polarity.
Diode method 	No	Yes	The diode connected in parallel with the load changes energy accumulated by the coil into a current, which then flows into the coil so that the current will be converted into Joule heat by the resistance of the inductive load. This time lag, between the moment the circuit is opened and the moment the load is reset, caused by this method is longer than that caused by the CR method.	The reversed dielectric strength value of the diode must be at least 10 times as large as the circuit voltage value. The forward current of the diode must be the same as or larger than the load current. The reversed dielectric strength value of the diode may be two to three times larger than the supply voltage if the surge protector is applied to electronic circuits with low circuit voltages.
Varistor method 	Yes	Yes	The varistor method prevents the imposition of high voltage between the contacts by using the constant voltage characteristic of the varistor. There is time lag between the moment the circuit is opened and the moment the load is reset. If the supply voltage is 24 or 48 V, insert the varistor in parallel with the load. If the supply voltage is 100 to 200 V, insert the varistor between the contacts.	---

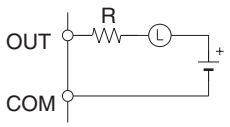
When switching a load with a high inrush current such as an incandescent lamp, suppress the inrush current as shown below.

Countermeasure 1



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

Countermeasure 2



Providing a limiting resistor

SECTION 1
Introduction

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

1-1 General Instruction Characteristics

1-1-1 Program Capacity

The program capacity tells the size of the user program area in the CPU Unit and is expressed as the number of program steps. The number of steps required in the user program area for each of the CS/CJ-series instructions varies from 1 to 7 steps, depending upon the instruction and the operands used with it.

CS Series

The following tables show the maximum number of steps that can be programmed in each CS-series CPU Unit.

- CS1-H CPU Units

Model	Program capacity	I/O points
CS1H-CPU67H	250K steps	5,120
CS1H-CPU66H	120K steps	
CS1H-CPU65H	60K steps	
CS1H-CPU64H	30K steps	
CS1H-CPU63H	20K steps	
CS1G-CPU45H	60K steps	
CS1G-CPU44H	30K steps	1,280
CS1G-CPU43H	20K steps	960
CS1G-CPU42H	10K steps	

- CS1 CPU Units

Model	Program capacity	I/O points
CS1H-CPU67-E	250K steps	5,120
CS1H-CPU66-E	120K steps	
CS1H-CPU65-E	60K steps	
CS1H-CPU64-E	30K steps	
CS1H-CPU63-E	20K steps	
CS1G-CPU45-E	60K steps	
CS1G-CPU44-E	30K steps	1,280
CS1G-CPU43-E	20K steps	960
CS1G-CPU42-E	10K steps	

- CS1D CPU Units for Single-CPU Systems

Model	Program capacity	I/O points
CS1D-CPU67H	250K steps	5,120
CS1D-CPU65H	60K steps	

CS1D CPU Units for Duplex-CPU Systems

Model	Program capacity	I/O points
CS1D-CPU42S	10K steps	960
CS1D-CPU44S	30K steps	1,280
CS1D-CPU65S	60K steps	5,120
CS1D-CPU67S	250K steps	

CJ Series

The following tables show the maximum number of steps that can be programmed in each CJ-series CPU Unit.

- CJ1-H CPU Units

Model	Program capacity	I/O points
CJ1H-CPU67H-R	250K steps	2,560
CJ1H-CPU66H-R	120K steps	
CJ1H-CPU65H-R	60K steps	
CJ1H-CPU64H-R	30K steps	
CJ1H-CPU67H	250K steps	
CJ1H-CPU66H	120K steps	
CJ1H-CPU65H	60K steps	
CJ1G-CPU45H	60K steps	1,280
CJ1G-CPU44H	30K steps	
CJ1G-CPU43H	20K steps	960
CJ1G-CPU42H	10K steps	

- CJ1 CPU Units

Model	Program capacity	I/O points
CJ1G-CPU45	60K steps	1,280
CJ1G-CPU44	30K steps	

- CJ1M CPU Units

Model	Program capacity	I/O points
CJ1M-CPU23	20K steps	640
CJ1M-CPU22	10K steps	320
CJ1M-CPU21	5K steps	160
CJ1M-CPU13	20K steps	640
CJ1M-CPU12	10K steps	320
CJ1M-CPU11	5K steps	160

Note Program capacity for CS/CJ-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/CJ-series instructions, and inaccuracies will occur if the capacity of a user program for another PLC is converted for a CS/CJ-series PLC based on the assumption that 1 word is 1 step. Refer to the information at the end of *SECTION 4 Instruction Execution Times and Number of Steps* for guidelines on converting program capacities from previous OMRON PLCs.

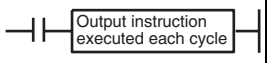
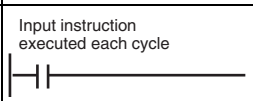
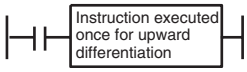
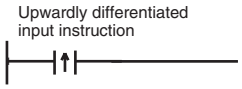
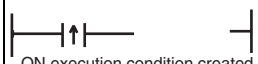
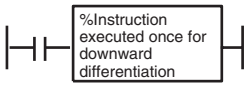
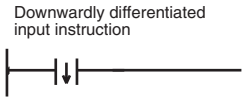
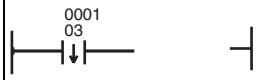
The number of steps in a program is not the same as the number of instructions. For example, LD and OUT require 1 step each, but MOV(021) requires 3 steps. Other instructions require up to 15 steps each. The number of steps required by an instruction is also increased by one step for each double-length operand used in it. For example, MOVL(498) normally requires 3 steps, but 4 steps will be required if a constant is specified for the source word operand, S. Refer to *SECTION 4 Instruction Execution Times and Number of Steps* for the number of steps required for each instruction.

1-1-2 Differentiated Instructions

Most instructions in CS/CJ-series PLCs are provided with both non-differentiated and upwardly differentiated variations, and some are also provided with a downwardly differentiated variation.

- A non-differentiated instruction is executed every time it is scanned.

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

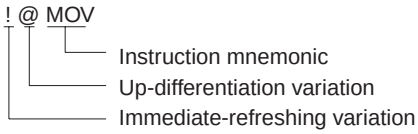
Variation	Instruction type	Operation	Format	Example
Non-differentiated	Output instructions (instructions requiring an execution condition)	The instruction is executed every cycle while the execution condition is true (ON).		
	Input instructions (instructions used as execution conditions)	The bit processing (such as read, comparison, or test) is performed every cycle. The execution condition is true while the result is ON.		
Upwardly differentiated (with @ prefix)	Output instructions	The instruction is executed just once when the execution condition goes from OFF to ON.		 MOV(021) executed once for each OFF to ON transition in CIO 000102.
	Input instructions (instructions used as execution conditions)	The bit processing (such as read, comparison, or test) is performed every cycle. The execution condition is true for one cycle when the result goes from OFF to ON.		 ON execution condition created for one cycle only for each OFF to ON transition in CIO 000103.
Downwardly differentiated (with % prefix)	Output instructions	The instruction is executed just once when the execution condition goes from ON to OFF.		 SET executed once for each ON to OFF transition in CIO 000102.
	Input instructions (instructions used as execution conditions)	The bit processing (such as read, comparison, or test) is performed every cycle. The execution condition is true for one cycle when the result goes from ON to OFF.		 ON execution condition created for one cycle only for each ON to OFF transition in CIO 000103.

Note The downwardly differentiated option (%) is available only for the LD, AND, OR, and RSET instructions. To create downwardly differentiated variations of other instructions, control the execution of the instruction with work bits controlled with DIFD(014) or DOWN(522).

1-1-3 Instruction Variations

The variation prefixes (@, %, and !) can be added to an instruction to create a differentiated instruction or provide immediate refreshing.

Variation		Prefix	Operation
Differentiation	Upwardly differentiated	@	Creates an upwardly differentiated instruction.
	Downwardly differentiated	%	Creates a downwardly differentiated instruction.
Immediate refreshing		!	The instruction's operand data in the I/O Area will be refreshed when the instruction is executed.



1-1-4 Instruction Location and Execution Conditions

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

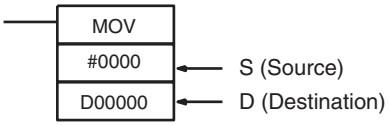
Instruction type		Location	Execution condition	Format	Examples
Input	Instructions that start logic conditions	At the left bus or at the start of an instruction block	Not required		LD, LD TST, and input comparison instructions such as LD >
	Connecting instructions	Between a starting instruction and output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

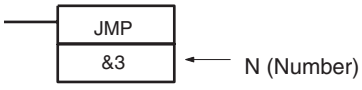
In addition to these instructions, the CS/CJ-series PLCs are equipped with block programming instructions. Refer to the description of the block programming instructions for details.

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

1-1-5 Inputting Data in Operands

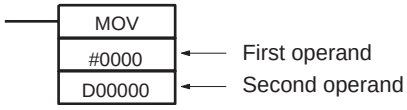
Operands are parameters that are set in advance with the I/O memory addresses or constants to be used when the instruction is executed. There are basically three kinds of operands: Source operands, destination operands, and numbers.





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

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



Specifying Bit Addresses

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

Specifying Word Addresses

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

Specifying Indirect DM/EM Addresses in Binary Mode

Description	Example	Instruction example
When the @ prefix is input before a DM or EM address, the contents of that word specifies another word that is used as the operand. The contents can be 0000 to 7FFF (0 to 32,767), corresponding to the desired word address in the DM or EM Area. @D□□□□□ Content 00000 to 32767 (0000 to 7FFF) D	---	---
When the contents of @D□□□□□ is between 0000 and 7FFF (00000 to 32,767), the corresponding word between D00000 and D32767 is specified.	@D00300 0 1 0 0 Decimal: 256 Specifies D00256. Add the @ prefix.	MOV #0001 @D00300

Description	Example	Instruction example
When the contents of @D□□□□□ is between 8000 and FFFF (32,768 to 65,535), the corresponding word between E0_00000 and E0_32767 in EM bank 0 is specified.	@D00300 8 0 0 1 Decimal: 32,769 ↓ Specifies E0_00001.	---
When the contents of @En□_□□□□□ is between 0000 and 7FFF (00000 to 32,767), the corresponding word between En□_00000 and En□_32767 is specified.	@E1_00200 0 1 0 1 Decimal: 257 ↓ Specifies E1_00257.	MOV #0001 @E1_00200
When the contents of @En□_□□□□□ is between 8000 and FFFF (32,768 to 65,535), the corresponding word between E (□+1) _00000 and E (□+1) _32767 (in the next EM bank) is specified.	@E1_00200 8 0 0 2 Decimal: 32770 ↓ Specifies E2_00002.	

Note When binary mode is selected in the PLC Setup, the DM Area and current EM bank addresses (bank 0 to C) are treated as consecutive memory addresses. A word in EM bank 0 will be specified if an indirectly addressed DM word contains a value greater than 32,767. For example, E00000 in bank 0 will be specified when the indirect-addressing DM word contains a hexadecimal value of 8000 (32,768).

A word in the next EM bank will be specified if an indirectly addressed EM word contains a value greater than 32,767. For example, E3_00000 will be specified when the indirect-addressing EM word in bank 2 contains a hexadecimal value of 8000 (32,768).

Specifying Indirect DM/EM Addresses in BCD Mode

Method	Description	Example	Instruction example
Indirect DM/EM addressing (BCD mode)	When the * prefix is input before a DM or EM address, the BCD contents of that word specify another word that is used as the operand. The contents can be 0000 to 9999, corresponding to the desired word address in the DM or EM Area. *D□□□□□ ↓ Content [ ] 0000 to 9999 (BCD) ↓ D [ ]	*D00200 0 1 0 0 ↓ Specifies D00100. Add the * prefix.	MOV #0001 *D00200

Addressing Index Registers

Method	Description		Example	Instruction example
Directly addressing Index Registers	MOVR(560) moves the PLC memory address of a word or bit to an Index Register (IR0 to IR15). (MOVRW(561) moves the PLC memory address of a timer or counter PV to an Index Register.)		IR0 IR2	MOVR 0010 IR0 Stores the PLC memory address of CIO 0010 in IR0. MOVR 000102 IR2 Stores the PLC memory address of CIO 000102 in IR2.
Indirect addressing with Index Registers	Basic operation (no offset)	The word or bit at the I/O memory address contained in IR□ is used as the operand. Input a comma before the Index Register to indicate indirect addressing. (The bit/word designation can be determined by the instruction or operand.)	,IR0 ,IR1	LD ,IR0 Loads the status of the bit at the I/O memory address contained in IR0. MOV #0001, IR1 Moves #0001 to the word at the I/O memory address contained in IR1.
	Constant offset	The offset value (–2,048 to +2,047) is added to the I/O memory address contained in IR□ and the resulting address is used as the operand. (The offset is converted to binary when the instruction is executed.)	+5 ,IR0 +31 ,IR1	LD +5 ,IR0 Adds 5 to the I/O memory address contained in IR0 and loads the status of the bit at that address. MOV #0001 +31 ,IR1 Adds 31 to the I/O memory address contained in IR1 and moves #0001 to the word at that address.
	DR offset	The signed binary content of the Data Register is added to the I/O memory address contained in IR□ and the resulting address is used as the operand.	DR0 ,IR0 DR0 ,IR1	LD DR0 ,IR0 Adds the content of DR0 to the I/O memory address contained in IR0 and loads the status of the bit at that address. MOV #0001 DR0 ,IR1 Adds the content of DR0 to the I/O memory address contained in IR1 and moves #0001 to the word at that address.
	Auto-increment	After the I/O memory address is read from IR□, the content of the Index Register is incremented by one or two. Increment by 1: ,R□+ Increment by 2: ,IR□++ **Note** Index registers will be incremented when the instruction is executed even if an error occurs and the Error Flag turns ON.	,IR0 + + ,IR1 +	LD ,IR0 + + Loads the status of the bit at the I/O memory address contained in IR0 and then increments the register by two. MOV #0001 ,IR1 + Moves #0001 to the word at the I/O memory address contained in IR1 and then increments the register by one.
	Auto-decrement	The content of IR□ is decremented by one or two and then the I/O memory address in the register is used as the operand. Decrement by 1: ,– IR□ Decrement by 2: ,– –IR□ **Note** Index registers will be decremented when the instruction is executed even if an error occurs and the Error Flag turns ON.	, – – IR0 , – IR1	LD , – – IR0 Decrements the content of IR0 by two and then loads the status of the bit at that I/O memory address. MOV #0001 , – IR1 Decrements the content of IR0 by one and then moves #0001 to the word at that I/O memory address.

Note Make sure that the contents of index registers indicate valid I/O memory addresses.

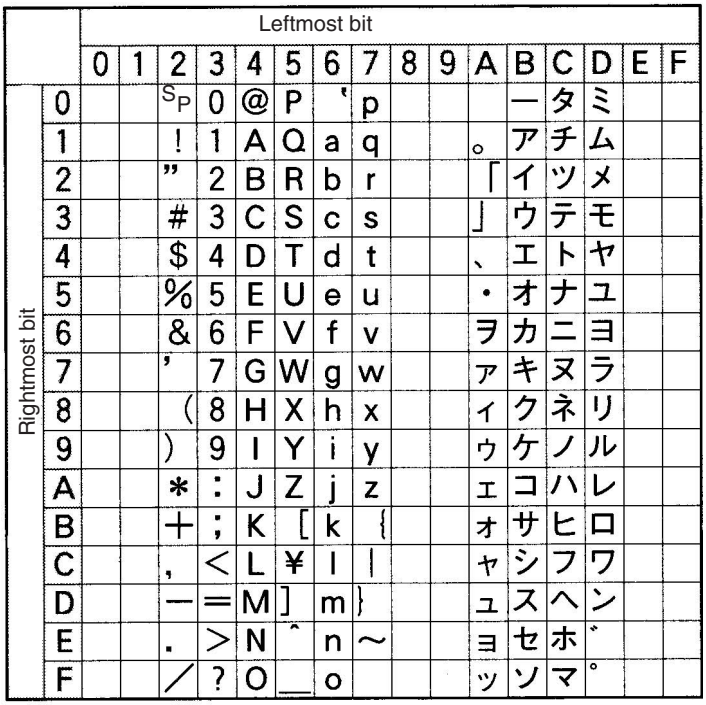
Specifying Constants

Method	Applicable operands	Data format	Code	Range	Example
Constant (16-bit data)	All binary data and binary data within a range	Unsigned binary	#	#0000 to #FFFF	MOV #0100 D00000 Stores #0100 hex (&256 decimal) in D00000. +#0009 #0001 D00001 Stores #000A hex (&10 decimal) in D00001.
		Signed decimal	±	–32,768 to +32,767	MOV –100 D00000 Stores –100 decimal (#FF9C hex) in D00000. +–9 –1 D00001 Stores –10 decimal (#FFF6 hex) in D00001.
		Unsigned decimal	&	&0 to &66,535	MOV &256 D00000 Stores –256 decimal (#0100 hex) in D00000. +&9 &1 D00001 Stores –10 decimal (#000A hex) in D00001.
	All BCD data and BCD data within a range	BCD	#	#0000 to #9999	MOV #0100 D00000 Stores #0100 (BCD) in D00000. +B #0009 #0001 D00001 Stores #0010 (BCD) in D00001.
Constant (32-bit data)	All binary data and binary data within a range	Unsigned binary	#	#0000 0000 to #FFFF FFFF	MOVL #12345678 D00000 Stores #12345678 hex in D00000 and D00001. D0001 D00000 1234 5678
		Signed decimal	+ –	–2,147,483,648 to +2,147,483,647	MOVL –12345678 D00000 Stores –12345678 decimal in D00000 and D00001.
		Unsigned decimal	&	&0 to &4,294,967,295	MOVL &12345678 D00000 Stores &12345678 decimal in D00000 and D00001.
	All BCD data and BCD data within a range	BCD	#	#0000 0000 to #9999 9999	MOVL #12345678 D00000 Stores #12345678 (BCD) in D00000 and D00001

Specifying Text Strings

Method	Description	Code	Examples	Instruction example
Text strings	Text is stored in ASCII (1 byte/character excluding special characters) starting with the lower byte of the lowest word in the range. If there is an odd number of characters, 00 (NULL) is stored in the higher byte of the last word in the range. If there is an even number of characters, 0000 (two NULLs) are stored in the word after the last in the range.		"ABCDE" "A" "B" "C" "D" "E" NUL || 41 42 43 44 45 00 "ABCD" "A" "B" "C" "D" NUL NUL || 41 42 43 44 00 00	MOV$ D00100 D00200 D00100 41 42 D00101 43 44 D00102 45 00 ↓ D00200 41 42 D00201 43 44 D00202 45 00

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



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

Note 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))

Classification	Instructions
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 000013 is stored in IR0.

The input condition is OFF (P_Off is the Always OFF Flag), so the OUT instruction sets 000013, which is indirectly addressed by IR0, to OFF. The OUT instruction is executed, so IR0 is incremented. As a result, the PLC memory address 000014, which was incremented by +1 in the IR0, is stored. Therefore, in the following cycle the OUT instruction turns OFF 000014.

Example 2

Ladder Program:

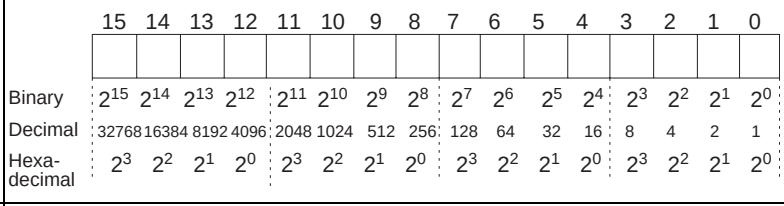
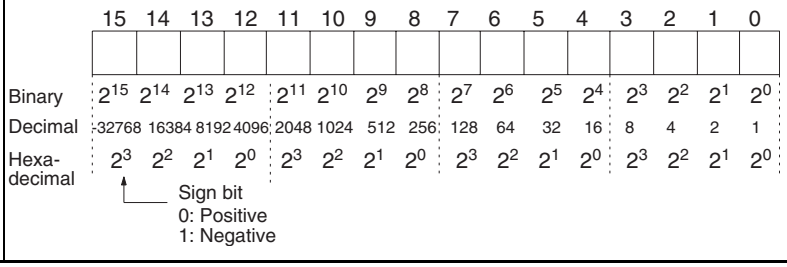
```
LD P_Off
SET, IR0+
```

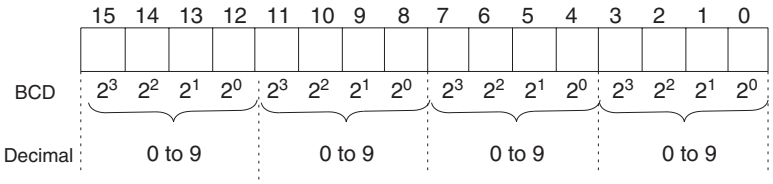
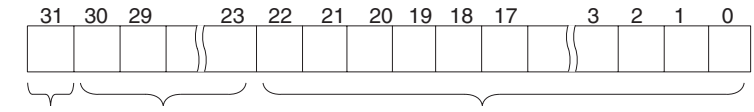
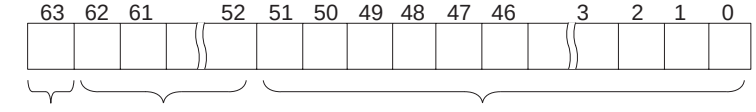
Operation: When the PLC memory address 000013 is stored in IR0.

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 000013.

1-1-6 Data Formats

The following table shows the data formats that can be used in CS/CJ-series PLCs.

Name	Format	Decimal range	Hexadecimal range
Unsigned binary data		0 to 65,535	0000 to FFFF
Signed binary data		–32,768 to +32,767	8000 to 7FFF

Name	Format	Decimal range	Hexadecimal range
BCD data		0 to 9,999	0000 to 9999
Floating-point decimal	 Sign of mantissa Exponent Mantissa Binary Value = $(-1)^{Sign} \times 1.[Mantissa] \times 2^{Exponent}$ Sign (bit 31) 1: negative or 0: positive Mantissa The mantissa includes 23 bits from bit 00 to bit 22 and indicates this portion below the decimal point in 1.□□□..... in binary. Exponent The exponent includes 8 bits from bit 23 to bit 30 and indicates n plus 127 in 2^n in binary. **Note** This format conforms to IEEE754 standards for single-precision floating-point data and is used only with instructions that convert or calculate floating-point data. It can be used to set or monitor from the I/O memory Edit and Monitor Screen on the CX-Programmer (not supported by the Programming Consoles). 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	 Sign of mantissa Exponent Mantissa Binary 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. **Note** This format conforms to IEEE754 standards for double-precision floating-point data and is used only with instructions that convert or calculate floating-point data. It can be used to set or monitor from the I/O memory Edit and Monitor Screen on the CX-Programmer (not supported by the Programming Consoles). As such, users do not need to know this format although they do need to know that the formatting takes up four words.	---	---

Signed Binary Numbers

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

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

1-2 Instruction Execution Checks

1-2-1 Errors Occurring at Instruction Execution

An instruction's operands and placement are checked when an instruction is input from a Peripheral Device or a program check is performed from a Peripheral Device (other than a Programming Console), but these are not final checks. The following four errors can occur when an instruction is executed.

Instruction Processing Error (ER Flag ON)

Normally, Instruction Processing Errors are non-fatal errors, but the PLC Setup can be set to treat Instruction Processing Errors as fatal errors. If this setting has been made, the Instruction Processing Error Flag (A29508) will be turned ON and program execution will stop when an Instruction Processing Error occurs.

Access Error (AER Flag ON)

Normally, Access Errors are non-fatal errors, but the PLC Setup can be set to treat these errors as fatal errors. If this setting has been made, the Illegal Access Error Flag (A29510) and the Indirect DM/EM BCD Error Flag (A29509) will be turned ON and program execution will stop when an Access Error occurs.

Illegal Instruction Error

The Illegal Instruction Error Flag (A29514) will be turned ON and program execution will stop when this error occurs.

UM (User Program Memory) Overflow Error

The UM Overflow Error Flag (A29515) will be turned ON and program execution will stop when this error occurs.

1-2-2 Fatal Errors (Program Errors)

Program execution will be stopped when one of the following program errors occurs. When a program error has occurred, the task number of the task that was being executed when program execution was stopped is written to A294 and the program address is written to A298 and A299.

Use the contents of these words to locate the program error and correct it as necessary.

Address	Description
A294	The task number of the current task is written to this word when program execution is stopped because of a program error. Cyclic tasks have task numbers 0000 to 001F (cyclic tasks 0 to 31). Interrupt tasks have task numbers 8000 to 80FF (interrupt tasks 0 to 255).
A298 and A299	The current program address is written to these words when program execution is stopped because of a program error. A299 contains the leftmost digits of the program address and A298 contains the rightmost digits of the program address.

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

Error type	Description	Related flags
No END Instruction	There is no END(001) instruction in the program.	No END Error Flag (A29511)
Task Error	There are three possible causes of a task error: 1) There is not an executable cyclic task. 2) There is not a program allocated to the task. 3) An interrupt was generated but the corresponding interrupt task does not exist.	Task Error Flag (A29512)
Instruction Processing Error*	The CPU attempted to execute an instruction, but the data provided in the instruction's operand was incorrect. *If the PLC Setup has been set to treat instruction errors as fatal errors (program errors), the Instruction Processing Error Flag (A29508) will be turned ON and program execution will stop.	Error (ER) Flag, Instruction Processing Error Flag (A29508)
Access Error*	There are five possible causes of an access error: 1) Reading/writing to the parameter area. 2) Writing to memory that is not installed. 3) Reading/writing to an EM bank that is EM file memory. 4) Writing to a read-only area. 5) The contents of a DM/EM word was not BCD although the PLC is set for BCD indirect addressing. *If the PLC Setup has been set to treat instruction errors as fatal errors (program errors), the Illegal Access Error Flag (A29510) will be turned ON and program execution will stop.	Access Error (AER) Flag, Illegal Access Error Flag (A29510)
Indirect DM/EM BCD Error*	The contents of a DM/EM word was not BCD although the PLC is set for BCD indirect addressing. *If the PLC Setup has been set to treat instruction errors as fatal errors (program errors), the Indirect DM/EM BCD Error Flag (A29509) will be turned ON and program execution will stop.	Access Error (AER) Flag, Indirect DM/EM BCD Error Flag (A29509)
Differentiation Overflow Error	Differentiated instructions were repeatedly inserted and deleted during online editing (over 31,072 times).	Differentiation Overflow Error Flag (A29513)
UM Overflow Error	The last address in UM (user program memory) has been exceeded.	UM Overflow Error Flag (A29515)
Illegal Instruction Error	The program contains an instruction that cannot be executed.	Illegal Instruction Error Flag (A29514)

SECTION 2
Summary of Instructions

This section provides a summary of instructions used with CS/CJ-series PLCs.

2-1 Instruction Classifications by Function

The following table lists the CS/CJ-series instructions by function. (The instructions appear by order of their function in *Section 3 Instructions*.)

*Instructions or instruction groups marked with a single asterisk are supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

**Instructions or instruction groups marked with two asterisks are supported by CJ1M CPU Units only.

***Instructions or instruction groups marked with three asterisks are not supported by CS1D CPU Units for Duplex-CPU Systems.

Note
1. CS/CJ-series CPU Unit Ver. 2.0 or later only
2. CJ1-H-R CPU Units only.
3. CJ1M-CPU21/22/23 CPU Unit Ver. 2.0 or later only
4. CS/CJ-series CPU Unit Ver. 2.0 or later only
   CJ1M CPU Unit (Pre-Ver. 2.0 or Unit Ver. 2.0 or later)

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Basic instructions	**Input**	LD	LOAD	LD NOT	LOAD NOT	AND	AND
		AND NOT	AND NOT	OR	OR	OR NOT	OR NOT
		AND LD	AND LOAD	OR LD	OR LOAD	---	---
	Output	OUT	OUTPUT	OUT NOT	OUTPUT NOT	---	---
Sequence input instructions	**---**	NOT	NOT	UP	CONDITION ON	DOWN	CONDITION OFF
	Bit test	LD TST	LD BIT TEST	LD TSTN	LD BIT TEST NOT	AND TST	AND BIT TEST NOT
		AND TSTN	AND BIT TEST NOT	OR TST	OR BIT TEST	OR TSTN	OR BIT TEST NOT
Sequence output instructions	**---**	KEEP	KEEP	DIFU	DIFFERENTIATE UP	DIFD	DIFFERENTIATE DOWN
		OUTB*	SINGLE BIT OUTPUT	---	---	---	---
	Set/Reset	SET	SET	RSET	RESET	SETA	MULTIPLE BIT SET
		RSTA	MULTIPLE BIT RESET	SETB*	SINGLE BIT SET	RSTB*	SINGLE BIT RESET
Sequence control instructions	**---**	END	END	NOP	NO OPERATION	---	---
	Interlock	IL	INTERLOCK	ILC	INTERLOCK CLEAR	MILH	MULTI-INTERLOCK DIFFERENTIATION HOLD
		MILR (See note 1.)	MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILC (See note 1.)	MULTI-INTERLOCK CLEAR	---	---
	Jump	JMP	JUMP	JME	JUMP END	CJP	CONDITIONAL JUMP
		CJPN	CONDITIONAL JUMP	JMP0	MULTIPLE JUMP	JME0	MULTIPLE JUMP END
	Repeat	FOR	FOR-NEXT LOOPS	BREAK	BREAK LOOP	NEXT	FOR-NEXT LOOPS

Classification	Sub-class		Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Timer and counter instructions	**BCD**	**Timer (with timer numbers)**	TIM	HUNDRED-MS TIMER	TIMH	TEN-MS TIMER	TMHH	ONE-MS TIMER
			TIMU (See note 2.)	TENTH-MS TIMER	TMUH (See note 2.)	HUN-DREDTH-MS TIMER	TTIM	ACCUMULA-TIVE TIMER
		Timer (without timer numbers)	TIML	LONG TIMER	MTIM	MULTI-OUT-PUT TIMER	---	---
		Counter (with counter numbers)	CNT	COUNTER	CNTR	REVERSIBLE TIMER	CNR	RESET TIMER/ COUNTER
	Binary*	**Timer (with timer numbers)**	TIMX	HUNDRED-MS TIMER	TIMHX	TEN-MS TIMER	TMHHX	ONE-MS TIMER
			TIMUX (See note 2.)	TENTH-MS TIMER	TMUHX (See note 2.)	HUN-DREDTH-MS TIMER	TTIMX	ACCUMULA-TIVE TIMER
		Timer (without timer numbers)	TIMLX	LONG TIMER	MTIMX	MULTI-OUT-PUT TIMER	---	---
		Counter (with counter numbers)	CNTX	COUNTER	CNTRX	REVERSIBLE TIMER	CNRX	RESET TIMER/ COUNTER
Comparison instructions	**Symbol comparison**		LD, AND, OR + =, <>, <, <=, >, >=	Symbol comparison (unsigned)	LD, AND, OR + =, <>, <, <=, >, >= + L	Symbol comparison (double-word, unsigned)	LD, AND, OR + =, <>, <, <=, >, >= +S	Symbol comparison (signed)
			LD, AND, OR + =, <>, <, <=, >, >= + SL	Symbol comparison (double-word, signed)	LD, AND, OR + = DT, <> DT, < DT, <= DT, > DT, >= DT (See note 1.)	Time comparison	---	---
	Data comparison (Condition Flags)		CMP	UNSIGNED COMPARE	CMPL	DOUBLE UNSIGNED COMPARE	CPS	SIGNED BINARY COMPARE
			CPSL	DOUBLE SIGNED BINARY COMPARE	ZCP*	AREA RANGE COMPARE	ZCPL*	DOUBLE AREA RANGE COMPARE
	Table compare		MCMP	MULTIPLE COMPARE	TCMP	TABLE COM-PARE	BCMP	UNSIGNED BLOCK COM-PARE
			BCMP2 (See note 3.)	EXPANDED BLOCK COM-PARE	---	---	---	---
Data movement instructions	**Single/ double-word**		MOV	MOVE	MOVL	DOUBLE MOVE	MVN	MOVE NOT
			MVNL	DOUBLE MOVE NOT	---	---	---	---
	Bit/digit		MOVB	MOVE BIT	MOVD	MOVE DIGIT	---	---
	Exchange		XCHG	DATA EXCHANGE	XCGL	DOUBLE DATA EXCHANGE	---	---
	Block/bit transfer		XFRB	MULTIPLE BIT TRANS-FER	XFER	BLOCK TRANSFER	BSET	BLOCK SET
	Distribute/ collect		DIST	SINGLE WORD DIS-TRIBUTE	COLL	DATA COL-LECT	---	---
	Index register		MOVR	MOVE TO REGISTER	MOVRW	MOVE TIMER/ COUNTER PV TO REGIS-TER	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Data shift instructions	**1-bit shift**	SFT	SHIFT REGISTER	SFTR	REVERSIBLE SHIFT REGISTER	ASLL	DOUBLE SHIFT LEFT
		ASL	ARITHMETIC SHIFT LEFT	ASR	ARITHMETIC SHIFT RIGHT	ASRL	DOUBLE SHIFT RIGHT
	0000 hex asynchronous	ASFT	ASYNCHRONOUS SHIFT REGISTER	---	---	---	---
	Word shift	WSFT	WORD SHIFT	---	---	---	---
	1-bit rotate	ROL	ROTATE LEFT	ROLL	DOUBLE ROTATE LEFT	RLNC	ROTATE LEFT WITHOUT CARRY
		RLNL	DOUBLE ROTATE LEFT WITHOUT CARRY	ROR	ROTATE RIGHT	RORL	DOUBLE ROTATE RIGHT
		RRNC	ROTATE RIGHT WITHOUT CARRY	RRNL	DOUBLE ROTATE RIGHT WITHOUT CARRY	---	---
	1 digit shift	SLD	ONE DIGIT SHIFT LEFT	SRD	ONE DIGIT SHIFT RIGHT	---	---
	Shift n-bit data	NSFL	SHIFT N-BIT DATA LEFT	NSFR	SHIFT N-BIT DATA RIGHT	---	---
	Shift n-bit	NASL	SHIFT N-BITS LEFT	NSLL	DOUBLE SHIFT N-BITS LEFT	NASR	SHIFT N-BITS RIGHT
		NSRL	DOUBLE SHIFT N-BITS RIGHT	---	---	---	---
Increment/ decrement instructions	**BCD**	++B	INCREMENT BCD	++BL	DOUBLE INCREMENT BCD	– –B	DECREMENT BCD
		– –BL	DOUBLE DECREMENT BCD	---	---	---	---
	Binary	++	INCREMENT BINARY	++L	DOUBLE INCREMENT BINARY	– –	DECREMENT BINARY
		– –L	DOUBLE DECREMENT BINARY	---	---	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Symbol math instructions	**Binary add**	+	SIGNED BINARY ADD WITHOUT CARRY	+L	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+C	SIGNED BINARY ADD WITH CARRY
		+CL	DOUBLE SIGNED BINARY ADD WITH CARRY	---	---	---	---
	BCD add	+B	BCD ADD WITHOUT CARRY	+BL	DOUBLE BCD ADD WITHOUT CARRY	+BC	BCD ADD WITH CARRY
		+BCL	DOUBLE BCD ADD WITH CARRY	---	---	---	---
	Binary subtract	–	SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–C	SIGNED BINARY SUBTRACT WITH CARRY
		–CL	DOUBLE SIGNED BINARY WITH CARRY	---	---	---	---
	BCD subtract	–B	BCD SUBTRACT WITHOUT CARRY	–BL	DOUBLE BCD SUBTRACT WITHOUT CARRY	–BC	BCD SUBTRACT WITH CARRY
		–BCL	DOUBLE BCD SUBTRACT WITH CARRY	---	---	---	---
	Binary multiply	*	SIGNED BINARY MULTIPLY	*L	DOUBLE SIGNED BINARY MULTIPLY	*U	UNSIGNED BINARY MULTIPLY
		*UL	DOUBLE UNSIGNED BINARY MULTIPLY	---	---	---	---
	BCD multiply	*B	BCD MULTIPLY	*BL	DOUBLE BCD MULTIPLY	---	---
	Binary divide	/	SIGNED BINARY DIVIDE	/L	DOUBLE SIGNED BINARY DIVIDE	/U	UNSIGNED BINARY DIVIDE
		/UL	DOUBLE UNSIGNED BINARY DIVIDE	---	---	---	---
	BCD divide	/B	BCD DIVIDE	/BL	DOUBLE BCD DIVIDE	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Conversion instructions	**BCD-binary conversions**	BIN	BCD TO BINARY	BINL	DOUBLE BCD TO DOUBLE BINARY	BCD	BINARY TO BCD
		BCDL	DOUBLE BINARY TO DOUBLE BCD	NEG	2'S COMPLEMENT	NEGL	DOUBLE 2'S COMPLEMENT
		SIGN	16-BIT TO 32-BIT SIGNED BINARY	---	---	---	---
	Decoder/ encoder	MLPX	DATA DECODER	DMPX	DATA ENCODER	---	---
	ASCII-hexadecimal conversions	ASC	ASCII CONVERT	HEX	ASCII TO HEX	---	---
	Line-column conversions	LINE	COLUMN TO LINE	COLM	LINE TO COLUMN	---	---
	Signed binary-BCD conversions	BINS	SIGNED BCD TO BINARY	BISL	DOUBLE SIGNED BCD TO BINARY	BCDS	SIGNED BINARY TO BCD
		BDSL	DOUBLE SIGNED BINARY TO BCD	GRY (See note 1.)	GRAY CODE CONVERSION	---	---
	Number-ASCII conversions	STR4	FOUR-DIGIT NUMBER TO ASCII	STR8	EIGHT-DIGIT NUMBER TO ASCII	STR16	SIXTEEN-DIGIT NUMBER TO ASCII
		NUM4	ASCII TO FOUR-DIGIT NUMBER	NUM8	ASCII TO EIGHT-DIGIT NUMBER	NUM16	ASCII TO SIXTEEN-DIGIT NUMBER
Logic instructions	**Logical AND/OR**	ANDW	LOGICAL AND	ANDL	DOUBLE LOGICAL AND	ORW	LOGICAL OR
		ORWL	DOUBLE LOGICAL OR	XORW	EXCLUSIVE OR	XORL	DOUBLE EXCLUSIVE OR
		XNRW	EXCLUSIVE NOR	XNRL	DOUBLE EXCLUSIVE NOR	---	---
	Complement	COM	COMPLEMENT	COML	DOUBLE COMPLEMENT	---	---
Special math instructions	**---**	ROTB	BINARY ROOT	ROOT	BCD SQUARE ROOT	APR	ARITHMETIC PROCESS
		FDIV	FLOATING POINT DIVIDE	BCNT	BIT COUNTER	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Floating-point math instructions	**Floating point/ binary convert**	FIX	FLOATING TO 16-BIT	FIXL	FLOATING TO 32-BIT	FLT	16-BIT TO FLOATING
		FLTL	32-BIT TO FLOATING	---	---	---	---
	Floating- point basic math	+F	FLOATING-POINT ADD	–F	FLOATING-POINT SUBTRACT	/F	FLOATING-POINT DIVIDE
		*F	FLOATING-POINT MULTIPLY	---	---	---	---
	High-speed trigonometric functions (See note 2.)	SINQ	HIGH-SPEED SINE	CONQ	HIGH-SPEED COSINE	TANQ	HIGH-SPEED TANGENT
	Floating- point trigonometric functions	RAD	DEGREES TO RADIANS	DEG	RADIANS TO DEGREES	SIN	SINE
		COS	COSINE	TAN	TANGENT	ASIN	ARC SINE
		ACOS	ARC COSINE	ATAN	ARC TANGENT	---	---
	Floating- point math	SQRT	SQUARE ROOT	EXP	EXPONENT	LOG	LOGARITHM
		PWR	EXPONENTIAL POWER	---	---	---	---
	Symbol comparison and conversion*	LD, AND, OR + =, <>, <, <=, >, >= + F	Symbol comparison (single-precision floating point)	FSTR*	FLOATING-POINT TO ASCII	FVAL*	ASCII TO FLOATING-POINT
	Single-precision floating point move (See note 2.)	MOVF	MOVE FLOATING-POINT (SINGLE)	---	---	---	---
Double-precision floating- point instructions*	**Floating point/ binary convert**	FIXD	DOUBLE FLOATING TO 16-BIT	FIXLD	DOUBLE FLOATING TO 32-BIT	DBL	16-BIT TO DOUBLE FLOATING
		DBLL	32-BIT TO DOUBLE FLOATING	---	---	---	---
	Floating- point basic math	+D	DOUBLE FLOATING-POINT ADD	–D	DOUBLE FLOATING-POINT SUBTRACT	/D	DOUBLE FLOATING-POINT DIVIDE
		*D	DOUBLE FLOATING-POINT MULTIPLY	---	---	---	---
	Floating- point trigonometric functions	RADD	DOUBLE DEGREES TO RADIANS	DEGD	DOUBLE RADIANS TO DEGREES	SIND	DOUBLE SINE
		COSD	DOUBLE COSINE	TAND	DOUBLE TANGENT	ASIND	DOUBLE ARC SINE
		ACOSD	DOUBLE ARC COSINE	ATAND	DOUBLE ARC TANGENT	---	---
	Floating- point math	SQRTD	DOUBLE SQUARE ROOT	EXPD	DOUBLE EXPONENT	LOGD	DOUBLE LOGARITHM
		PWRD	DOUBLE EXPONENTIAL POWER	---	---	---	---
	Symbol comparison	LD, AND, OR + =, <>, <, <=, >, >= + D	Symbol comparison (double-precision floating point)	---	---	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Table data processing instructions	**Stack processing**	SSET	SET STACK	PUSH	PUSH ONTO STACK	LIFO	LAST IN FIRST OUT
		FIFO	FIRST IN FIRST OUT	SNUM*	STACK SIZE READ	SREAD*	STACK DATA READ
		SWRIT*	STACK DATA OVERWRITE	SINS*	STACK DATA INSERT	SDEL*	STACK DATA DELETE
	1-record/ multiple-word processing	DIM	DIMENSION RECORD TABLE	SETR	SET RECORD LOCATION	GETR	GET RECORD NUMBER
	Record-to- word processing	SRCH	DATA SEARCH	MAX	FIND MAXIMUM	MIN	FIND MINIMUM
		SUM	SUM	FCS	FRAME CHECKSUM	---	---
	Byte processing	SWAP	SWAP BYTES	---	---	---	---
Data control instructions	**---**	PID	PID CONTROL	PIDAT*	PID CONTROL WITH AUTOTUNING	LMT	LIMIT CONTROL
		BAND	DEAD BAND CONTROL	ZONE	DEAD ZONE CONTROL	TPO (See note 1.)	TIME-PROPORTIONAL OUTPUT
		SCL	SCALING	SCL2	SCALING 2	SCL3	SCALING 3
		AVG	AVERAGE	---	---	---	---
Subroutines instructions	**---**	SBS	SUBROUTINE CALL	MCRO	MACRO	SBN	SUBROUTINE ENTRY
		RET	SUBROUTINE RETURN	GSBS*	GLOBAL SUBROUTINE CALL	GSBN*	GLOBAL SUBROUTINE ENTRY
		GRET*	GLOBAL SUBROUTINE RETURN	---	---	---	---
Interrupt control instructions	**---**	MSKS***	SET INTERRUPT MASK	MSKR***	READ INTERRUPT MASK	CLI***	CLEAR INTERRUPT
		DI	DISABLE INTERRUPTS	EI	ENABLE INTERRUPTS	---	---
High-speed counter/ pulse output instructions**	**---**	INI	MODE CONTROL	PRV	HIGH-SPEED COUNTER PV READ	PRV2 (See note 2.)	COUNTER FREQUENCY CONVERT
		CTBL	COMPARISON TABLE LOAD	SPED	SPEED OUTPUT	PULS	SET PULSES
		PLS2	PULSE OUTPUT	ACC	ACCELERATION Control	ORG	ORIGIN SEARCH
Step instructions	**---**	PWM	PULSE WITH VARIABLE DUTY FACTOR	STEP	STEP DEFINE	SNXT	STEP START
Basic I/O Unit instructions	**---**	IORF	I/O REFRESH	FIORF (See note 2.)	SPECIAL I/O UNIT I/O REFRESH	DLNK*	CPU BUS UNIT I/O REFRESH
		SDEC	7-SEGMENT DECODER	DSW (See note 1.)	DIGITAL SWITCH INPUT	TKY (See note 1.)	TEN KEY INPUT
		HKY (See note 1.)	HEXADECIMAL KEY INPUT	MTR (See note 1.)	MATRIX INPUT	7SEG (See note 1.)	7-SEGMENT DISPLAY OUTPUT
		IORD	INTELLIGENT I/O READ	IOWR	INTELLIGENT I/O WRITE	DLNK*	CPU BUS UNIT I/O REFRESH

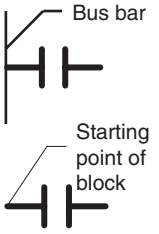
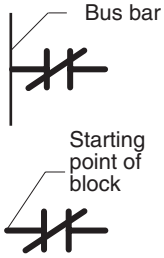
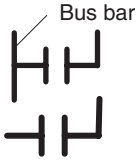
Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Serial communications instructions	**---**	PMCR	PROTOCOL MACRO	TXD	TRANSMIT	RXD	RECEIVE
		STUP	CHANGE SERIAL PORT SETUP	---	---	---	---
Network instructions	**---**	SEND	NETWORK SEND	RECV	NETWORK RECEIVE	CMND	DELIVER COMMAND
		EXPLT (See note 1.)	SEND GENERAL EXPICIT	EGATR (See note 1.)	EXPLICIT GET ATTRIBUTE	ESATR (See note 1.)	EXPLICIT SET ATTRIBUTE
		ECHRD (See note 1.)	EXPLICIT WORD READ	ECHWR (See note 1.)	EXPLICIT WORD WRITE	---	---
Display instructions	**---**	MSG	DISPLAY MESSAGE	---	---	---	---
File memory instructions	**---**	FREAD	READ DATA FILE	FWRIT	WRITE DATA FILE	TWRIT	WRITE TEXT FILE
Clock instructions	**---**	CADD	CALENDAR ADD	CSUB	CALENDAR SUBTRACT	SEC	HOURS TO SECONDS
		HMS	SECONDS TO HOURS	DATE	CLOCK ADJUSTMENT	---	---
Debugging instructions	**---**	TRSM	TRACE MEMORY SAMPLING	---	---	---	---
Failure diagnosis instructions	**---**	FAL	FAILURE ALARM	FALS	SEVERE FAILURE ALARM	FPD	FAILURE POINT DETECTION
Other instructions	**---**	STC	SET CARRY	CLC	CLEAR CARRY	EMBC	SELECT EM BANK
		WDT	EXTEND MAXIMUM CYCLE TIME	CCS*	SAVE CONDITION FLAGS	CCL*	LOAD CONDITION FLAGS
		FRMCV*	CONVERT ADDRESS FROM CV	TOCV*	CONVERT ADDRESS TO CV	IOSP***	DISABLE PERIPHERAL SERVICING
		IORS***	ENABLE PERIPHERAL SERVICING	---	---	---	---

Classification	Sub-class		Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Block programming instructions	**Define block program area**		BPRG	BLOCK PROGRAM BEGIN	BEND	BLOCK PROGRAM END	---	---
	Block program start/stop		BPPS	BLOCK PROGRAM PAUSE	BPRS	BLOCK PROGRAM RESTART	---	---
	EXIT		EXIT *bit_address*	Conditional END	EXIT NOT *bit_address*	Conditional END NOT	input_condition EXIT	Conditional END
	IF branch processing		IF *bit_address*	CONDITIONAL BLOCK BRANCHING	IF NOT *bit_address*	CONDITIONAL BLOCK BRANCHING (NOT)	ELSE	CONDITIONAL BLOCK BRANCHING (ELSE)
			IEND	CONDITIONAL BLOCK BRANCHING END	---	---	---	---
	WAIT		WAIT *bit_address*	ONE CYCLE AND WAIT	WAIT NOT *bit_address*	ONE CYCLE AND WAIT NOT	*input_condition* WAIT	ONE CYCLE AND WAIT
	Timer/ counter	**BCD**	TIMW	HUNDRED-MS TIMER WAIT	CNTW	COUNTER WAIT	TMHW	TEN-MS TIMER WAIT
		Binary*	TIMWX	HUNDRED-MS TIMER WAIT	CNTWX	COUNTER WAIT	TMHWX	TEN-MS TIMER WAIT
	Repeat		LOOP	LOOP BLOCK	LEND *bit_address*	LOOP BLOCK END	LEND NOT *bit_address*	LOOP BLOCK END NOT
			input_ condition LEND	LOOP BLOCK END	---	---	---	---
Text string processing instructions	**---**		MOV$	MOV STRING	+$	CONCATENATE STRING	LEFT$	GET STRING LEFT
			RIGHT$	GET STRING RIGHT	MID$	GET STRING MIDDLE	FIND$	FIND IN STRING
			LEN$	STRING LENGTH	RPLC$	REPLACE IN STRING	DEL$	DELETE STRING
			XCHG$	EXCHANGE STRING	CLR$	CLEAR STRING	INS$	INSERT INTO STRING
			LD, AND, OR + =$, <>$, <$, <=$, >$, >=$	STRING COMPARISON	---	---	---	---
Task control instructions	**---**		TKON	TASK ON	TKOF	TASK OFF	---	---

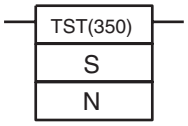
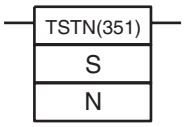
2-2 Instruction Functions

2-2-1 Sequence Input Instructions

*1: Not supported by CS1D CPU Units for Duplex-CPU Systems.
*2: Supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.
*3: Supported by CS1-H, CJ1-H, and CJ1M CPU Units only.

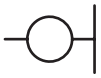
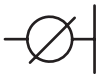
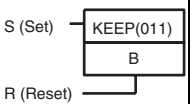
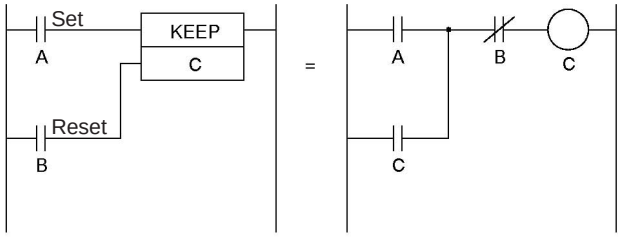
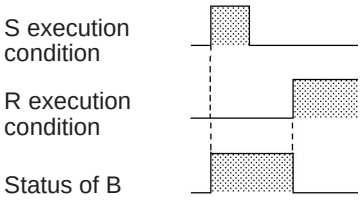
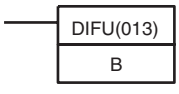
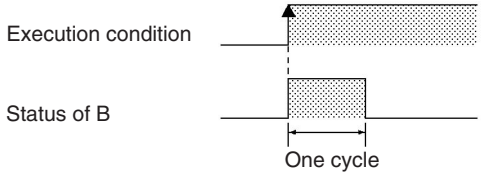
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
LOAD LD @LD %LD !LD*1 !@LD*1 !%LD*1		Indicates a logical start and creates an ON/OFF execution condition based on the ON/OFF status of the specified operand bit.	Start of logic Not required	161
LOAD NOT LD NOT @LD NOT*2 %LD NOT*2 !LD NOT*1 !@LD NOT*3 !%LD NOT*3		Indicates a logical start and creates an ON/OFF execution condition based on the reverse of the ON/OFF status of the specified operand bit.	Start of logic Not required	163
AND AND @AND %AND !AND*1 !@AND*1 !%AND*1		Takes a logical AND of the status of the specified operand bit and the current execution condition.	Continues on rung Required	165
AND NOT AND NOT @AND NOT*2 %AND NOT*2 !AND NOT*1 !@AND NOT*3 !%AND NOT*3		Reverses the status of the specified operand bit and takes a logical AND with the current execution condition.	Continues on rung Required	167
OR OR @OR %OR !OR*1 !@OR*1 !%OR*1		Takes a logical OR of the ON/OFF status of the specified operand bit and the current execution condition.	Continues on rung Required	169
OR NOT OR NOT @OR NOT*2 %OR NOT*2 !OR NOT*1 !@OR NOT*3 !%OR NOT*3		Reverses the status of the specified bit and takes a logical OR with the current execution condition	Continues on rung Required	171

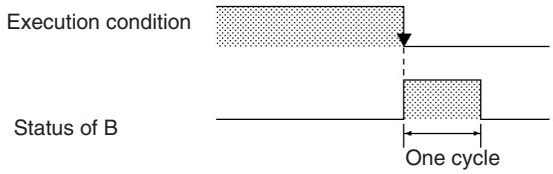
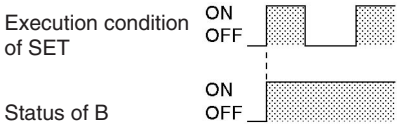
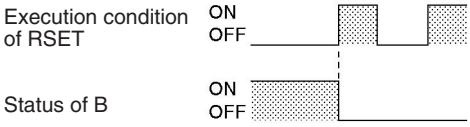
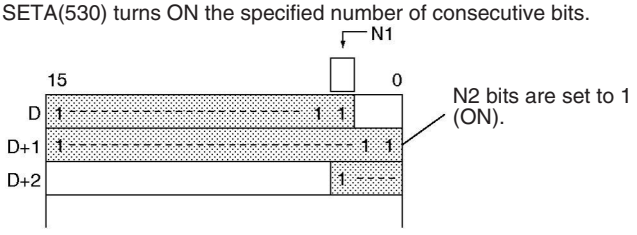
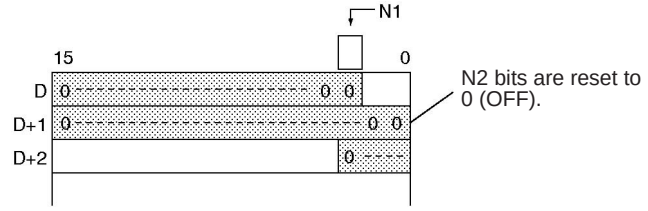
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
AND LOAD AND LD	Logic block Logic block	Takes a logical AND between logic blocks. LD to: Logic block A LD to: Logic block B AND LD ····· Serial connection between logic block A and logic block B.	Continues on rung Required	172
OR LOAD OR LD	Logic block Logic block	Takes a logical OR between logic blocks. LD to: Logic block A LD to: Logic block B OR LD ····· Parallel connection between logic block A and logic block B.	Continues on rung Required	174
NOT NOT 520	---	Reverses the execution condition.	Continues on rung Required	180
CONDITION ON UP 521	UP(521)	UP(521) turns ON the execution condition for one cycle when the execution condition goes from OFF to ON.	Continues on rung Required	181
CONDITION OFF DOWN 522	DOWN(522)	DOWN(522) turns ON the execution condition for one cycle when the execution condition goes from ON to OFF.	Continues on rung Required	181
BIT TEST LD TST 350	TST(350) S N **S**: Source word **N**: Bit number	LD TST(350), AND TST(350), and OR TST(350) are used in the program like LD, AND, and OR; the execution condition is ON when the specified bit in the specified word is ON and OFF when the bit is OFF.	Continues on rung Not required	182
BIT TEST LD TSTN 351	TSTN(351) S N **S**: Source word **N**: Bit number	LD TSTN(351), AND TSTN(351), and OR TSTN(351) are used in the program like LD NOT, AND NOT, and OR NOT; the execution condition is OFF when the specified bit in the specified word is ON and ON when the bit is OFF.	Continues on rung Not required	182
BIT TEST AND TST 350	AND TST(350) S N **S**: Source word **N**: Bit number	LD TST(350), AND TST(350), and OR TST(350) are used in the program like LD, AND, and OR; the execution condition is ON when the specified bit in the specified word is ON and OFF when the bit is OFF.	Continues on rung Required	182
BIT TEST AND TSTN 351	AND TSTN(351) S N **S**: Source word **N**: Bit number	LD TSTN(351), AND TSTN(351), and OR TSTN(351) are used in the program like LD NOT, AND NOT, and OR NOT; the execution condition is OFF when the specified bit in the specified word is ON and ON when the bit is OFF.	Continues on rung Required	182

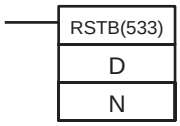
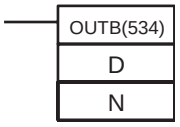
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BIT TEST OR TST 350	 **S**: Source word **N**: Bit number	LD TST(350), AND TST(350), and OR TST(350) are used in the program like LD, AND, and OR; the execution condition is ON when the specified bit in the specified word is ON and OFF when the bit is OFF.	Continues on rung Required	182
BIT TEST OR TSTN 351	 **S**: Source word **N**: Bit number	LD TSTN(351), AND TSTN(351), and OR TSTN(351) are used in the program like LD NOT, AND NOT, and OR NOT; the execution condition is OFF when the specified bit in the specified word is ON and ON when the bit is OFF.	Continues on rung Required	182

2-2-2 Sequence Output Instructions

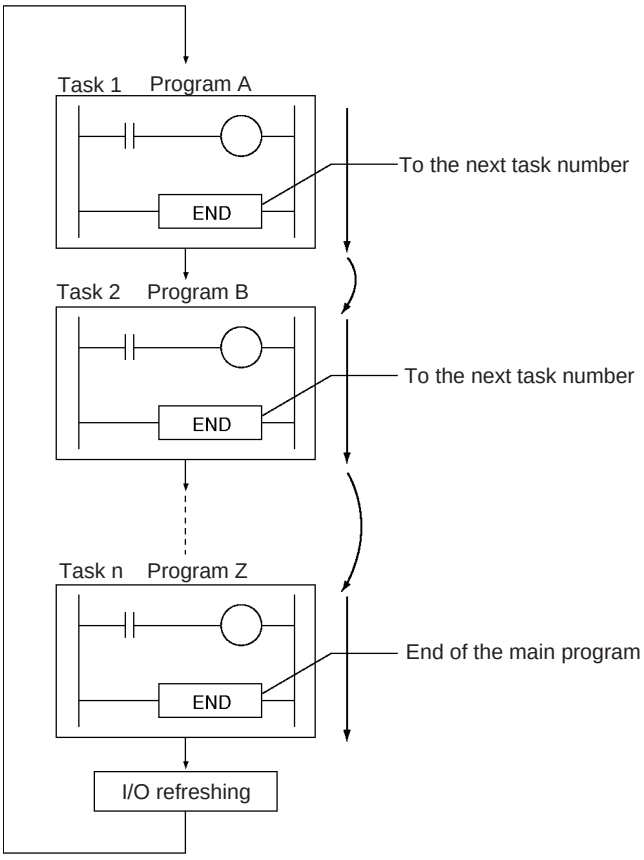
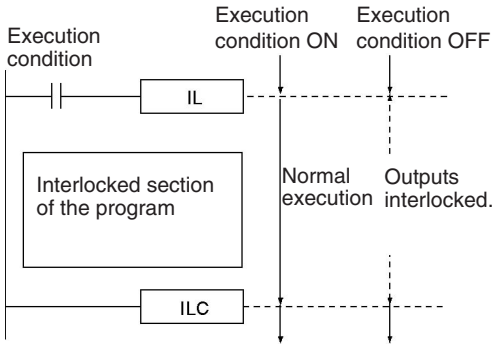
*1: Not supported by CS1D CPU Units for Duplex-CPU Systems.

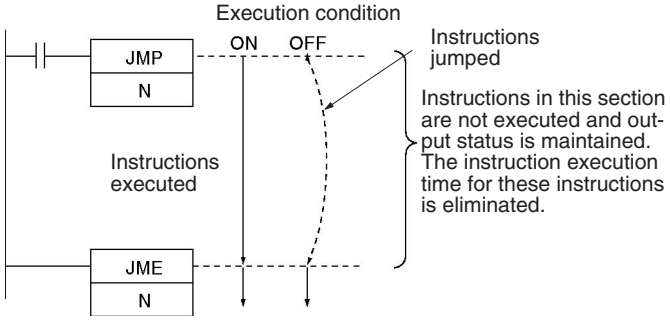
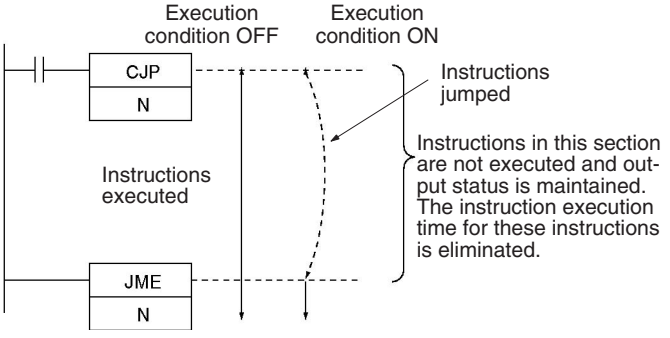
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
OUTPUT OUT !OUT*1		Outputs the result (execution condition) of the logical processing to the specified bit.	Output Required	185
OUTPUT NOT OUT NOT !OUT NOT*1		Reverses the result (execution condition) of the logical processing, and outputs it to the specified bit.	Output Required	187
KEEP KEEP !KEEP*1 011	 **B**: Bit	Operates as a latching relay.  	Output Required	188
DIFFERENTIATE UP DIFU !DIFU*1 013	 **B**: Bit	DIFU(013) turns the designated bit ON for one cycle when the execution condition goes from OFF to ON (rising edge). 	Output Required	193

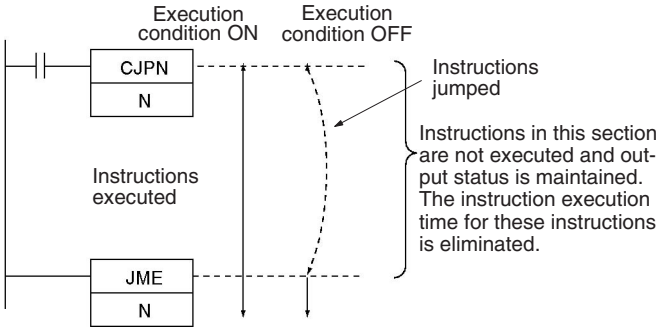
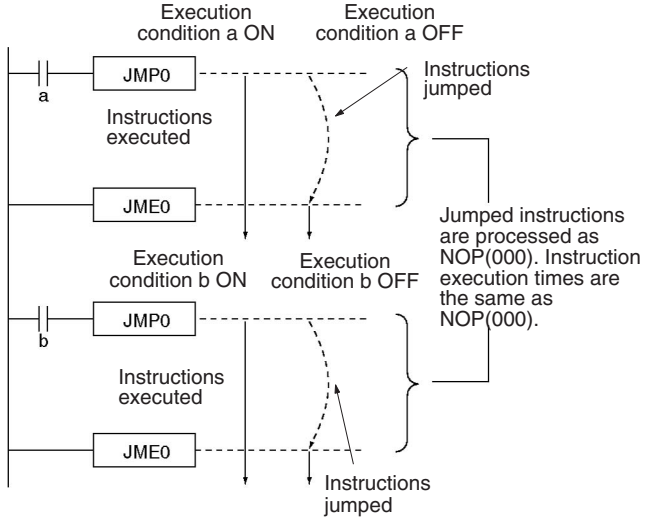
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DIFFERENTIATE DOWN DIFD !DIFD*1 014	DIFD(014) B **B**: Bit	DIFD(014) turns the designated bit ON for one cycle when the execution condition goes from ON to OFF (falling edge). 	Output Required	193
SET SET @SET %SET !SET*1 !@SET*1 !%SET*1	SET B **B**: Bit	SET turns the operand bit ON when the execution condition is ON. 	Output Required	195
RESET RSET @RSET %RSET !RSET*1 !@RSET*1 !%RSET*1	RSET B **B**: Bit	RSET turns the operand bit OFF when the execution condition is ON. 	Output Required	195
MULTIPLE BIT SET SETA @SETA 530	SETA(530) D N1 N2 **D**: Beginning word **N1**: Beginning bit **N2**: Number of bits	SETA(530) turns ON the specified number of consecutive bits. 	Output Required	198
MULTIPLE BIT RESET RSTA @RSTA 531	RSTA(531) D N1 N2 **D**: Beginning word **N1**: Beginning bit **N2**: Number of bits	RSTA(531) turns OFF the specified number of consecutive bits. 	Output Required	198
SINGLE BIT SET (CS1-H, CJ1-H, CJ1M, or CS1D only) SETB @SETB !SETB*1 !@SETB*1	SETB(532) D N **D**: Word address **N**: Bit number	SETB(532) turns ON the specified bit in the specified word when the execution condition is ON. Unlike the SET instruction, SETB(532) can be used to set a bit in a DM or EM word.	Output Required	201

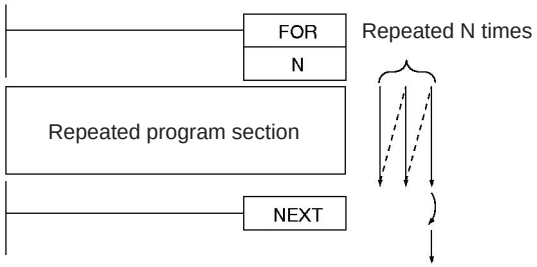
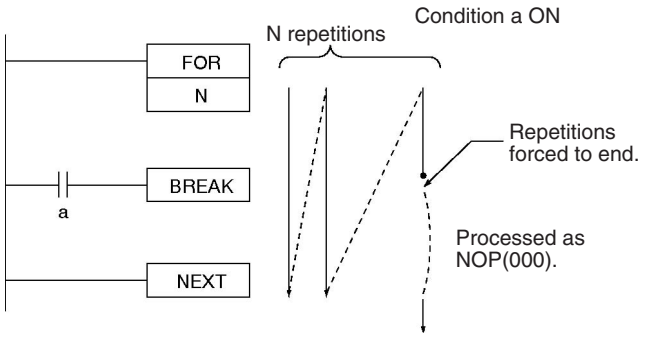
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SINGLE BIT RESET (CS1-H, CJ1-H, CJ1M, or CS1D only) RSTB @RSTB !RSTB*1 !@RSTB*1	 **D**: Word address **N**: Bit number	RSTB(533) turns OFF the specified bit in the specified word when the execution condition is ON. Unlike the RSET instruction, RSTB(533) can be used to reset a bit in a DM or EM word.	Output Required	201
SINGLE BIT OUTPUT (CS1-H, CJ1-H, CJ1M, or CS1D only) OUTB @OUTB !OUTB*1	 **D**: Word address **N**: Bit number	OUTB(534) outputs the result (execution condition) of the logical processing to the specified bit. Unlike the OUT instruction, OUTB(534) can be used to control a bit in a DM or EM word.	Output Required	204

2-2-3 Sequence Control Instructions

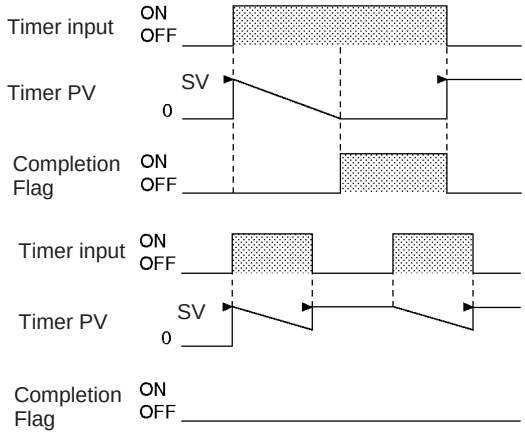
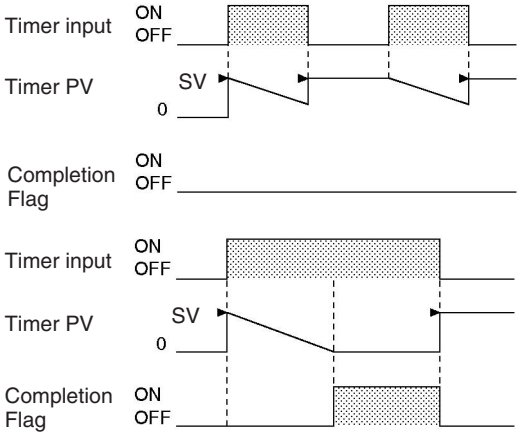
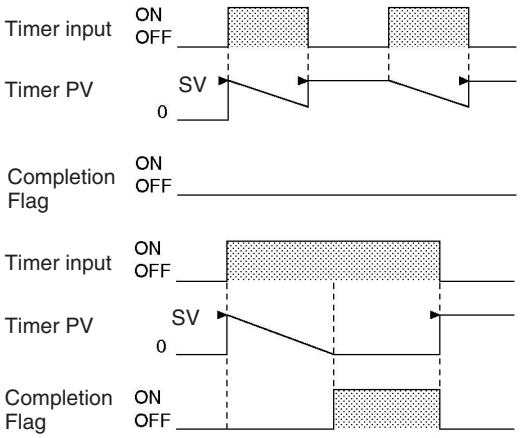
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
END END 001	END(001)	Indicates the end of a program. END(001) completes the execution of a program for that cycle. No instructions written after END(001) will be executed. Execution proceeds to the program with the next 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. 	Output Not required	206
NO OPERATION NOP 000		This instruction has no function. (No processing is performed for NOP(000).)	Output Not required	207
INTERLOCK IL 002	IL(002)	Interlocks all outputs between IL(002) and ILC(003) when the execution condition for IL(002) is OFF. IL(002) and ILC(003) are normally used in pairs. 	Output Required	210

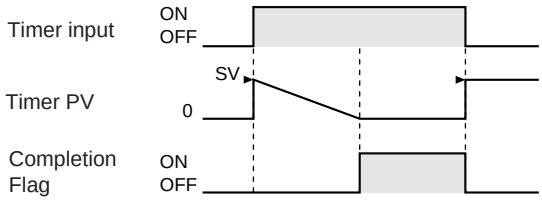
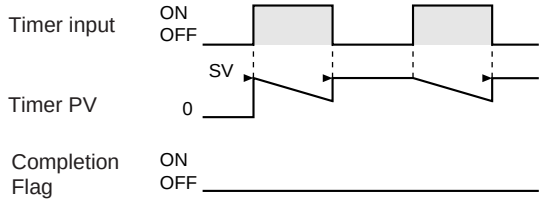
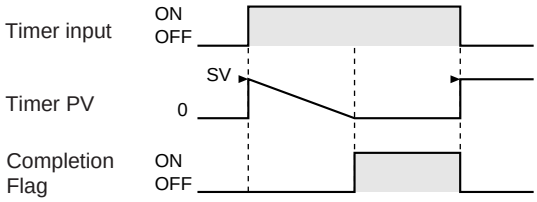
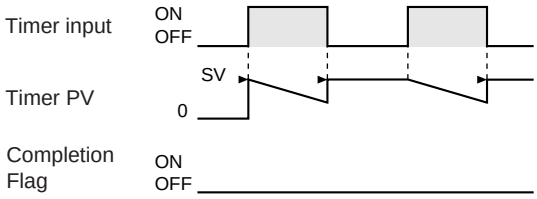
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
INTERLOCK CLEAR ILC 003	ILC(003)	All outputs between IL(002) and ILC(003) are interlocked when the execution condition for IL(002) is OFF. IL(002) and ILC(003) are normally used in pairs.	Output Not required	210
MULTI-INTERLOCK DIFFERENTIATION HOLD MILH 517 CS/CJ-series CPU Unit Ver. 2.0 or later only	MILH (517) N D **N**: Interlock number **D**: Interlock Status Bit	When the execution condition for MILH(517) is OFF, the outputs for all instructions between that MILH(517) instruction and the next MILC(519) instruction are interlocked. MILH(517) and MILC(519) are used as a pair. MILH(517)/MILC(519) interlocks can be nested (e.g., MILH(517)—MILH(517)—MILC(519)—MILC(519)). 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 while it was interlocked.	Output Required	214
MULTI-INTERLOCK DIFFERENTIATION RELEASE MILR 518 CS/CJ-series CPU Unit Ver. 2.0 or later only	MILR (518) N D **N**: Interlock number **D**: Interlock Status Bit	When the execution condition for MILR(518) is OFF, the outputs for all instructions between that MILR(518) instruction and the next MILC(519) instruction are interlocked.MILR(518) and MILC(519) are used as a pair. MILR(518)/MILC(519) interlocks can be nested (e.g., MILR(518)—MILR(518)—MILC(519)—MILC(519)). 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.	Output Required	214
MULTI-INTERLOCK CLEAR MILC 519 CS/CJ-series CPU Unit Ver. 2.0 or later only	MILC (519) N **N**: Interlock number	Clears an interlock started by an MILH(517) or MILR(518) with the same interlock number. All outputs between MILH(517)/MILR(518) and the corresponding MILC(519) with the same interlock number are interlocked when the execution condition for MILH(517)/MILR(518) is OFF.	Output Not required	214
JUMP JMP 004	JMP(004) N **N**: Jump number	When the execution condition for JMP(004) is OFF, program execution jumps directly to the first JME(005) in the program with the same jump number. JMP(004) and JME(005) are used in pairs. 	Output Required	228
CONDITIONAL JUMP CJP 510	CJP(510) N **N**: Jump number	The operation of CJP(510) is the basically the opposite of JMP(004). When the execution condition for CJP(510) is ON, program execution jumps directly to the first JME(005) in the program with the same jump number. CJP(510) and JME(005) are used in pairs. 	Output Required	232

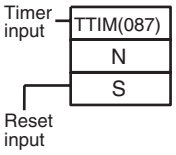
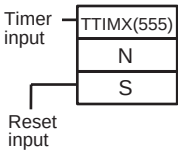
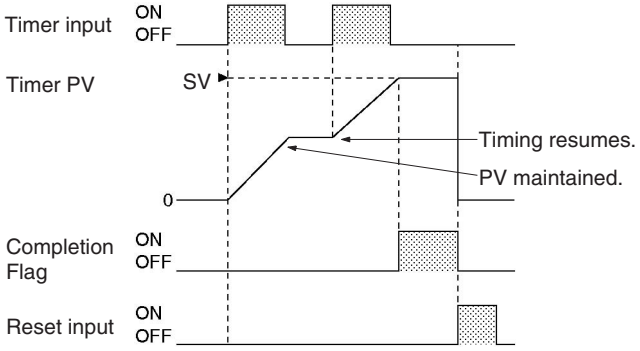
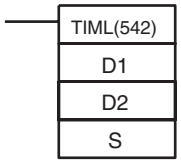
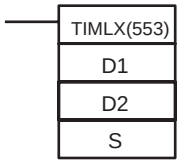
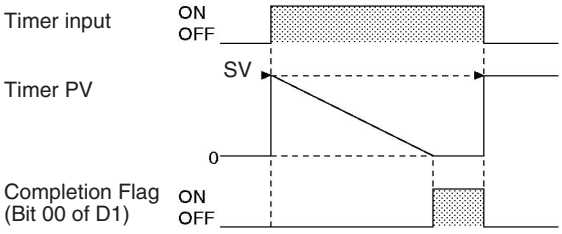
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
JUMP END JME 005	JME(005) N **N**: Jump number	Indicates the end of a jump initiated by JMP(004) or CJP(510).	Output Not required	228
CONDITIONAL JUMP CJPN 511	CJPN(511) N **N**: Jump number	The operation of CJPN(511) is almost identical to JMP(004). When the execution condition for CJP(004) is OFF, program execution jumps directly to the first JME(005) in the program with the same jump number. CJPN(511) and JME(005) are used in pairs. 	Output Not required	232
MULTIPLE JUMP JMP0 515	JMP0(515)	When the execution condition for JMP0(515) is OFF, all instructions from JMP0(515) to the next JME0(516) in the program are processed as NOP(000). Use JMP0(515) and JME0(516) in pairs. There is no limit on the number of pairs that can be used in the program. 	Output Required	236
MULTIPLE JUMP END JME0 516	JME0(516)	When the execution condition for JMP0(515) is OFF, all instructions from JMP0(515) to the next JME0(516) in the program are processed as NOP(000). Use JMP0(515) and JME0(516) in pairs. There is no limit on the number of pairs that can be used in the program.	Output Not required	236

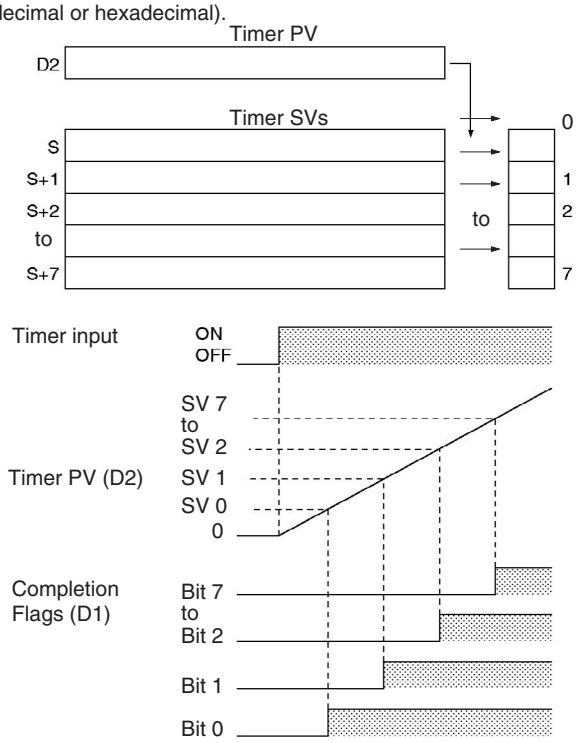
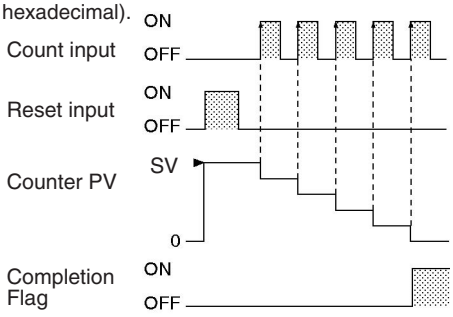
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FOR-NEXT LOOPS FOR 512	FOR(512) N **N**: Number of loops	The instructions between FOR(512) and NEXT(513) are repeated a specified number of times. FOR(512) and NEXT(513) are used in pairs. 	Output Not required	238
BREAK LOOP BREAK 514	BREAK(514)	Programmed in a FOR-NEXT loop to cancel the execution of the loop for a given execution condition. The remaining instructions in the loop are processed as NOP(000) instructions. 	Output Required	241
FOR-NEXT LOOPS NEXT 513	NEXT(513)	The instructions between FOR(512) and NEXT(513) are repeated a specified number of times. FOR(512) and NEXT(513) are used in pairs.	Output Not required	238

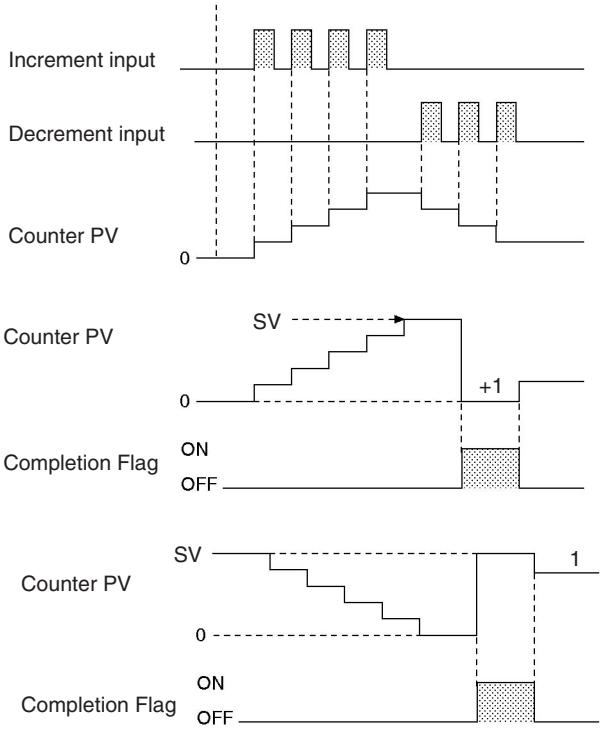
2-2-4 Timer and Counter Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
HUNDRED-MS TIMER TIM (BCD) TIMX (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	TIM N S **N**: Timer number **S**: Set value TIMX(550) N S **N**: Timer number **S**: Set value	TIM/TIMX(550) operates a decrementing timer with units of 0.1-s. The setting range for the set value (SV) is 0 to 999.9 s for BCD and 0 to 6,553.5 s for binary (decimal or hexadecimal). 	Output Required	245
TEN-MS TIMER TIMH 015 (BCD) TIMHX 551 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	TIMH(015) N S **N**: Timer number **S**: Set value TIMHX(551) N S **N**: Timer number **S**: Set value	TIMH(015)/TIMHX(551) operates a decrementing timer with units of 10-ms. The setting range for the set value (SV) is 0 to 99.99 s for BCD and 0 to 655.35 s for binary (decimal or hexadecimal). 	Output Required	249
ONE-MS TIMER TMHH 540 (BCD) TMHHX 552 (BCD) (CS1-H, CJ1-H, CJ1M, or CS1D only)	TMHH(540) N S **N**: Timer number **S**: Set value TMHHX(552) N S **N**: Timer number **S**: Set value	TMHH(540)/TMHHX(552) operates a decrementing timer with units of 1-ms. The setting range for the set value (SV) is 0 to 9.999 s for BCD and 0 to 65.535 s for binary (decimal or hexadecimal). 	Output Required	253

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
TENTH-MS TIMER (CJ1-H-R only) TIMU 541 (BCD) TIMUX 556 (BCD)	TIMU(541) N S **N**: Timer number **S**: Set value TIMUX(556) N S **N**: Timer number **S**: Set value	TIMU(541)/TIMUX(556) operates an decrementing timer with units of 0.1-s. The setting range for the set value (SV) is 0 to 0.999 s for BCD and 0 to 6,553.5 s for binary (decimal or hexadecimal).  Timer Input Turns OFF before Completion Flag Turns ON  **Note:** The timer's present value cannot be accessed for a TENTH-MS TIMER instruction.	Output Required	256
HUNDREDTH-MS TIMER (CJ1-H-R only) TMUH 554 (BCD) TMUHX 557 (BCD)	TMUH(554) N S **N**: Timer number **S**: Set value TMUHX(557) N S **N**: Timer number **S**: Set value	TMUH(554)/TMUHX(557) operates an decrementing timer with units of 0.01-s. The setting range for the set value (SV) is 0 to 0.0999 s for BCD and 0 to 0.65535 s for binary (decimal or hexadecimal).  Timer Input Turns OFF before Completion Flag Turns ON  **Note:** The timer's present value cannot be accessed for a HUNDREDTH-MS TIMER instruction.	Output Required	259

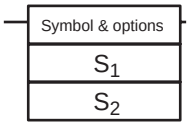
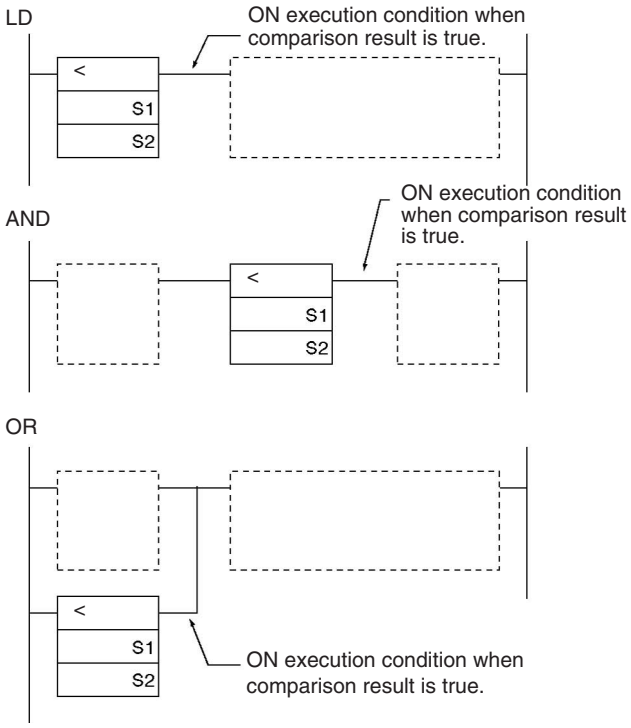
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ACCUMULATIVE TIMER TTIM 087 (BCD) TTIMX 555 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	 **N**: Timer number **S**: Set value  **N**: Timer number **S**: Set value	TTIM(087)/TTIMX(555) operates an incrementing timer with units of 0.1-s. The setting range for the set value (SV) is 0 to 999.9 s for BCD and 0 to 6,553.5 s for binary (decimal or hexadecimal). 	Output Required	262
LONG TIMER TIML 542 (BCD) TIMLX 553 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	 **D1**: Completion Flag **D2**: PV word **S**: SV word  **D1**: Completion Flag **D2**: PV word **S**: SV word	TIML(542)/TIMLX(553) operates a decrementing timer with units of 0.1-s that can time up to approx. 115 days for BCD and 49,710 days for binary (decimal or hexadecimal). 	Output Required	266

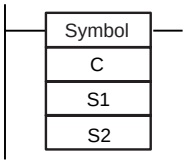
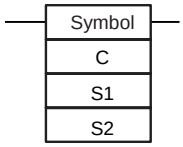
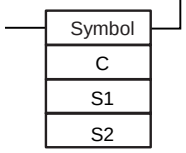
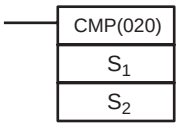
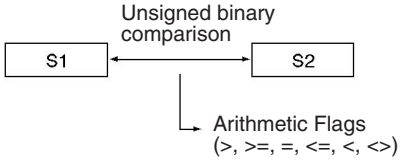
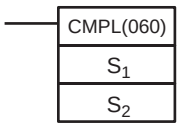
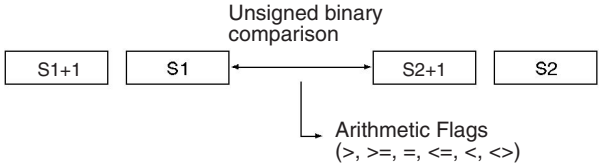
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
MULTI-OUTPUT TIMER MTIM 543 (BCD) MTIMX 554 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	MTIM(543) D1 D2 S **D1**: Completion Flags **D2**: PV word **S**: 1st SV word MTIMX(554) D1 D2 S **D1**: Completion Flags **D2**: PV word **S**: 1st SV word	MTIM(543)/MTIMX(554) operates a 0.1-s incrementing timer with 8 independent SVs and Completion Flags. The setting range for the set value (SV) is 0 to 999.9 s for BCD and 0 to 6,553.5 s for binary (decimal or hexadecimal). 	Output Required	269
COUNTER CNT (BCD) CNTX 546 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	Count input CNT N S Reset input **N**: Counter number **S**: Set value Count input CNTX(546) N S Reset input **N**: Counter number **S**: Set value	CNT/CNTX(546) operates a decrementing counter. The setting range for the set value (SV) is 0 to 9,999 for BCD and 0 to 65,535 for binary (decimal or hexadecimal). 	Output Required	275

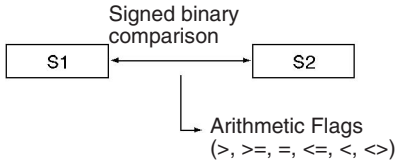
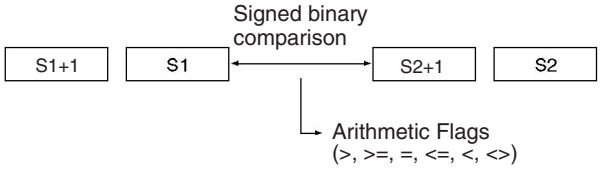
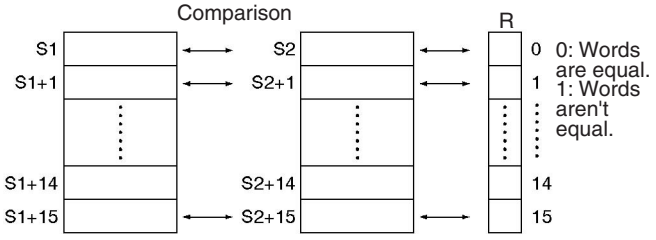
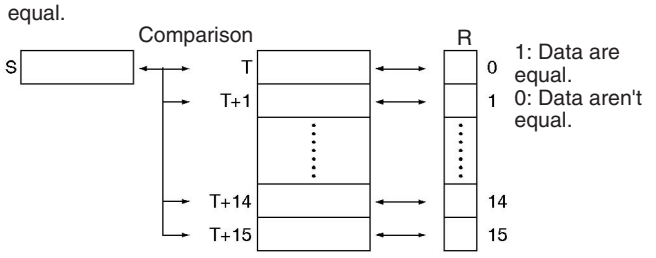
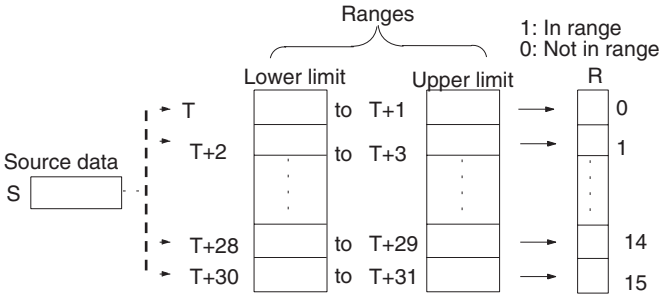
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
REVERSIBLE COUNTER CNTR 012 (BCD) CNTRX 548 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	Increment input Decrement input Reset input CNTR(012) N S **N**: Counter number **S**: Set value	CNTR(012)/CNTRX(548) operates a reversible counter. 	Output Required	278
	Increment input Decrement input Reset input CNTRX(548) N S **N**: Counter number **S**: Set value			
RESET TIMER/ COUNTER CNR @CNR 545 (BCD) CNRX @CNRX 547 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	CNR(545) N1 N2 N_1: 1st number in range N_2: Last number in range	CNR(545)/CNRX(547) resets the timers or counters within the specified range of timer or counter numbers. Sets the set value (SV) to the maximum of 9999.	Output Required	282
	CNRX(547) N1 N2 N_1: 1st number in range N_2: Last number in range			

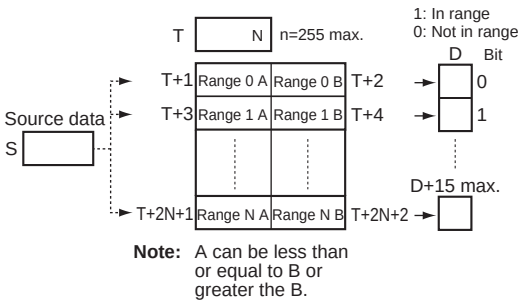
2-2-5 Comparison Instructions

*1: Not supported by CS1D CPU Units for Duplex-CPU Systems.

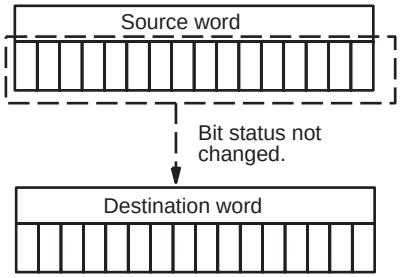
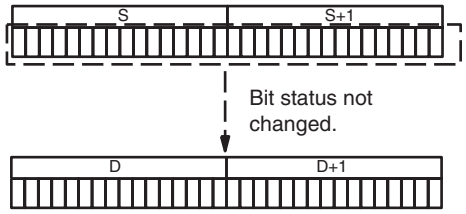
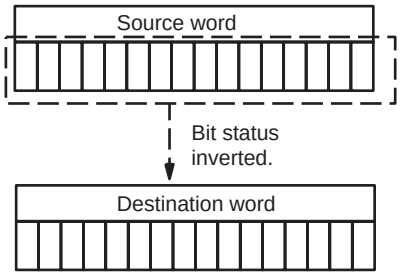
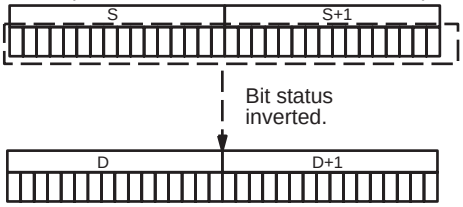
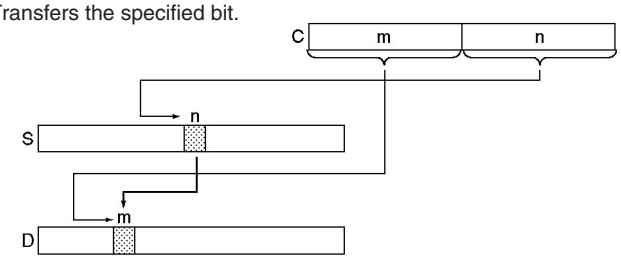
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
Symbol Comparison (Unsigned) LD, AND, OR + =, <>, <, <=, >, >= 300 (=) 305 (<>) 310 (<) 315 (<=) 320 (>) 325(>=)	 **S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (unsigned) compare two values (constants and/or the contents of specified words) in 16-bit binary data and create an ON execution condition when the comparison condition is true. There are three types of symbol comparison instructions, LD (LOAD), AND, and OR. 	LD: Not required AND, OR: Required	291
Symbol Comparison (Double-word, unsigned) LD, AND, OR + =, <>, <, <=, >, >= + L 301 (=) 306 (<>) 311 (<) 316 (<=) 321 (>) 326 (>=)	**S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (double-word, unsigned) compare two values (constants and/or the contents of specified double-word data) in unsigned 32-bit binary data and create an ON execution condition when the comparison condition is true. There are three types of symbol comparison instructions, LD (LOAD), AND, and OR.	LD: Not required AND, OR: Required	291
Symbol Comparison (Signed) LD, AND, OR + =, <>, <, <=, >, >= +S 302 (=) 307 (<>) 312 (<) 317 (<=) 322 (>) 327 (>=)	**S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (signed) compare two values (constants and/or the contents of specified words) in signed 16-bit binary (4-digit hexadecimal) and create an ON execution condition when the comparison condition is true. There are three types of symbol comparison instructions, LD (LOAD), AND, and OR.	LD: Not required AND, OR: Required	291

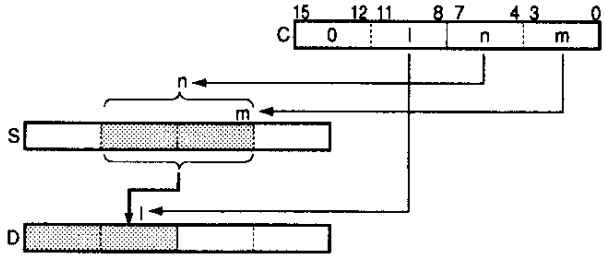
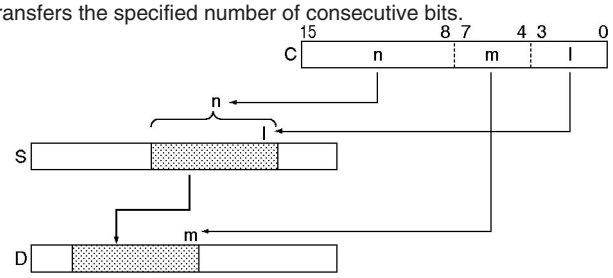
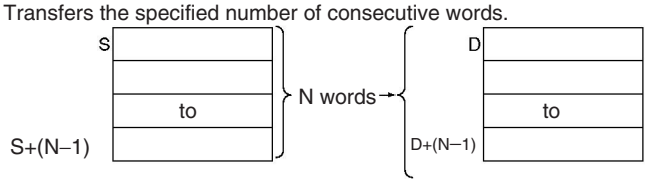
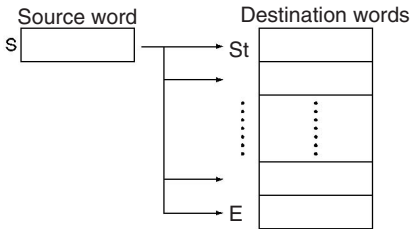
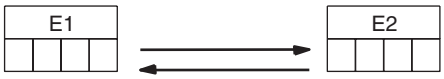
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
Symbol Comparison (Double-word, signed) LD, AND, OR + =, <>, <, <=, >, >= +SL 303 (=) 308 (<>) 313 (<) 318 (<=) 323 (>) 328 (>=)	**S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (double-word, signed) compare two values (constants and/or the contents of specified double-word data) in signed 32-bit binary (8-digit hexadecimal) and create an ON execution condition when the comparison condition is true. There are three types of symbol comparison instructions, LD (LOAD), AND, and OR.	LD: Not required AND, OR: Required	291
Time Comparison LD, AND, OR + = DT, <> DT, < DT, <= DT, > DT, >= DT 341 (= DT) 342 (<> DT) 343 (< DT) 344 (<= DT) 345 (> DT) 346 (>= DT) (CS/CJ-series CPU Unit Ver. 2.0 or later only)	LD (LOAD):  AND:  OR:  **C:** Control word **S1:** 1st word of present time **S2:** 1st word of comparison time	Time comparison instructions compare two BCD time values and create an ON execution condition when the comparison condition is true. There are three types of time comparison instructions, LD (LOAD), AND, and OR. Time values (year, month, day, hour, minute, and second) can be masked/unmasked in the comparison so it is easy to create calendar timer functions.	LD: Not required AND, OR: Required	297
UNSIGNED COMPARE CMP !CMP*1 020	 **S_1**: Comparison data 1 **S_2**: Comparison data 2	Compares two unsigned binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area. 	Output Required	303
DOUBLE UNSIGNED COMPARE CMPL 060	 **S_1**: Comparison data 1 **S_2**: Comparison data 2	Compares two double unsigned binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area. 	Output Required	306

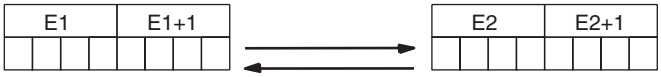
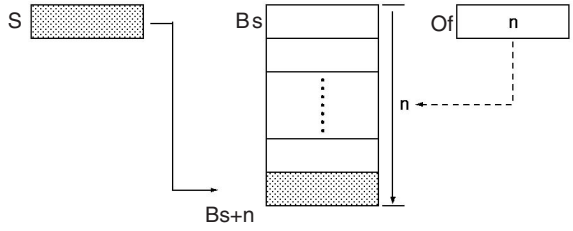
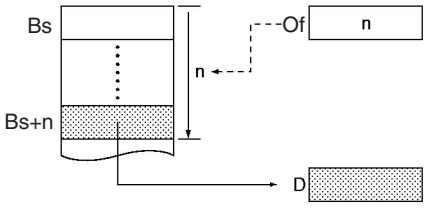
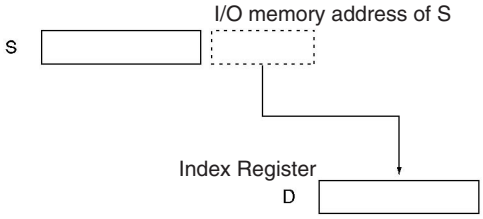
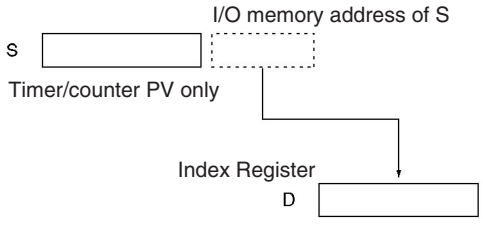
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SIGNED BINARY COMPARE CPS !CPS*1 114	CPS(114) S_1 S_2 **S1**: Comparison data 1 **S2**: Comparison data 2	Compares two signed binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area. 	Output Required	309
DOUBLE SIGNED BINARY COMPARE CPSL 115	CPSL(115) S_1 S_2 **S1**: Comparison data 1 **S2**: Comparison data 2	Compares two double signed binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area. 	Output Required	312
MULTIPLE COMPARE MCMP @MCMP 019	MCMP(019) S_1 S_2 R **S1**: 1st word of set 1 **S2**: 1st word of set 2 **R**: Result word	Compares 16 consecutive words with another 16 consecutive words and turns ON the corresponding bit in the result word where the contents of the words **are not** equal. 	Output Required	315
TABLE COMPARE TCMP @TCMP 085	TCMP(085) S T R **S**: Source data **T**: 1st word of table **R**: Result word	Compares the source data to the contents of 16 words and turns ON the corresponding bit in the result word when the contents are equal. 	Output Required	317
UNSIGNED BLOCK COMPARE BCMP @BCMP 068	BCMP(068) S T R **S**: Source data **T**: 1st word of table **R**: Result word	Compares the source data to 16 ranges (defined by 16 lower limits and 16 upper limits) and turns ON the corresponding bit in the result word when the source data is within the range. 	Output Required	320

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
EXPANDED BLOCK COMPARE BCMP2 @BCMP2 502 (CS1-H, CJ1-H, or CS1D CPU Unit Ver. 2.0 or later only) CJ1M CPU Unit (Pre-Ver. 2.0 or Unit Ver. 2.0 or later)	BCMP2(502) S T R **S**: Source data **T**: 1st word of block **R**: Result word	Compares the source data to up to 256 ranges (defined by upper and lower limits) and turns ON the corresponding bit in the result word when the source data is within a range.  **Note:** A can be less than or equal to B or greater the B.	Output Required	322
AREA RANGE COMPARE ZCP @ZCP 088 (CS1-H, CJ1-H, CJ1M, or CS1D only)	ZCP(088) CD LL UL **CD**: Compare data (1 word) **LL**: Lower limit of range **UL**: Upper limit of range	Compares the 16-bit unsigned binary value in CD (word contents or constant) to the range defined by LL and UL and outputs the results to the Arithmetic Flags in the Auxiliary Area.	Output Required	326
DOUBLE AREA RANGE COMPARE ZCPL @ZCPL 116 (CS1-H, CJ1-H, CJ1M, or CS1D only)	ZCPL(116) CD LL UL **CD**: Compare data (2 words) **LL**: Lower limit of range **UL**: Upper limit of range	Compares the 32-bit unsigned binary value in CD and CD+1 (word contents or constant) to the range defined by LL and UL and outputs the results to the Arithmetic Flags in the Auxiliary Area.	Output Required	329

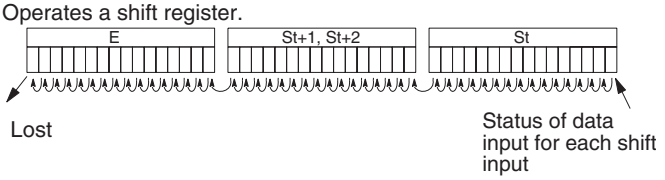
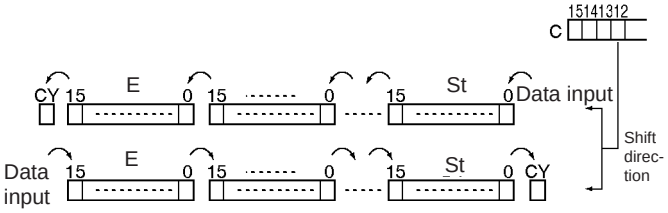
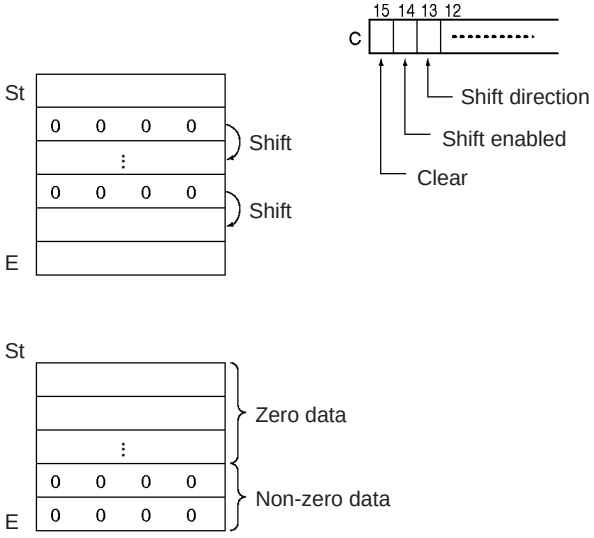
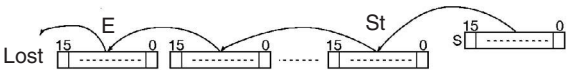
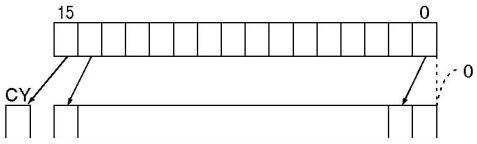
2-2-6 Data Movement Instructions

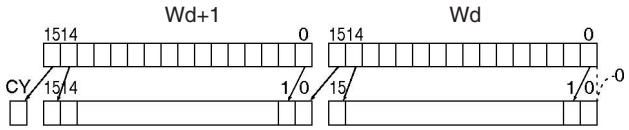
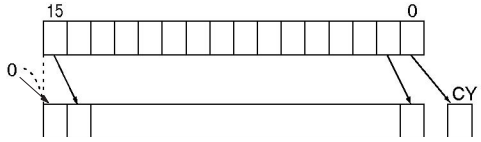
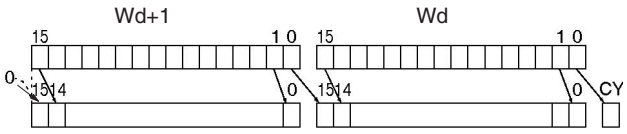
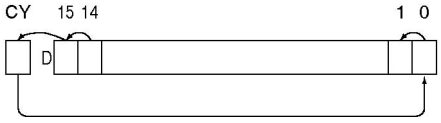
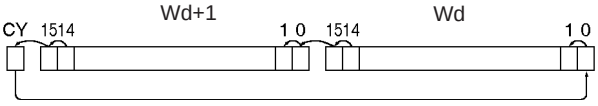
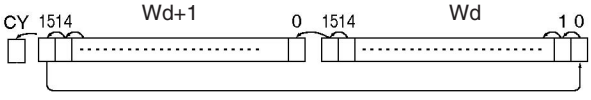
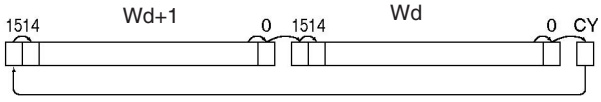
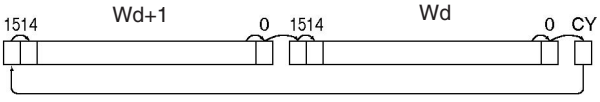
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
MOVE MOV @MOV !MOV !@MOV 021	MOV(021) S D **S**: Source **D**: Destination	Transfers a word of data to the specified word. 	Output Required	331
DOUBLE MOVE MOVL @MOVL 498	MOVL(498) S D **S**: 1st source word **D**: 1st destination word	Transfers two words of data to the specified words. 	Output Required	334
MOVE NOT MVN @MVN 022	MVN(022) S D **S**: Source **D**: Destination	Transfers the complement of a word of data to the specified word. 	Output Required	333
DOUBLE MOVE NOT MVNL @MVNL 499	MVNL(499) S D **S**: 1st source word **D**: 1st destination word	Transfers the complement of two words of data to the specified words. 	Output Required	336
MOVE BIT MOVB @MOVB 082	MOVB(082) S C D **S**: Source word or data **C**: Control word **D**: Destination word	Transfers the specified bit. 	Output Required	337

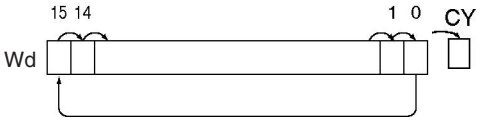
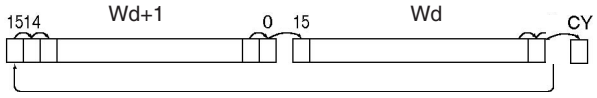
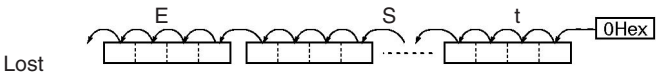
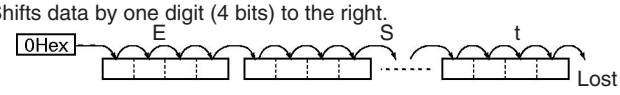
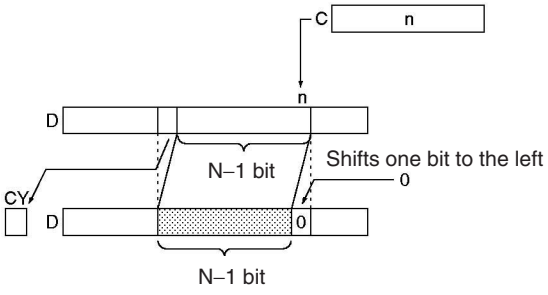
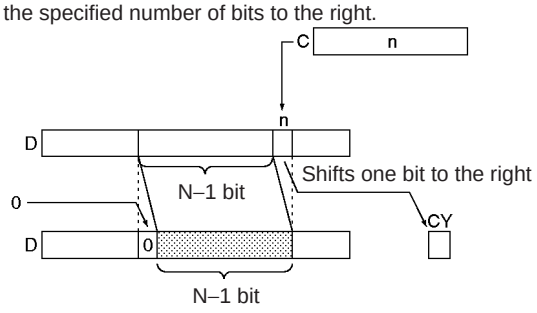
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
MOVE DIGIT MOVD @MOVD 083	MOVD(083) S C D **S**: Source word or data **C**: Control word **D**: Destination word	Transfers the specified digit or digits. (Each digit is made up of 4 bits.) 	Output Required	339
MULTIPLE BIT TRANSFER XFRB @XFRB 062	XFRB(062) C S D **C**: Control word **S**: 1st source word **D**: 1st destination word	Transfers the specified number of consecutive bits. 	Output Required	342
BLOCK TRANSFER XFER @XFER 070	XFER(070) N S D **N**: Number of words **S**: 1st source word **D**: 1st destination word	Transfers the specified number of consecutive words. 	Output Required	344
BLOCK SET BSET @BSET 071	BSET(071) S St E **S**: Source word **St**: Starting word **E**: End word	Copies the same word to a range of consecutive words. 	Output Required	347
DATA EXCHANGE XCHG @XCHG 073	XCHG(073) E1 E2 **E1**: 1st exchange word **E2**: Second exchange word	Exchanges the contents of the two specified words. 	Output Required	349

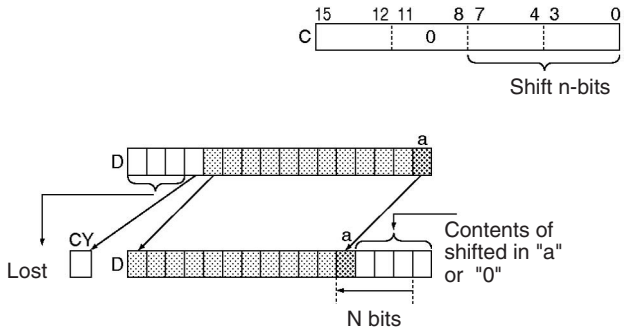
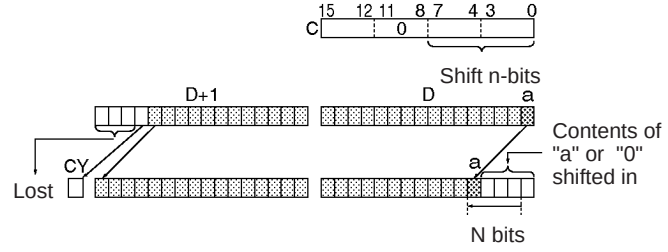
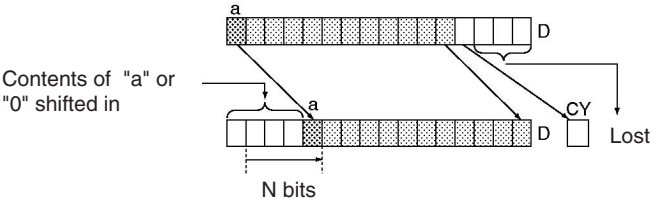
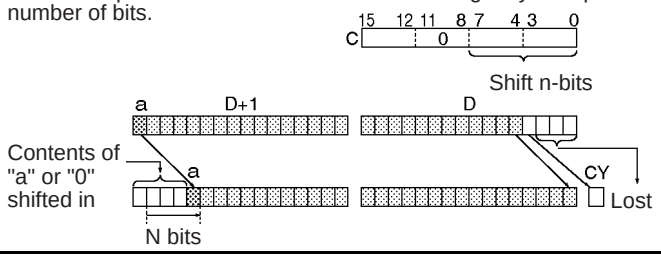
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE DATA EXCHANGE XCGL @XCGL 562	XCGL(562) E1 E2 **E1**: 1st exchange word **E2**: Second exchange word	Exchanges the contents of a pair of consecutive words with another pair of consecutive words. 	Output Required	350
SINGLE WORD DISTRIBUTE DIST @DIST 080	DIST(080) S Bs Of **S**: Source word **Bs**: Destination base address **Of**: Offset	Transfers the source word to a destination word calculated by adding an offset value to the base address. 	Output Required	352
DATA COLLECT COLL @COLL 081	COLL(081) Bs Of D **Bs**: Source base address **Of**: Offset **D**: Destination word	Transfers the source word (calculated by adding an offset value to the base address) to the destination word. 	Output Required	354
MOVE TO REGISTER MOVR @MOVR 560	MOVR(560) S D **S**: Source (desired word or bit) **D**: Destination (Index Register)	Sets the internal I/O memory address of the specified word, bit, or timer/counter Completion Flag in the specified Index Register. (Use MOVRW(561) to set the internal I/O memory address of a timer/counter PV in an Index Register.) 	Output Required	356
MOVE TIMER/ COUNTER PV TO REGISTER MOVRW @MOVRW 561	MOVRW(561) S D **S**: Source (desired TC number) **D**: Destination (Index Register)	Sets the internal I/O memory address of the specified timer or counter's PV in the specified Index Register. (Use MOVR(560) to set the internal I/O memory address of a word, bit, or timer/counter Completion Flag in an Index Register.) 	Output Required	358

2-2-7 Data Shift Instructions

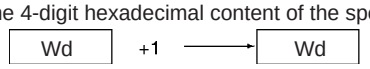
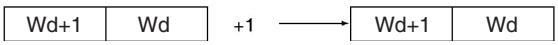
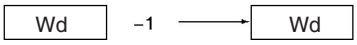
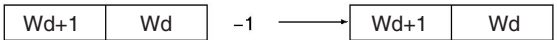
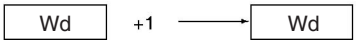
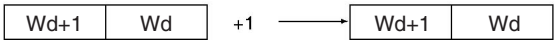
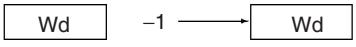
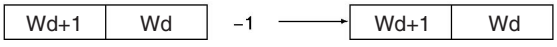
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SHIFT REGISTER SFT 010	Data input Shift input Reset input SFT(010) St E **St**: Starting word **E**: End word	Operates a shift register. 	Output Required	361
REVERSIBLE SHIFT REGISTER SFTR @SFTR 084	SFTR(084) C St E **C**: Control word **St**: Starting word **E**: End word	Creates a shift register that shifts data to either the right or the left. 	Output Required	362
ASYNCHRONOUS SHIFT REGISTER ASFT @ASFT 017	ASFT(017) C St E **C**: Control word **St**: Starting word **E**: End word	Shifts all non-zero word data within the specified word range either towards St or toward E, replacing 0000Hex word data. 	Output Required	365
WORD SHIFT WSFT @WSFT 016	WSFT(016) S St E **S**: Source word **St**: Starting word **E**: End word	Shifts data between St and E in word units. 	Output Required	368
ARITHMETIC SHIFT LEFT ASL @ASL 025	ASL(025) Wd **Wd**: Word	Shifts the contents of Wd one bit to the left. 	Output Required	370

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE SHIFT LEFT ASLL @ASLL 570	ASLL(570) Wd **Wd**: Word	Shifts the contents of Wd and Wd +1 one bit to the left. 	Output Required	371
ARITHMETIC SHIFT RIGHT ASR @ASR 026	ASR(026) Wd **Wd**: Word	Shifts the contents of Wd one bit to the right. 	Output Required	373
DOUBLE SHIFT RIGHT ASRL @ASRL 571	ASRL(571) Wd **Wd**: Word	Shifts the contents of Wd and Wd +1 one bit to the right. 	Output Required	374
ROTATE LEFT ROL @ROL 027	ROL(027) Wd **Wd**: Word	Shifts all Wd bits one bit to the left including the Carry Flag (CY). 	Output Required	376
DOUBLE ROTATE LEFT ROLL @ROLL 572	ROLL(572) Wd **Wd**: Word	Shifts all Wd and Wd + 1 bits one bit to the left including the Carry Flag (CY). 	Output Required	378
ROTATE LEFT WITHOUT CARRY RLNC @RLNC 574	RLNC(574) Wd **Wd**: Word	Shifts all Wd bits one bit to the left not including the Carry Flag (CY). 	Output Required	383
DOUBLE ROTATE LEFT WITHOUT CARRY RLNL @RLNL 576	RLNL(576) Wd **Wd**: Word	Shifts all Wd and Wd +1 bits one bit to the left not including the Carry Flag (CY). 	Output Required	385
ROTATE RIGHT ROR @ROR 028	ROR(028) Wd **Wd**: Word	Shifts all Wd bits one bit to the right including the Carry Flag (CY). 	Output Required	380
DOUBLE ROTATE RIGHT RORL @RORL 573	RORL(573) Wd **Wd**: Word	Shifts all Wd and Wd +1 bits one bit to the right including the Carry Flag (CY). 	Output Required	381

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ROTATE RIGHT WITHOUT CARRY RRNC @RRNC 575	RRNC(575) Wd **Wd**: Word	Shifts all Wd bits one bit to the right not including the Carry Flag (CY). The contents of the rightmost bit of Wd shifts to the leftmost bit and to the Carry Flag (CY). 	Output Required	387
DOUBLE ROTATE RIGHT WITHOUT CARRY RRNL @RRNL 577	RRNL(577) Wd **Wd**: Word	Shifts all Wd and Wd +1 bits one bit to the right not including the Carry Flag (CY). The contents of the rightmost bit of Wd +1 is shifted to the leftmost bit of Wd, and to the Carry Flag (CY). 	Output Required	388
ONE DIGIT SHIFT LEFT SLD @SLD 074	SLD(074) St E **St**: Starting word **E**: End word	Shifts data by one digit (4 bits) to the left. 	Output Required	390
ONE DIGIT SHIFT RIGHT SRD @SRD 075	SRD(075) St E **St**: Starting word **E**: End word	Shifts data by one digit (4 bits) to the right. 	Output Required	392
SHIFT N-BIT DATA LEFT NSFL @NSFL 578	NSFL(578) D C N **D**: Beginning word for shift **C**: Beginning bit **N**: Shift data length	Shifts the specified number of bits to the left. 	Output Required	393
SHIFT N-BIT DATA RIGHT NSFR @NSFR 579	NSFR(579) D C N **D**: Beginning word for shift **C**: Beginning bit **N**: Shift data length	Shifts the specified number of bits to the right. 	Output Required	395

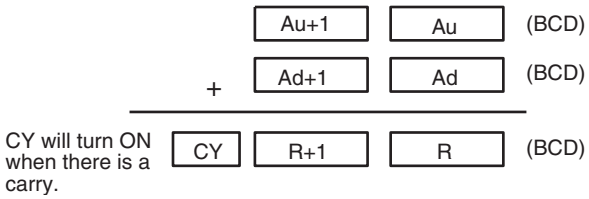
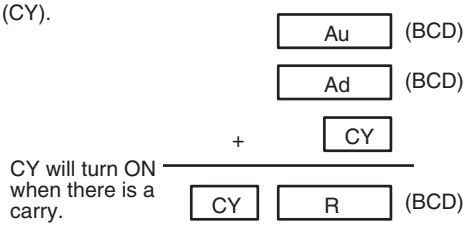
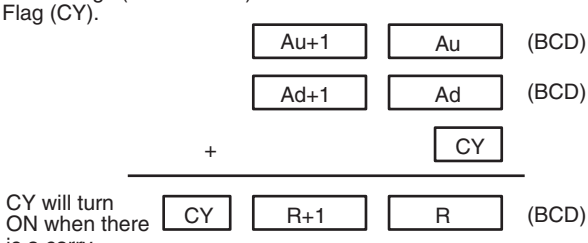
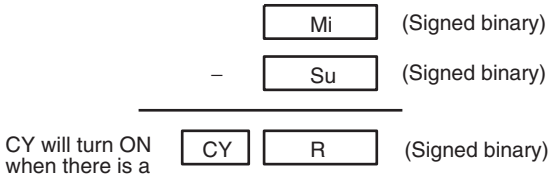
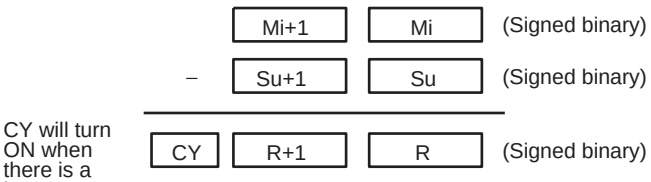
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SHIFT N-BITS LEFT NASL @NASL 580	NASL(580) D C **D**: Shift word **C**: Control word	Shifts the specified 16 bits of word data to the left by the specified number of bits. 	Output Required	397
DOUBLE SHIFT N-BITS LEFT NSLL @NSLL 582	NSLL(582) D C **D**: Shift word **C**: Control word	Shifts the specified 32 bits of word data to the left by the specified number of bits. 	Output Required	400
SHIFT N-BITS RIGHT NASR @NASR 581	NASR(581) D C **D**: Shift word **C**: Control word	Shifts the specified 16 bits of word data to the right by the specified number of bits. 	Output Required	403
DOUBLE SHIFT N-BITS RIGHT NSRL @NSRL 583	NSRL(583) D C **D**: Shift word **C**: Control word	Shifts the specified 32 bits of word data to the right by the specified number of bits. 	Output Required	405

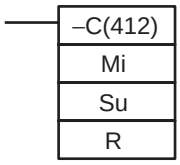
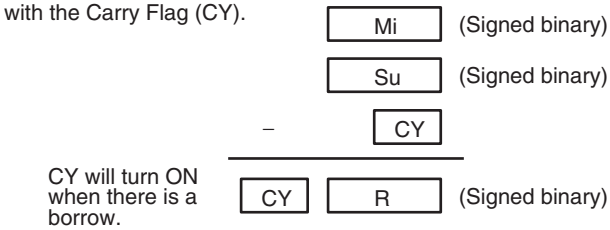
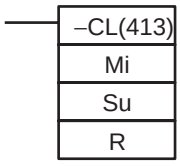
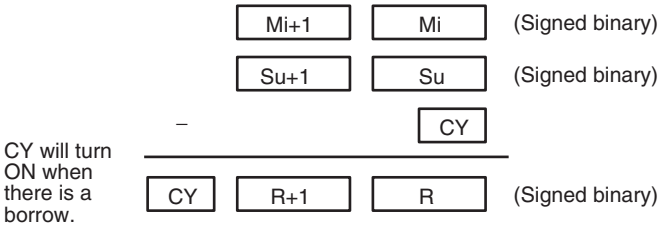
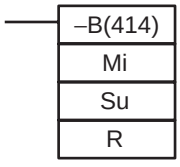
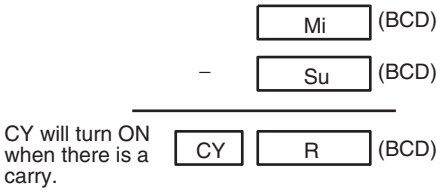
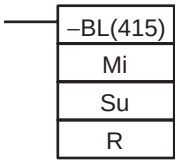
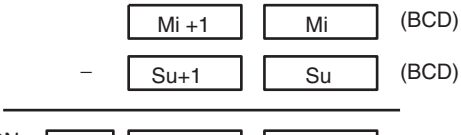
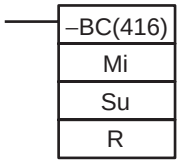
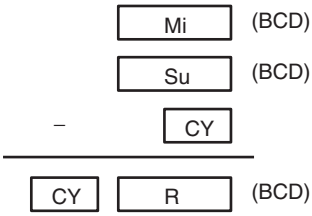
2-2-8 Increment/Decrement Instructions

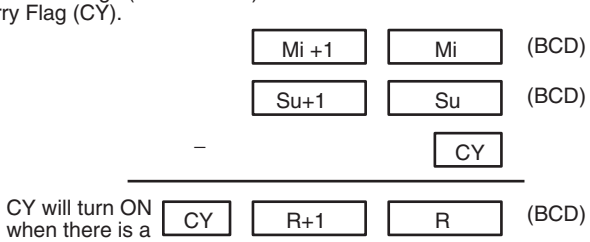
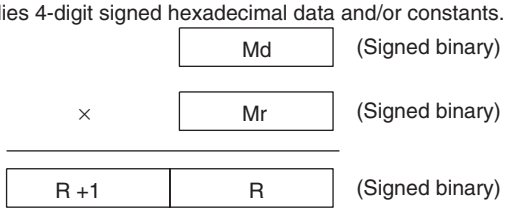
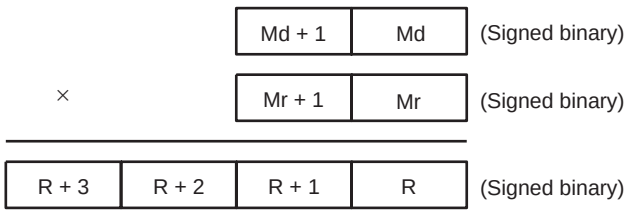
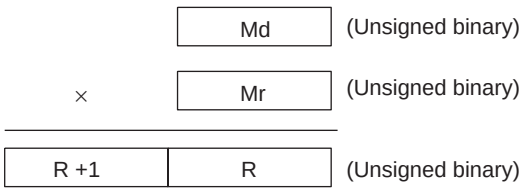
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
INCREMENT BINARY ++ @++ 590	++(590) Wd **Wd**: Word	Increments the 4-digit hexadecimal content of the specified word by 1. 	Output Required	409
DOUBLE INCREMENT BINARY ++L @++L 591	++L(591) Wd **Wd**: Word	Increments the 8-digit hexadecimal content of the specified words by 1. 	Output Required	411
DECREMENT BINARY – – @– – 592	– –(592) Wd **Wd**: Word	Decrements the 4-digit hexadecimal content of the specified word by 1. 	Output Required	413
DOUBLE DECREMENT BINARY – –L @– –L 593	– – L(593) Wd **Wd**: 1st word	Decrements the 8-digit hexadecimal content of the specified words by 1. 	Output Required	415
INCREMENT BCD ++B @++B 594	++B(594) Wd **Wd**: Word	Increments the 4-digit BCD content of the specified word by 1. 	Output Required	417
DOUBLE INCREMENT BCD ++BL @++BL 595	++BL(595) Wd **Wd**: 1st word	Increments the 8-digit BCD content of the specified words by 1. 	Output Required	419
DECREMENT BCD – –B @– –B 596	– –B(596) Wd **Wd**: Word	Decrements the 4-digit BCD content of the specified word by 1. 	Output Required	421
DOUBLE DECREMENT BCD – –BL @– –BL 597	– – BL(597) Wd **Wd**: 1st word	Decrements the 8-digit BCD content of the specified words by 1. 	Output Required	423

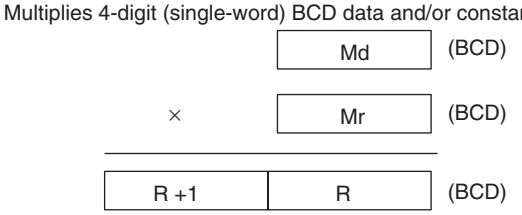
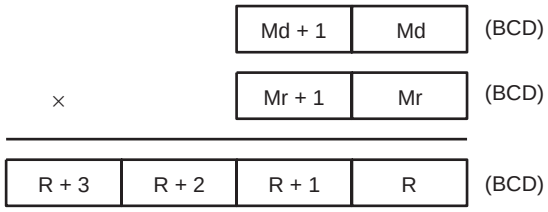
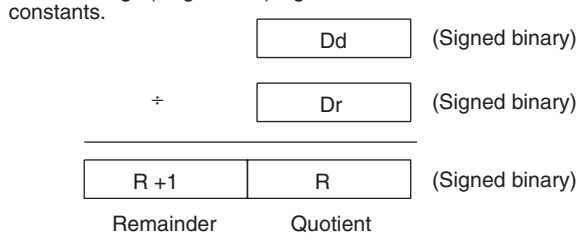
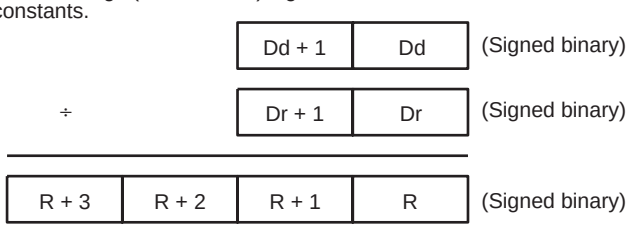
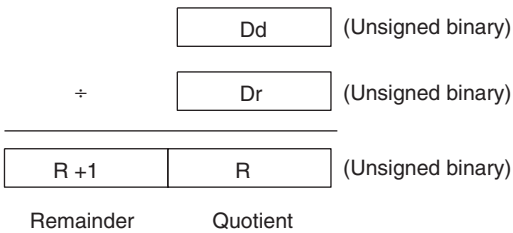
2-2-9 Symbol Math Instructions

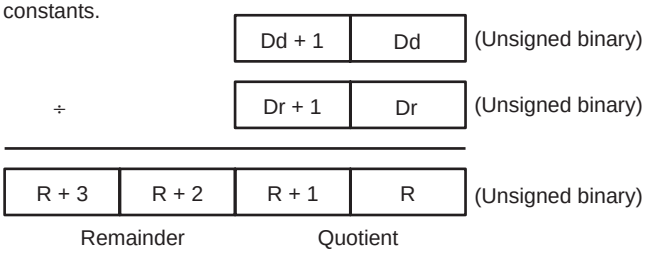
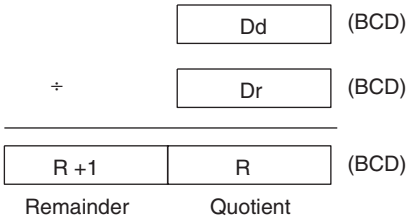
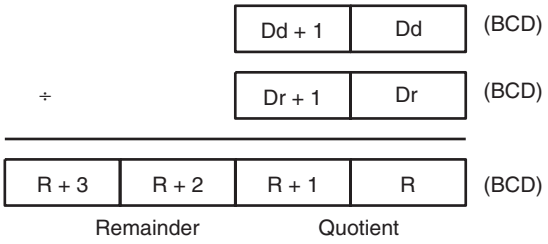
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SIGNED BINARY ADD WITHOUT CARRY + @+ 400	+(400) Au Ad R **Au**: Augend word **Ad**: Addend word **R**: Result word	Adds 4-digit (single-word) hexadecimal data and/or constants. Au (Signed binary) + Ad (Signed binary) CY will turn ON when there is a carry. CY R (Signed binary)	Output Required	426
DOUBLE SIGNED BINARY ADD WITHOUT CARRY +L @+L 401	+L(401) Au Ad R **Au**: 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds 8-digit (double-word) hexadecimal data and/or constants. Au+1 Au (Signed binary) + Ad+1 Ad (Signed binary) CY will turn ON when there is a carry. CY R+1 R (Signed binary)	Output Required	428
SIGNED BINARY ADD WITH CARRY +C @+C 402	+C(402) Au Ad R **Au**: Augend word **Ad**: Addend word **R**: Result word	Adds 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY). Au (Signed binary) Ad (Signed binary) + CY CY will turn ON when there is a carry. CY R (Signed binary)	Output Required	430
DOUBLE SIGNED BINARY ADD WITH CARRY +CL @+CL 403	+CL(403) Au Ad R **Au**: 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY). Au+1 Au (Signed binary) Ad+1 Ad (Signed binary) + CY CY will turn ON when there is a carry. CY R+1 R (Signed binary)	Output Required	432
BCD ADD WITHOUT CARRY +B @+B 404	+B(404) Au Ad R **Au**: Augend word **Ad**: Addend word **R**: Result word	Adds 4-digit (single-word) BCD data and/or constants. Au (BCD) + Ad (BCD) CY will turn ON when there is a carry. CY R (BCD)	Output Required	434

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE BCD ADD WITHOUT CARRY +BL @+BL 405	+BL(405) Au Ad R **Au**: 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds 8-digit (double-word) BCD data and/or constants. 	Output Required	435
BCD ADD WITH CARRY +BC @+BC 406	+BC(406) Au Ad R **Au**: Augend word **Ad**: Addend word **R**: Result word	Adds 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY). 	Output Required	437
DOUBLE BCD ADD WITH CARRY +BCL @+BCL 407	+BCL(407) Au Ad R **Au**: 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY). 	Output Required	439
SIGNED BINARY SUBTRACT WITHOUT CARRY – @– 410	–(410) Mi Su R **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 4-digit (single-word) hexadecimal data and/or constants. 	Output Required	440
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY –L @–L 411	– L(411) Mi Su R **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 8-digit (double-word) hexadecimal data and/or constants. 	Output Required	442

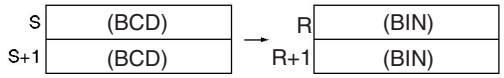
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SIGNED BINARY SUBTRACT WITH CARRY −C @−C 412	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).  CY will turn ON when there is a borrow.	Output Required	446
DOUBLE SIGNED BINARY WITH CARRY −CL @−CL 413	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).  CY will turn ON when there is a borrow.	Output Required	448
BCD SUBTRACT WITHOUT CARRY −B @−B 414	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 4-digit (single-word) BCD data and/or constants.  CY will turn ON when there is a carry.	Output Required	451
DOUBLE BCD SUBTRACT WITHOUT CARRY −BL @−BL 415	 **Mi**: 1st minuend word **Su**: 1st subtrahend word **R**: 1st result word	Subtracts 8-digit (double-word) BCD data and/or constants.  CY will turn ON when there is a borrow.	Output Required	452
BCD SUBTRACT WITH CARRY −BC @−BC 416	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).  CY will turn ON when there is a borrow.	Output Required	456

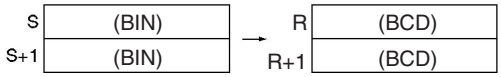
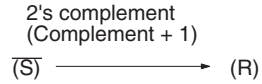
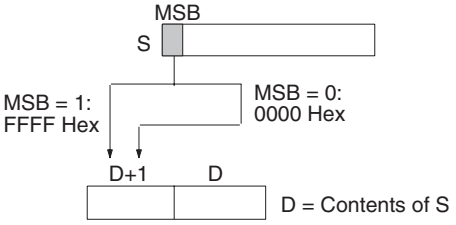
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE BCD SUBTRACT WITH CARRY –BCL @–BCL 417	–BCL(417) Mi Su R **Mi**: 1st minuend word **Su**: 1st subtrahend word **R**: 1st result word	Subtracts 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY). 	Output Required	457
SIGNED BINARY MULTIPLY * @* 420	*(420) Md Mr R **Md**: Multiplicand word **Mr**: Multiplier word **R**: Result word	Multiplies 4-digit signed hexadecimal data and/or constants. 	Output Required	459
DOUBLE SIGNED BINARY MULTIPLY *L @*L 421	*L(421) Md Mr R **Md**: 1st multiplicand word **Mr**: 1st multiplier word **R**: 1st result word	Multiplies 8-digit signed hexadecimal data and/or constants. 	Output Required	461
UNSIGNED BINARY MULTIPLY *U @*U 422	*U(422) Md Mr R **Md**: Multiplicand word **Mr**: Multiplier word **R**: Result word	Multiplies 4-digit unsigned hexadecimal data and/or constants. 	Output Required	463
DOUBLE UNSIGNED BINARY MULTIPLY *UL @*UL 423	*UL(423) Md Mr R **Md**: 1st multiplicand word **Mr**: 1st multiplier word **R**: 1st result word	Multiplies 8-digit unsigned hexadecimal data and/or constants. 	Output Required	465

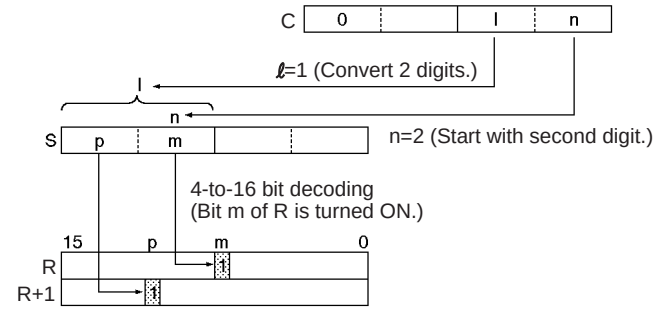
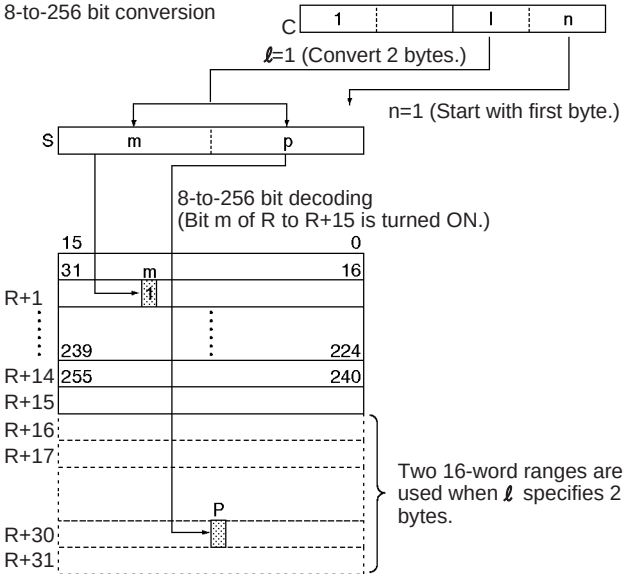
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BCD MULTIPLY *B @*B 424	*B(424) Md Mr R **Md**: Multiplicand word **Mr**: Multiplier word **R**: Result word	Multiplies 4-digit (single-word) BCD data and/or constants. 	Output Required	467
DOUBLE BCD MULTIPLY *BL @*BL 425	*BL(425) Md Mr R **Md**: 1st multiplicand word **Mr**: 1st multiplier word **R**: 1st result word	Multiplies 8-digit (double-word) BCD data and/or constants. 	Output Required	469
SIGNED BINARY DIVIDE / @/ 430	/(430) Dd Dr R **Dd**: Dividend word **Dr**: Divisor word **R**: Result word	Divides 4-digit (single-word) signed hexadecimal data and/or constants. 	Output Required	471
DOUBLE SIGNED BINARY DIVIDE /L @/L 431	/L(431) Dd Dr R **Dd**: 1st dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides 8-digit (double-word) signed hexadecimal data and/or constants. 	Output Required	473
UNSIGNED BINARY DIVIDE /U @/U 432	/U(432) Dd Dr R **Dd**: Dividend word **Dr**: Divisor word **R**: Result word	Divides 4-digit (single-word) unsigned hexadecimal data and/or constants. 	Output Required	475

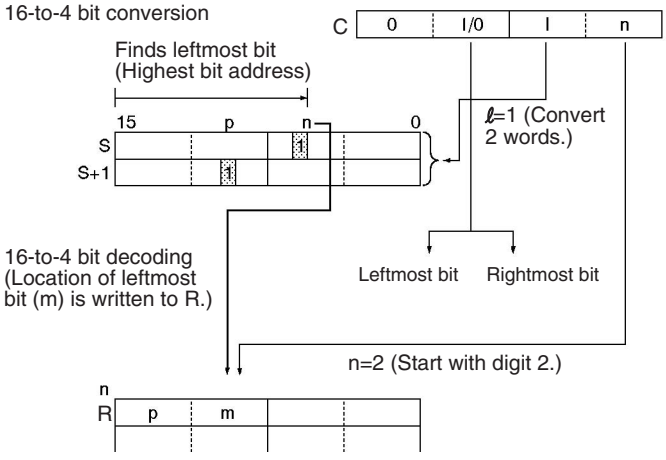
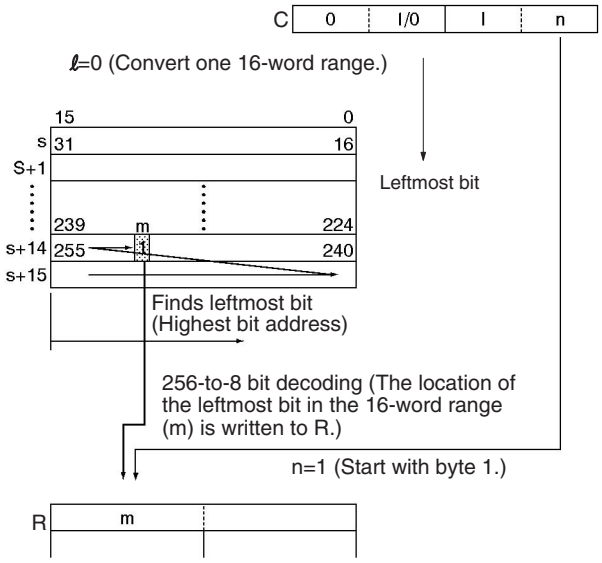
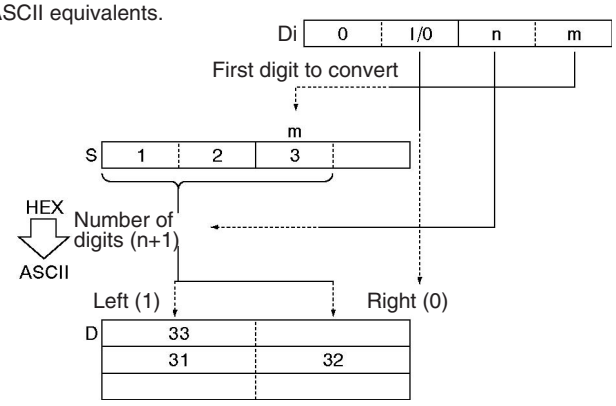
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE UNSIGNED BINARY DIVIDE /UL @/UL 433	/UL(433) Dd Dr R **Dd**: 1st dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides 8-digit (double-word) unsigned hexadecimal data and/or constants. 	Output Required	477
BCD DIVIDE /B @/B 434	/B(434) Dd Dr R **Dd**: Dividend word **Dr**: Divisor word **R**: Result word	Divides 4-digit (single-word) BCD data and/or constants. 	Output Required	479
DOUBLE BCD DIVIDE /BL @/BL 435	/BL(435) Dd Dr R **Dd**: 1st dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides 8-digit (double-word) BCD data and/or constants. 	Output Required	481

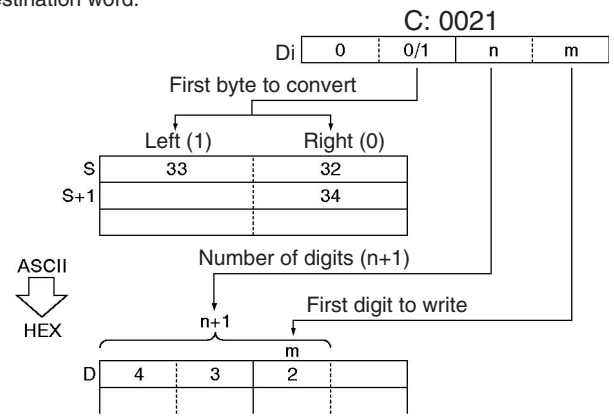
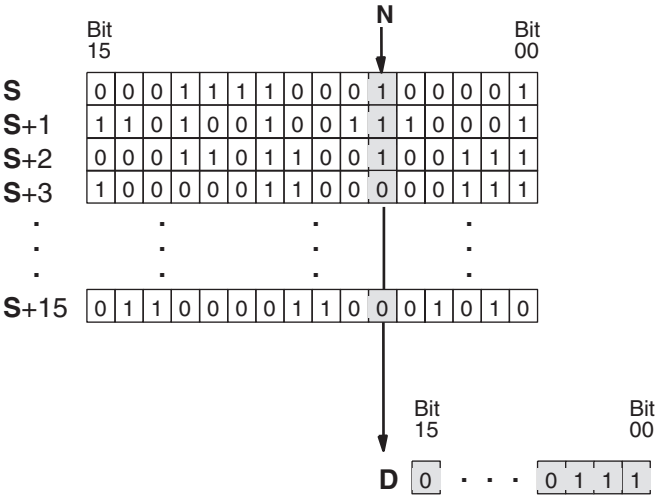
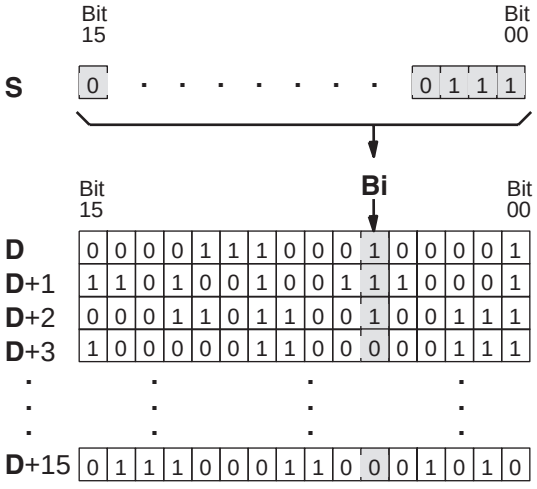
2-2-10 Conversion Instructions

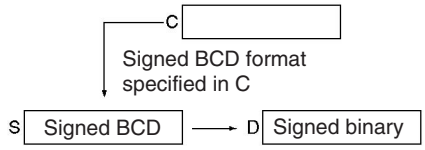
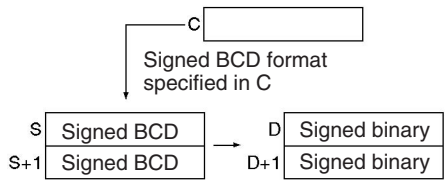
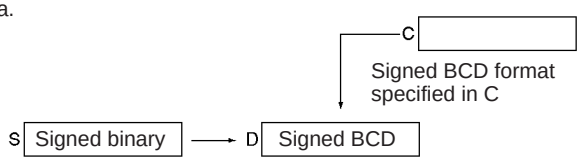
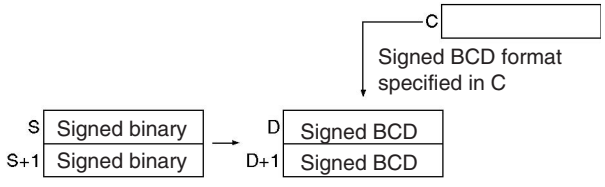
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BCD TO BINARY BIN @BIN 023	BIN(023) S R **S**: Source word **R**: Result word	Converts BCD data to binary data. 	Output Required	483
DOUBLE BCD TO DOUBLE BINARY BINL @BINL 058	BINL(058) S R **S**: 1st source word **R**: 1st result word	Converts 8-digit BCD data to 8-digit hexadecimal (32-bit binary) data. 	Output Required	485

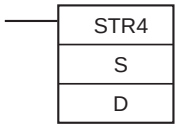
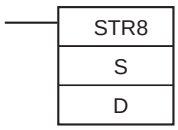
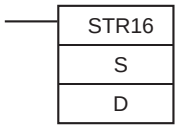
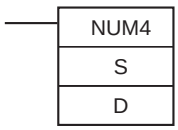
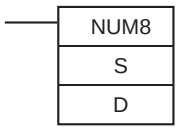
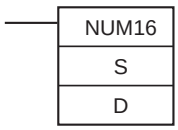
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BINARY TO BCD BCD @BCD 024	BCD(024) S R **S**: Source word **R**: Result word	Converts a word of binary data to a word of BCD data. 	Output Required	487
DOUBLE BINARY TO DOUBLE BCD BCDL @BCDL 059	BCDL(059) S R **S**: 1st source word **R**: 1st result word	Converts 8-digit hexadecimal (32-bit binary) data to 8-digit BCD data. 	Output Required	489
2'S COMPLEMENT NEG @NEG 160	NEG(160) S R **S**: Source word **R**: Result word	Calculates the 2's complement of a word of hexadecimal data. 	Output Required	491
DOUBLE 2'S COMPLEMENT NEGL @NEGL 161	NEGL(161) S R **S**: 1st source word **R**: 1st result word	Calculates the 2's complement of two words of hexadecimal data. 	Output Required	493
16-BIT TO 32-BIT SIGNED BINARY SIGN @SIGN 600	SIGN(600) S R **S**: Source word **R**: 1st result word	Expands a 16-bit signed binary value to its 32-bit equivalent. 	Output Required	494

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DATA DECODER MLPX @MLPX 076	MLPX(076) S C R **S**: Source word **C**: Control word **R**: 1st result word	Reads the numerical value in the specified digit (or byte) in the source word, turns ON the corresponding bit in the result word (or 16-word range), and turns OFF all other bits in the result word (or 16-word range). 4-to-16 bit conversion  8-to-256 bit conversion 	Output Required	496

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DATA ENCODER DMPX @DMPX 077	DMPX(077) S R C **S**: 1st source word **R**: Result word **C**: Control word	FInds the location of the first or last ON bit within the source word (or 16-word range), and writes that value to the specified digit (or byte) in the result word. 16-to-4 bit conversion  256-to-8 bit conversion 	Output Required	500
ASCII CONVERT ASC @ASC 086	ASC(086) S Di D **S**: Source word **Di**: Digit designator **D**: 1st destination word	Converts 4-bit hexadecimal digits in the source word into their 8-bit ASCII equivalents. 	Output Required	504

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ASCII TO HEX HEX @HEX 162	HEX(162) S Di D **S**: 1st source word **Di**: Digit designator **D**: Destination word	Converts up to 4 bytes of ASCII data in the source word to their hexadecimal equivalents and writes these digits in the specified destination word. 	Output Required	508
COLUMN TO LINE LINE @LINE 063	LINE(063) S N D **S**: 1st source word **N**: Bit number **D**: Destination word	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. 	Output Required	512
LINE TO COLUMN COLM @COLM 064	COLM(064) S D N **S**: Source word **D**: 1st destination word **N**: Bit number	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). 	Output Required	514

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SIGNED BCD TO BINARY BINS @BINS 470	BINS(470) C S D **C**: Control word **S**: Source word **D**: Destination word	Converts one word of signed BCD data to one word of signed binary data. 	Output Required	517
DOUBLE SIGNED BCD TO BINARY BISL @BISL 472	BISL(472) C S D **C**: Control word **S**: 1st source word **D**: 1st destination word	Converts double signed BCD data to double signed binary data. 	Output Required	520
SIGNED BINARY TO BCD BCDS @BCDS 471	BCDS(471) C S D **C**: Control word **S**: Source word **D**: Destination word	Converts one word of signed binary data to one word of signed BCD data. 	Output Required	523
DOUBLE SIGNED BINARY TO BCD BDSL @BDSL 473	BDSL(473) C S D **C**: Control word **S**: 1st source word **D**: 1st destination word	Converts double signed binary data to double signed BCD data. 	Output Required	525
GRAY CODE CONVERSION GRY 474 (CS/CJ-series Unit Ver. 2.0 or later only, including CS1-H, CJ1-H, and CJ1M CPU Units from lot number 030201 and later)	GRY (474) C S D **C**: Control word **S**: Source word **D**: 1st destination word	Converts the Gray code data in the specified word to binary, BCD, or angle (°) data at the specified resolution.	Output Required	529

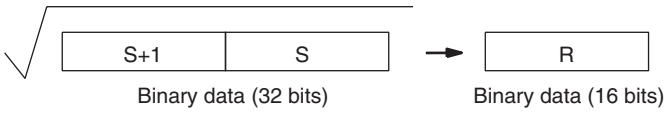
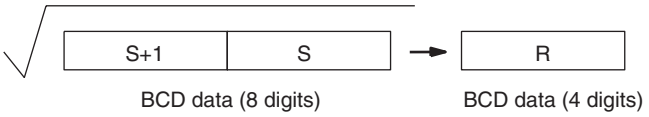
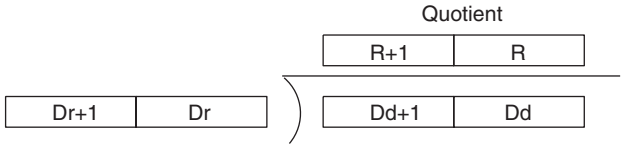
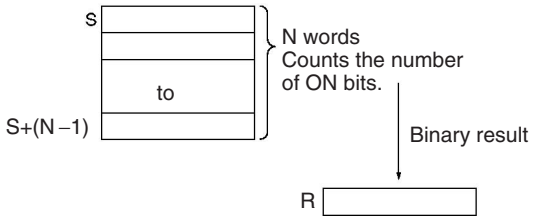
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FOUR-DIGIT NUMBER TO ASCII STR4 @STR4 601 (CS/CJ-series CPU Units with unit version 4.0 or later only)	 **S**: Numeric **D**: ASCII text	Converts a 4-digit hexadecimal number (#0000 to #FFFF) to ASCII data (4 characters).	Output Required	534
EIGHT-DIGIT NUMBER TO-ASCII STR8 @STR8 602 (CS/CJ-series CPU Units with unit version 4.0 or later only)	 **S**: Numeric **D**: ASCII text	Converts an 8-digit hexadecimal number (#0000 0000 to #FFFF FFFF) to ASCII data (8 characters).	Output Required	537
SIXTEEN-DIGIT NUMBER TO ASCII STR16 @STR16 603 (CS/CJ-series CPU Units with unit version 4.0 or later only)	 **S**: Numeric **D**: ASCII text	Converts a 16-digit hexadecimal number (#0000 0000 0000 0000 to #FFFF FFFF FFFF FFFF) to ASCII data (16 characters).	Output Required	539
ASCII TO FOUR-DIGIT NUMBER NUM4 @NUM4 604 (CS/CJ-series CPU Units with unit version 4.0 or later only)	 **S**: ASCII text **D**: Numeric	Converts 4 characters of ASCII data to a 4-digit hexadecimal number.	Output Required	541
ASCII TO EIGHT-DIGIT NUMBER NUM8 @NUM8 605 (CS/CJ-series CPU Units with unit version 4.0 or later only)	 **S**: ASCII text **D**: Numeric	Converts 8 characters of ASCII data to an 8-digit hexadecimal number.	Output Required	544
ASCII TO SIX-TEEN-DIGIT NUMBER NUM16 @NUM16 606 (CS/CJ-series CPU Units with unit version 4.0 or later only)	 **S**: ASCII text **D**: Numeric	Converts 16 characters of ASCII data to a 16-digit hexadecimal number.	Output Required	545

2-2-11 Logic Instructions

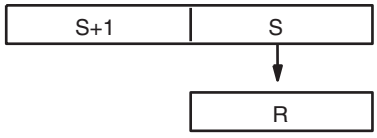
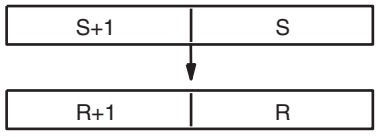
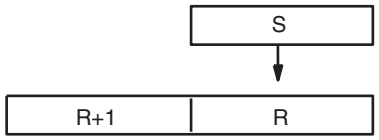
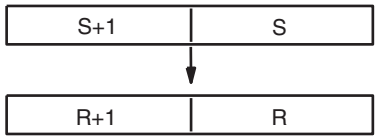
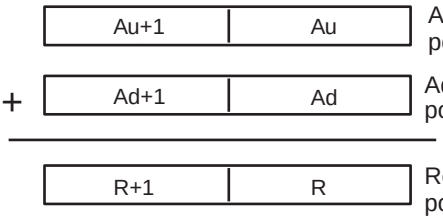
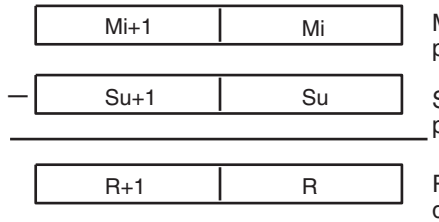
<table>
<tr><th>Instruction
Mnemonic
Code</th><th>Symbol/Operand</th><th>Function</th><th>Location
Execution condition</th><th>Page</th></tr>
<tr><td>LOGICAL AND
ANDW
@ANDW
034</td><td>ANDW(034)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical AND of corresponding bits in single words of word data and/or constants.
$I_1 . I_2 \rightarrow R$
<table><tr><th>I_1</th><th>I_2</th><th>R</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>548</td></tr>
<tr><td>DOUBLE
LOGICAL AND
ANDL
@ANDL
610</td><td>ANDL(610)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical AND of corresponding bits in double words of word data and/or constants.
$(I_1, I_1+1) . (I_2, I_2+1) \rightarrow (R, R+1)$
<table><tr><th>I_1, I_1+1</th><th>I_2, I_2+1</th><th>R, R+1</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>550</td></tr>
<tr><td>LOGICAL OR
ORW
@ORW
035</td><td>ORW(035)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical OR of corresponding bits in single words of word data and/or constants.
$I_1 + I_2 \rightarrow R$
<table><tr><th>I_1</th><th>I_2</th><th>R</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>551</td></tr>
<tr><td>DOUBLE
LOGICAL OR
ORWL
@ORWL
611</td><td>ORWL(611)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical OR of corresponding bits in double words of word data and/or constants.
$(I_1, I_1+1) + (I_2, I_2+1) \rightarrow (R, R+1)$
<table><tr><th>I_1, I_1+1</th><th>I_2, I_2+1</th><th>R, R+1</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>553</td></tr>
<tr><td>EXCLUSIVE OR
XORW
@XORW
036</td><td>XORW(036)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical exclusive OR of corresponding bits in single words of word data and/or constants.
$I_1 . \overline{I_2} + \overline{I_1} . I_2 \rightarrow R$
<table><tr><th>I_1</th><th>I_2</th><th>R</th></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>555</td></tr>
</table>

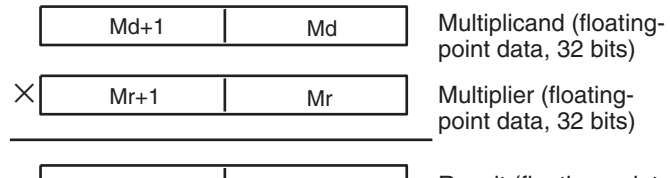
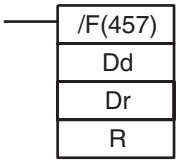
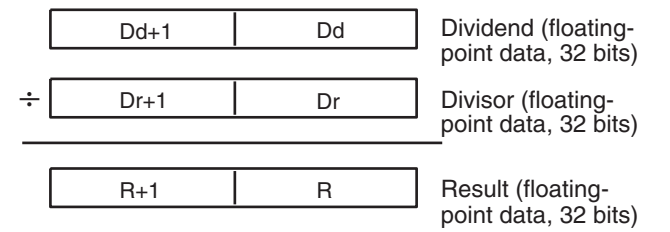
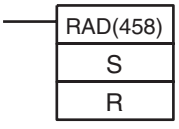
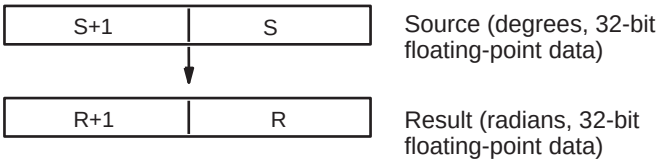
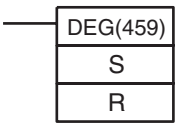
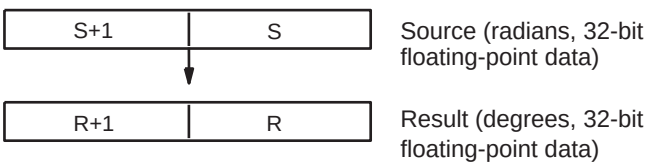
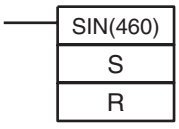
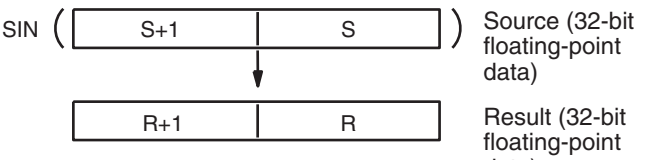
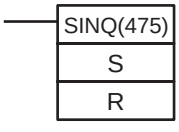
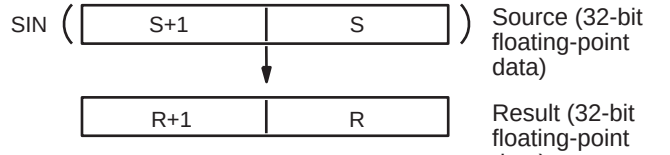
<table>
<tr><th>Instruction
Mnemonic
Code</th><th>Symbol/Operand</th><th>Function</th><th>Location
Execution condition</th><th>Page</th></tr>
<tr><td>DOUBLE EXCLUSIVE OR
XORL
@XORL
612</td><td>XORL(612)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical exclusive OR of corresponding bits in double words of word data and/or constants.
$(I_1,I_1+1).(\overline{I_2,I_2+1})+(\overline{I_1,I_1+1}).(I_2,I_2+1)\rightarrow(R,R+1)$
<table><tr><th>I_1,I_1+1</th><th>I_2,I_2+1</th><th>R, R+1</th></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>557</td></tr>
<tr><td>EXCLUSIVE NOR
XNRW
@XNRW
037</td><td>XNRW(037)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical exclusive NOR of corresponding single words of word data and/or constants.
$I_1.I_2+\overline{I_1}.\overline{I_2}\rightarrow R$
<table><tr><th>I_1</th><th>I_2</th><th>R</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table></td><td>Output
Required</td><td>559</td></tr>
<tr><td>DOUBLE EXCLUSIVE NOR
XNRL
@XNRL
613</td><td>XNRL(613)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: 1st result word</td><td>Takes the logical exclusive NOR of corresponding bits in double words of word data and/or constants.
$(I_1,I_1+1).(I_2,I_2+1)+(\overline{I_1,I_1+1}).(\overline{I_2,I_2+1})\rightarrow(R,R+1)$
<table><tr><th>I_1,I_1+1</th><th>I_2,I_2+1</th><th>R, R+1</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table></td><td>Output
Required</td><td>560</td></tr>
<tr><td>COMPLEMENT
COM
@COM
029</td><td>COM(029)
Wd
Wd: Word</td><td>Turns OFF all ON bits and turns ON all OFF bits in Wd.
$\overline{Wd}\rightarrow Wd$: 1 → 0 and 0 → 1</td><td>Output
Required</td><td>562</td></tr>
<tr><td>DOUBLE COMPLEMENT
COML
@COML
614</td><td>COML(614)
Wd
Wd: Word</td><td>Turns OFF all ON bits and turns ON all OFF bits in Wd and Wd+1.
$\overline{(Wd+1.\ Wd)}\rightarrow(Wd+1.\ Wd)$</td><td>Output
Required</td><td>564</td></tr>
</table>

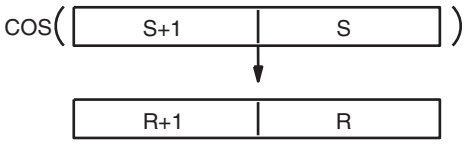
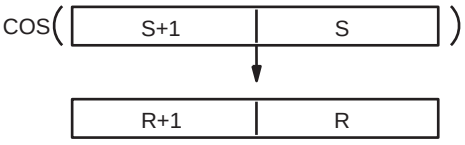
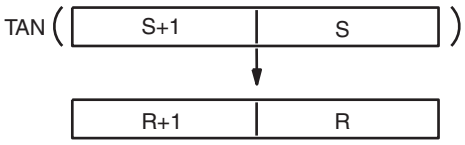
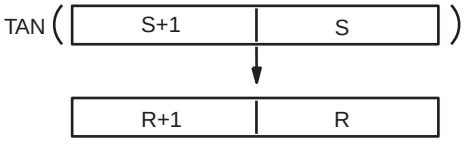
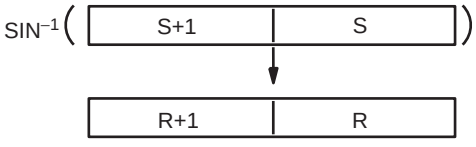
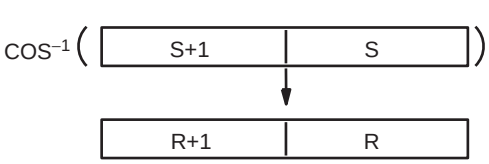
2-2-12 Special Math Instructions

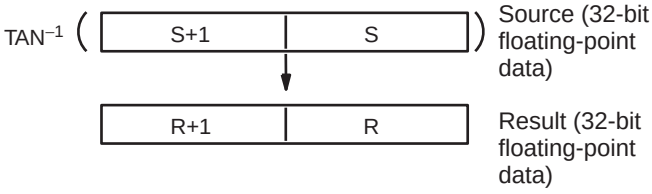
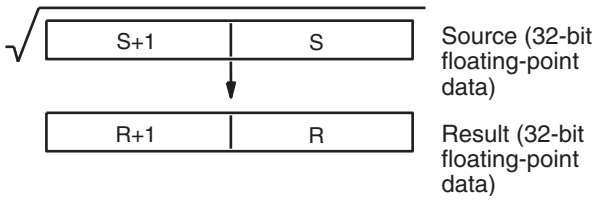
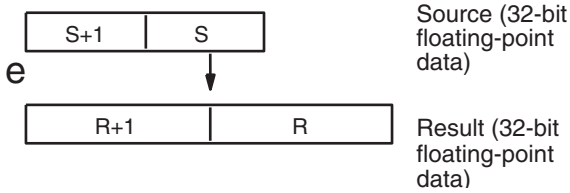
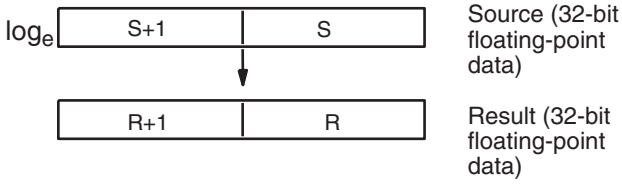
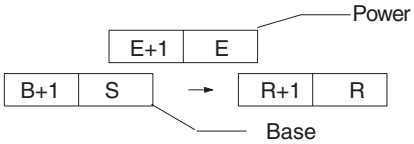
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BINARY ROOT ROTB @ROTB 620	ROTB(620) S R **S**: 1st source word **R**: Result word	Computes the square root of the 32-bit binary content of the specified words and outputs the integer portion of the result to the specified result word. 	Output Required	565
BCD SQUARE ROOT ROOT @ROOT 072	ROOT(072) S R **S**: 1st source word **R**: Result word	Computes the square root of an 8-digit BCD number and outputs the integer portion of the result to the specified result word. 	Output Required	567
ARITHMETIC PROCESS APR @APR 069	APR(069) C S R **C**: Control word **S**: Source data **R**: Result word	Calculates the sine, cosine, or a linear extrapolation of the source data. The linear extrapolation function allows any relationship between X and Y to be approximated with line segments.	Output Required	571
FLOATING POINT DIVIDE FDIV @FDIV 079	FDIV(079) Dd Dr R **Dd**: 1st dividend word **Dr**: 1st divisor word **R**: 1st result word	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). 	Output Required	583
BIT COUNTER BCNT @BCNT 067	BCNT(067) N S R **N**: Number of words **S**: 1st source word **R**: Result word	Counts the total number of ON bits in the specified word(s). 	Output Required	587

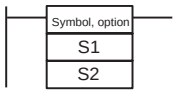
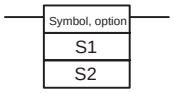
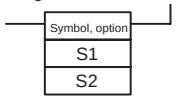
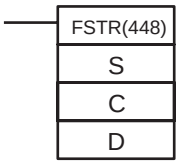
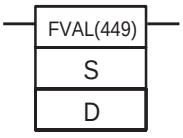
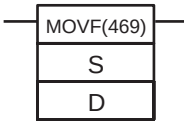
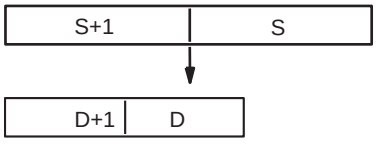
2-2-13 Floating-point Math Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FLOATING TO 16-BIT FIX @FIX 450	FIX(450) S R **S**: 1st source word **R**: Result word	Converts a 32-bit floating-point value to 16-bit signed binary data and places the result in the specified result word.  Floating-point data (32 bits) Signed binary data (16 bits)	Output Required	594
FLOATING TO 32-BIT FIXL @FIXL 451	FIXL(451) S R **S**: 1st source word **R**: 1st result word	Converts a 32-bit floating-point value to 32-bit signed binary data and places the result in the specified result words.  Floating-point data (32 bits) Signed binary data (32 bits)	Output Required	596
16-BIT TO FLOATING FLT @FLT 452	FLT(452) S R **S**: Source word **R**: 1st result word	Converts a 16-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.  Signed binary data (16 bits) Floating-point data (32 bits)	Output Required	597
32-BIT TO FLOATING FLTL @FLTL 453	FLTL(453) S R **S**: 1st source word **R**: 1st result word	Converts a 32-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.  Signed binary data (32 bits) Floating-point data (32 bits)	Output Required	599
FLOATING-POINT ADD +F @+F 454	+F(454) Au Ad R **Au**: 1st augend word **AD**: 1st addend word **R**: 1st result word	Adds two 32-bit floating-point numbers and places the result in the specified result words.  Augend (floating-point data, 32 bits) Addend (floating-point data, 32 bits) Result (floating-point data, 32 bits)	Output Required	601
FLOATING-POINT SUB-TRACT −F @−F 455	F(455) Mi Su R **Mi**: 1st Minuend word **Su**: 1st Subtrahend word **R**: 1st result word	Subtracts one 32-bit floating-point number from another and places the result in the specified result words.  Minuend (floating-point data, 32 bits) Subtrahend (floating-point data, 32 bits) Result (floating-point data, 32 bits)	Output Required	603

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FLOATING-POINT MULTIPLY *F @*F 456	 **Md**: 1st Multiplicand word **Mr**: 1st Multiplier word **R**: 1st result word	Multiplies two 32-bit floating-point numbers and places the result in the specified result words. 	Output Required	605
FLOATING-POINT DIVIDE /F @/F 457	 **Dd**: 1st Dividend word **Dr**: 1st Divisor word **R**: 1st result word	Divides one 32-bit floating-point number by another and places the result in the specified result words. 	Output Required	607
DEGREES TO RADIANS RAD @RAD 458	 **S**: 1st source word **R**: 1st result word	Converts a 32-bit floating-point number from degrees to radians and places the result in the specified result words. 	Output Required	609
RADIANS TO DEGREES DEG @DEG 459	 **S**: 1st source word **R**: 1st result word	Converts a 32-bit floating-point number from radians to degrees and places the result in the specified result words. 	Output Required	610
SINE SIN @SIN 460	 **S**: 1st source word **R**: 1st result word	Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words. 	Output Required	612
HIGH-SPEED SINE (CJ1-H-R only) SINQ @SINQ 475	 **S**: 1st source word **R**: 1st result word	Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words. 	Output Required	614

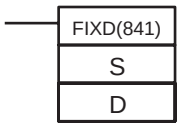
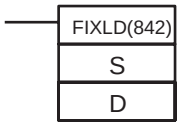
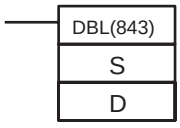
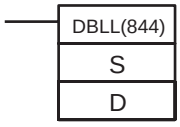
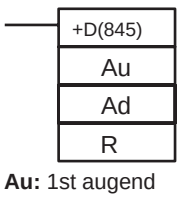
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
COSINE COS @COS 461	COS(461) S R **S**: 1st source word **R**: 1st result word	Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words.  Source (32-bit floating-point data) Result (32-bit floating-point data)	Output Required	615
HIGH-SPEED COSINE (CJ1-H-R only) COSQ @COSQ 476	COSQ(476) S R **S**: 1st source word **R**: 1st result word	Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words.  Source (32-bit floating-point data) Result (32-bit floating-point data)	Output Required	617
TANGENT TAN @TAN 462	TAN(462) S R **S**: 1st source word **R**: 1st result word	Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words.  Source (32-bit floating-point data) Result (32-bit floating-point data)	Output Required	619
HIGH-SPEED TANGENT (CJ1-H-R only) TANQ @TANQ 477	TANQ(477) S R **S**: 1st source word **R**: 1st result word	Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words.  Source (32-bit floating-point data) Result (32-bit floating-point data)	Output Required	621
ARC SINE ASIN @ASIN 463	ASIN(463) S R **S**: 1st source word **R**: 1st result word	Calculates the arc sine of a 32-bit floating-point number and places the result in the specified result words. (The arc sine function is the inverse of the sine function; it returns the angle that produces a given sine value between –1 and 1.)  Source (32-bit floating-point data) Result (32-bit floating-point data)	Output Required	623
ARC COSINE ACOS @ACOS 464	ACOS(464) S R **S**: 1st source word **R**: 1st result word	Calculates the arc cosine of a 32-bit floating-point number and places the result in the specified result words. (The arc cosine function is the inverse of the cosine function; it returns the angle that produces a given cosine value between –1 and 1.)  Source (32-bit floating-point data) Result (32-bit floating-point data)	Output Required	625

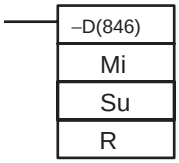
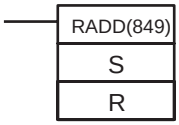
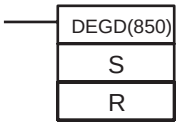
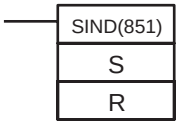
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ARC TANGENT ATAN @ATAN 465	ATAN(465) S R **S**: 1st source word **R**: 1st result word	Calculates the arc tangent of a 32-bit floating-point number and places the result in the specified result words. (The arc tangent function is the inverse of the tangent function; it returns the angle that produces a given tangent value.) 	Output Required	627
SQUARE ROOT SQRT @SQRT 466	SQRT(466) S R **S**: 1st source word **R**: 1st result word	Calculates the square root of a 32-bit floating-point number and places the result in the specified result words. 	Output Required	629
EXPONENT EXP @EXP 467	EXP(467) S R **S**: 1st source word **R**: 1st result word	Calculates the natural (base e) exponential of a 32-bit floating-point number and places the result in the specified result words. 	Output Required	631
LOGARITHM LOG @LOG 468	LOG(468) S R **S**: 1st source word **R**: 1st result word	Calculates the natural (base e) logarithm of a 32-bit floating-point number and places the result in the specified result words. 	Output Required	633
EXPONENTIAL POWER PWR @PWR 840	PWR(840) B E R **B**: 1st base word **E**: 1st exponent word **R**: 1st result word	Raises a 32-bit floating-point number to the power of another 32-bit floating-point number. 	Output Required	635

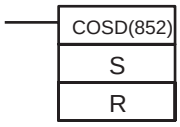
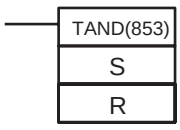
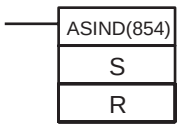
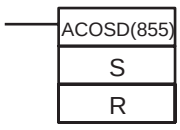
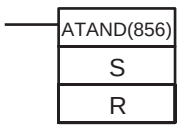
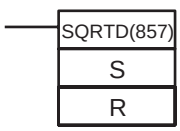
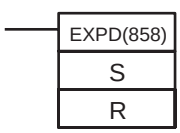
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FLOATING SYMBOL COMPARISON (CS1-H, CJ1-H, CJ1M, or CS1D only) LD, AND. or OR + =F (329), <>F (330), <F (331), <=F (332), >F (333), or >=F (334)	Using LD:  Using AND:  Using OR:  **S1**: Comparison data 1 **S2**: Comparison data 2	Compares the specified single-precision data (32 bits) or constants and creates an ON execution condition if the comparison result is true. Three kinds of symbols can be used with the floating-point symbol comparison instructions: LD (Load), AND, and OR.	LD: Not required AND or OR: Required	636
FLOATING-POINT TO ASCII (CS1-H, CJ1-H, CJ1M, or CS1D only) FSTR @FSTR 448	 **S:** 1st source word **C**: Control word **D**: Destination word	Converts the specified single-precision floating-point data (32-bit decimal-point or exponential format) to text string data (ASCII) and outputs the result to the destination word.	Output required	640
ASCII TO FLOATING-POINT (CS1-H, CJ1-H, CJ1M, or CS1D only) FVAL @FVAL 449	 **S:** Source word **D**: 1st destination word	Converts the specified text string (ASCII) representation of single-precision floating-point data (decimal-point or exponential format) to 32-bit single-precision floating-point data and outputs the result to the destination words.	Output required	645
MOVE FLOATING-POINT (SINGLE) (CJ1-H-R only) MOVF @MOVF 469	 **S**: First source word **D**: First destination word	Transfers the specified 32-bit floating-point number to the destination words.  	Output required	649

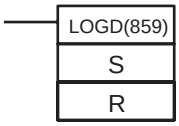
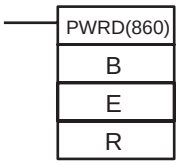
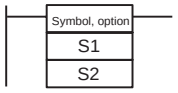
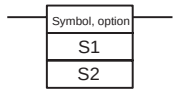
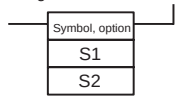
2-2-14 Double-precision Floating-point Instructions

The Double-precision Floating-point Instructions are supported only by the CS1-H, CJ1-H, CJ1M, or CS1D CPU Units.

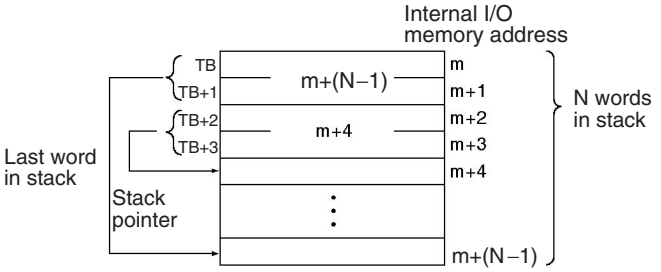
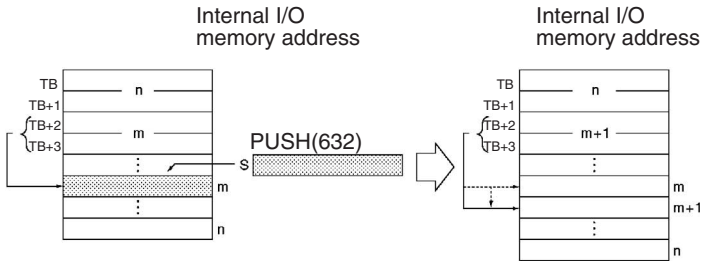
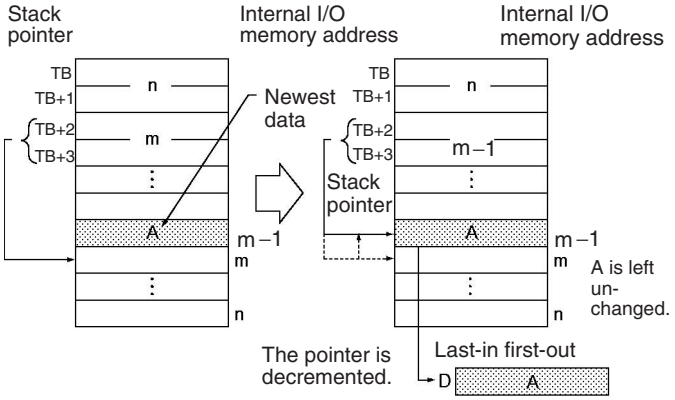
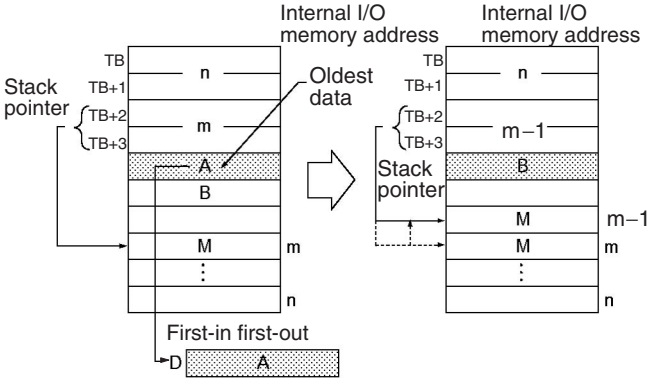
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE FLOATING TO 16-BIT BINARY FIXD @FIXD 841	 **S:** 1st source word **D**: Destination word	Converts the specified double-precision floating-point data (64 bits) to 16-bit signed binary data and outputs the result to the destination word.	Output Required	657
DOUBLE FLOATING TO 32-BIT BINARY FIXLD @FIXLD 842	 **S:** 1st source word **D**: 1st destination word	Converts the specified double-precision floating-point data (64 bits) to 32-bit signed binary data and outputs the result to the destination words.	Output Required	658
16-BIT BINARY TO DOUBLE FLOATING DBL @DBL 843	 **S:** Source word **D**: 1st destination word	Converts the specified 16-bit signed binary data to double-precision floating-point data (64 bits) and outputs the result to the destination words.	Output Required	660
32-BIT BINARY TO DOUBLE FLOATING DBLL @DBLL 844	 **S:** 1st source word **D**: 1st destination word	Converts the specified 32-bit signed binary data to double-precision floating-point data (64 bits) and outputs the result to the destination words.	Output Required	661
DOUBLE FLOATING-POINT ADD +D @+D 845	 **Au:** 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds the specified double-precision floating-point values (64 bits each) and outputs the result to the result words.	Output Required	663

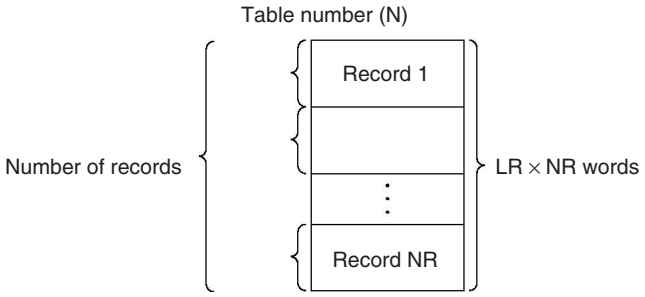
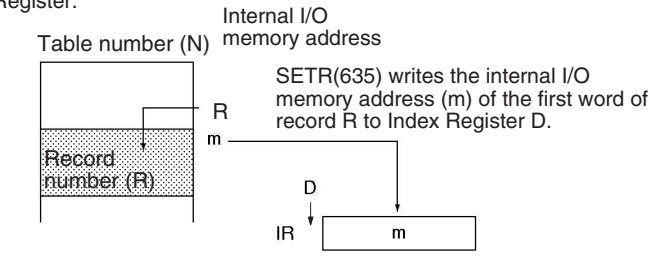
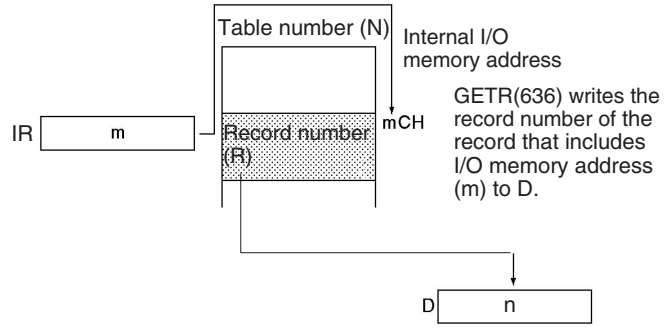
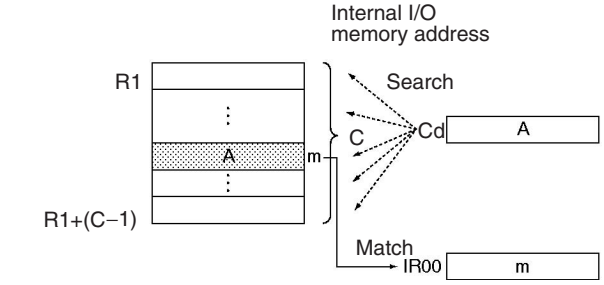
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE FLOATING-POINT SUBTRACT −D @−D 846	 **Mi:** 1st minuend word **Su**: 1st subtrahend word **R**: 1st result word	Subtracts the specified double-precision floating-point values (64 bits each) and outputs the result to the result words.	Output Required	665
DOUBLE FLOATING-POINT MULTIPLY *D @*D 847	 **Md:** 1st multiplicand word **Mr**: 1st multiplier word **R**: 1st result word	Multiplies the specified double-precision floating-point values (64 bits each) and outputs the result to the result words.	Output Required	667
DOUBLE FLOATING-POINT DIVIDE /D @/D 848	 **Dd:** 1st Dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides the specified double-precision floating-point values (64 bits each) and outputs the result to the result words.	Output Required	669
DOUBLE DEGREES TO RADIANS RADD @RADD 849	 **S:** 1st source word **R**: 1st result word	Converts the specified double-precision floating-point data (64 bits) from degrees to radians and outputs the result to the result words.	Output Required	671
DOUBLE RADIANS TO DEGREES DEGD @DEGD 850	 **S:** 1st source word **R**: 1st result word	Converts the specified double-precision floating-point data (64 bits) from radians to degrees and outputs the result to the result words.	Output Required	673
DOUBLE SINE SIND @SIND 851	 **S:** 1st source word **R**: 1st result word	Calculates the sine of the angle (radians) in the specified double-precision floating-point data (64 bits) and outputs the result to the result words.	Output Required	674

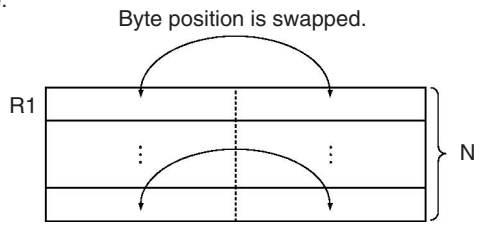
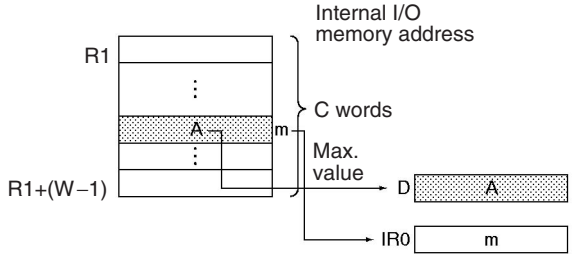
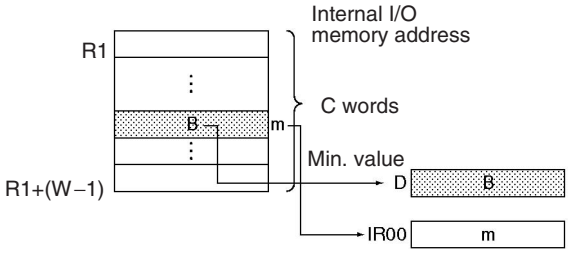
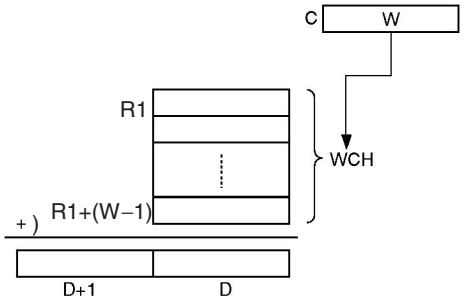
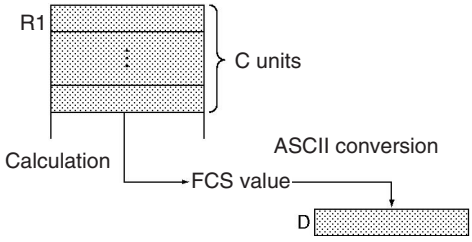
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE COSINE COSD @COSD 852	 **S:** 1st source word **R**: 1st result word	Calculates the cosine of the angle (radians) in the specified double-precision floating-point data (64 bits) and outputs the result to the result words.	Output Required	676
DOUBLE TANGENT TAND @TAND 853	 **S:** 1st source word **R**: 1st result word	Calculates the tangent of the angle (radians) in the specified double-precision floating-point data (64 bits) and outputs the result to the result words.	Output Required	678
DOUBLE ARC SINE ASIND @ASIND 854	 **S:** 1st source word **R**: 1st result word	Calculates the angle (in radians) from the sine value in the specified double-precision floating-point data (64 bits) and outputs the result to the result words. (The arc sine function is the inverse of the sine function; it returns the angle that produces a given sine value between -1 and 1.)	Output Required	680
DOUBLE ARC COSINE ACOSD @ACOSD 855	 **S:** 1st source word **R**: 1st result word	Calculates the angle (in radians) from the cosine value in the specified double-precision floating-point data (64 bits) and outputs the result to the result words. (The arc cosine function is the inverse of the cosine function; it returns the angle that produces a given cosine value between -1 and 1.)	Output Required	682
DOUBLE ARC TANGENT ATAND @ATAND 856	 **S:** 1st source word **R**: 1st result word	Calculates the angle (in radians) from the tangent value in the specified double-precision floating-point data (64 bits) and outputs the result to the result words. (The arc tangent function is the inverse of the tangent function; it returns the angle that produces a given tangent value.)	Output Required	684
DOUBLE SQUARE ROOT SQRTD @SQRTD 857	 **S:** 1st source word **R**: 1st result word	Calculates the square root of the specified double-precision floating-point data (64 bits) and outputs the result to the result words.	Output Required	686
DOUBLE EXPONENT EXPD @EXPD 858	 **S:** 1st source word **R**: 1st result word	Calculates the natural (base e) exponential of the specified double-precision floating-point data (64 bits) and outputs the result to the result words.	Output Required	688

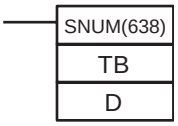
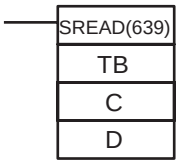
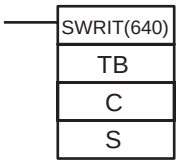
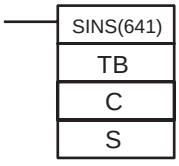
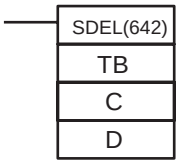
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE LOGARITHM LOGD @LOGD 859	 **S:** 1st source word **R**: 1st result word	Calculates the natural (base e) logarithm of the specified double-precision floating-point data (64 bits) and outputs the result to the result words.	Output Required	690
DOUBLE EXPONENTIAL POWER PWRD @PWRD 860	 **B:** 1st base word **E**: 1st exponent word **R**: 1st result word	Raises a double-precision floating-point number (64 bits) to the power of another double-precision floating-point number and outputs the result to the result words.	Output Required	692
DOUBLE SYMBOL COMPARISON LD, AND. or OR + =D (335), <>D (336), <D (337), <=D (338), >D (339), or >=D (340)	Using LD:  Using AND:  Using OR:  **S1**: Comparison data 1 **S2**: Comparison data 2	Compares the specified double-precision data (64 bits) and creates an ON execution condition if the comparison result is true. Three kinds of symbols can be used with the floating-point symbol comparison instructions: LD (Load), AND, and OR.	LD: Not required AND or OR: Required	694

2-2-15 Table Data Processing Instructions

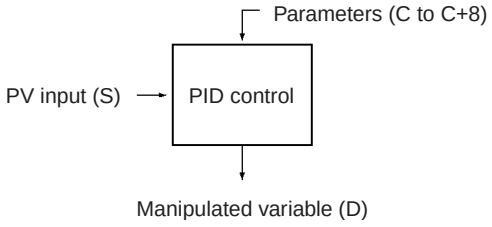
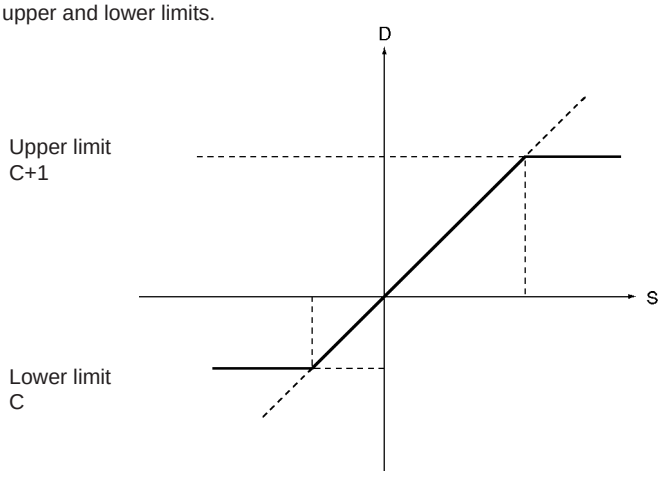
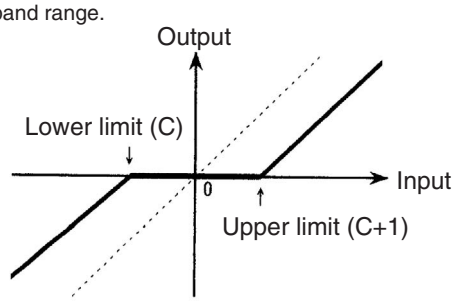
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SET STACK SSET @SSET 630	SSET(630) TB N **TB**: 1st stack address **N**: Number of words	Defines a stack of the specified length beginning at the specified word and initializes the words in the data region to all zeroes. 	Output Required	703
PUSH ONTO STACK PUSH @PUSH 632	PUSH(632) TB S **TB**: 1st stack address **S**: Source word	Writes one word of data to the specified stack. 	Output Required	706
LAST IN FIRST OUT LIFO @LIFO 634	LIFO(634) TB D **TB**: 1st stack address **D**: Destination word	Reads the last word of data written to the specified stack (the newest data in the stack). 	Output Required	712
FIRST IN FIRST OUT FIFO @FIFO 633	FIFO(633) TB D **TB**: 1st stack address **D**: Destination word	Reads the first word of data written to the specified stack (the oldest data in the stack). 	Output Required	709

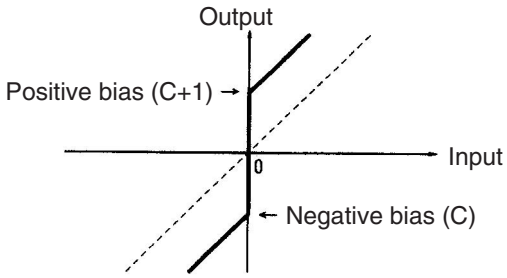
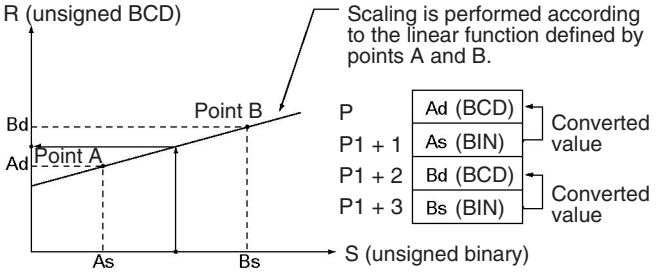
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DIMENSION RECORD TABLE DIM @DIM 631	DIM(631) N LR NR TB **N**: Table number **LR**: Length of each record **NR**: Number of records **TB**: 1st table word	Defines a record table by declaring the length of each record and the number of records. Up to 16 record tables can be defined. 	Output Required	715
SET RECORD LOCATION SETR @SETR 635	SETR(635) N R D **N**: Table number **R**: Record number **D**: Destination Index Register	Writes the location of the specified record (the internal I/O memory address of the beginning of the record) in the specified Index Register. 	Output Required	718
GET RECORD NUMBER GETR @GETR 636	GETR(636) N IR D **N**: Table number **IR**: Index Register **D**: Destination word	Returns the record number of the record at the internal I/O memory address contained in the specified Index Register. 	Output Required	720
DATA SEARCH SRCH @SRCH 181	SRCH(181) C R1 Cd **C**: 1st control word **R1**: 1st word in range **Cd**: Comparison data	Searches for a word of data within a range of words. 	Output Required	722

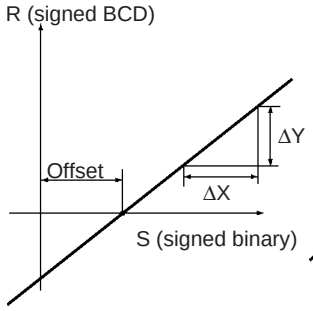
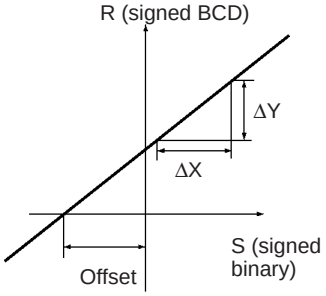
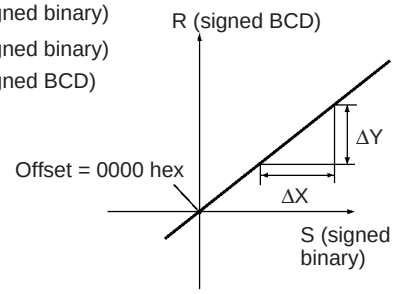
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SWAP BYTES SWAP @SWAP 637	SWAP(637) N R1 **N**: Number of words **R1**: 1st word in range	Switches the leftmost and rightmost bytes in all of the words in the range. Byte position is swapped. 	Output Required	725
FIND MAXIMUM MAX @MAX 182	MAX(182) C R1 D **C**: 1st control word **R1**: 1st word in range **D**: Destination word	Finds the maximum value in the range. 	Output Required	727
FIND MINIMUM MIN @MIN 183	MIN(183) C R1 D **C**: 1st control word **R1**: 1st word in range **D**: Destination word	Finds the minimum value in the range. 	Output Required	731
SUM SUM @SUM 184	SUM(184) C R1 D **C**: 1st control word **R1**: 1st word in range **D**: 1st destination word	Adds the bytes or words in the range and outputs the result to two words. 	Output Required	735
FRAME CHECK-SUM FCS @FCS 180	FCS(180) C R1 D **C**: 1st control word **R1**: 1st word in range **D**: 1st destination word	Calculates the ASCII FCS value for the specified range. 	Output Required	738

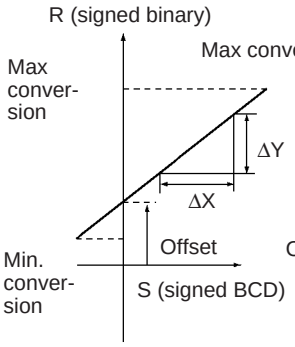
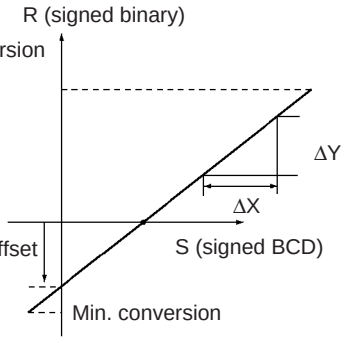
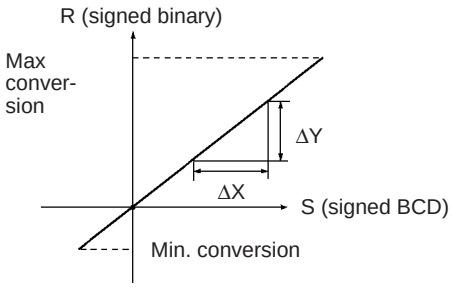
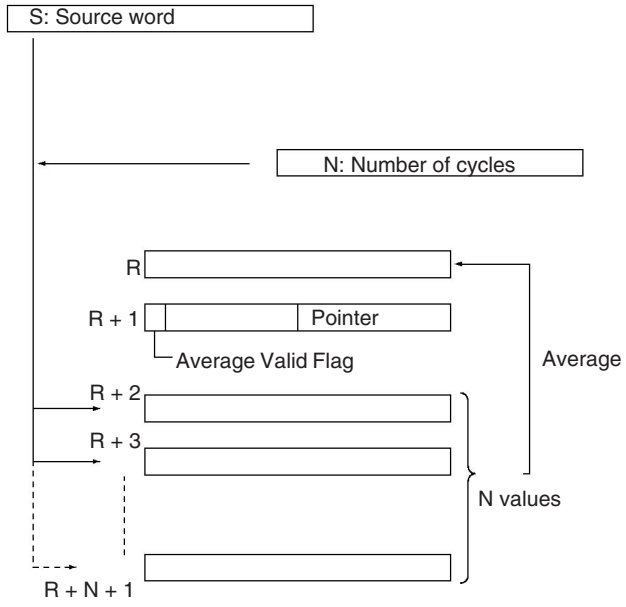
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
STACK SIZE READ (CS1-H, CJ1-H, CJ1M, or CS1D only) SNUM @SNUM 638	 **TB**: First stack address **D**: Destination word	Counts the amount of stack data (number of words) in the specified stack.	Output required	742
STACK DATA READ (CS1-H, CJ1-H, CJ1M, or CS1D only) SREAD @SREAD 639	 **TB**: First stack address **C**: Offset value **D**: Destination word	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).	Output required	744
STACK DATA OVERWRITE (CS1-H, CJ1-H, CJ1M, or CS1D only) SWRIT @SWRIT 640	 **TB**: First stack address **C**: Offset value **S**: Source data	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).	Output required	747
STACK DATA INSERT (CS1-H, CJ1-H, CJ1M, or CS1D only) SINS @SINS 641	 **TB**: First stack address **C**: Offset value **S**: Source data	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 insertion point (how many data elements before the current pointer position).	Output required	750
STACK DATA DELETE (CS1-H, CJ1-H, CJ1M, or CS1D only) SDEL @SDEL 642	 **TB**: First stack address **C**: Offset value **D**: Destination word	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 deletion point (how many data elements before the current pointer position).	Output required	753

2-2-16 Data Control Instructions

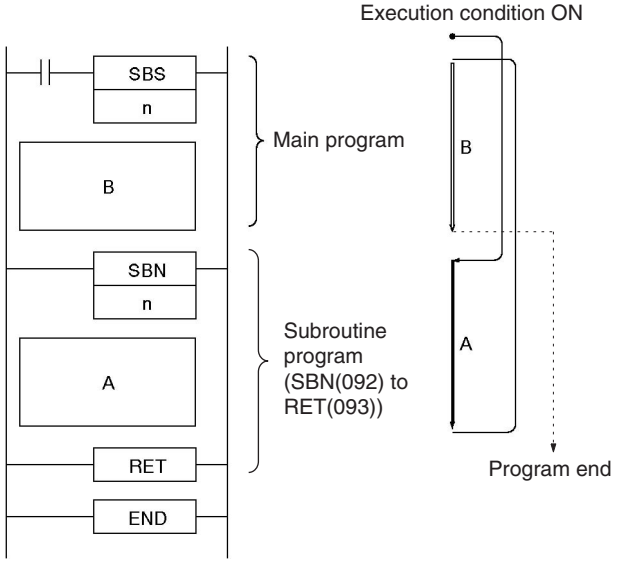
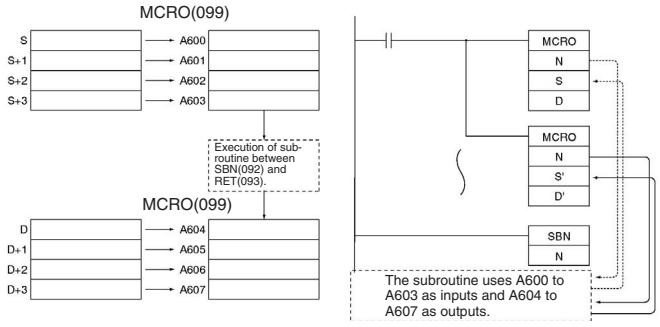
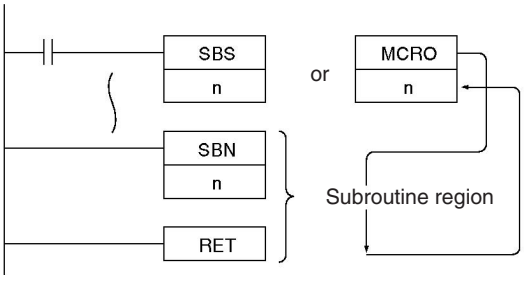
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
PID CONTROL PID 190	PID(190) S C D **S**: Input word **C**: 1st parameter word **D**: Output word	Executes PID control according to the specified parameters. 	Output Required	757
PID CONTROL WITH AUTOTUNING PIDAT 191 (CS1-H, CJ1-H, or CJ1M only)	PIDAT(191) S C D **S**: Input word **C**: 1st parameter word **D**: Output word	Executes PID control according to the specified parameters. The PID constants can be auto-tuned with PIDAT(191).	Output required	769
LIMIT CONTROL LMT @LMT 680	LMT(680) S C D **S**: Input word **C**: 1st limit word **D**: Output word	Controls output data according to whether or not input data is within upper and lower limits. 	Output Required	779
DEAD BAND CONTROL BAND @BAND 681	BAND(681) S C D **S**: Input word **C**: 1st limit word **D**: Output word	Controls output data according to whether or not input data is within the dead band range. 	Output Required	781

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DEAD ZONE CONTROL ZONE @ZONE 682	ZONE(682) S C D **S**: Input word **C**: 1st limit word **D**: Output word	Adds the specified bias to input data and outputs the result. 	Output Required	784
TIME-PROPORTIONAL OUTPUT TPO 685 (CS/CJ-series Unit Ver. 2.0 or later only)	TPO (685) S C R **S**: Input word **C**: 1st parameter word **R**: Pulse Output Bit	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.	Output Required	787
SCALING SCL @SCL 194	SCL(194) S P1 R **S**: Source word **P1**: 1st parameter word **R**: Result word	Converts unsigned binary data into unsigned BCD data according to the specified linear function. 	Output Required	795

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SCALING 2 SCL2 @SCL2 486	SCL2(486) S P1 R **S**: Source word **P1**: 1st parameter word **R**: Result word	Converts signed binary data into signed BCD data according to the specified linear function. An offset can be input in defining the linear function. **Positive Offset**  **Negative Offset**  P1 Offset (Signed binary) P1 + 1 ΔY (Signed binary) P1 + 2 ΔX (Signed BCD) **Offset of 0000** 	Output Required	800

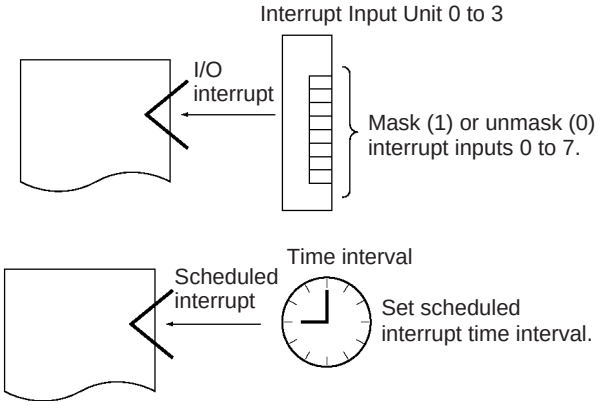
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SCALING 3 SCL3 @SCL3 487	SCL3(487) S P1 R **S**: Source word **P1**: 1st parameter word **R**: Result word	Converts signed BCD data into signed binary data according to the specified linear function. An offset can be input in defining the linear function. **Positive Offset**  **Negative Offset**  **Offset of 0000** 	Output Required	804
AVERAGE AVG 195	AVG(195) S N R **S**: Source word **N**: Number of cycles **R**: Result word	Calculates the average value of an input word for the specified number of cycles. 	Output Required	807

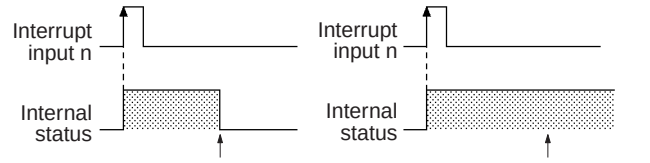
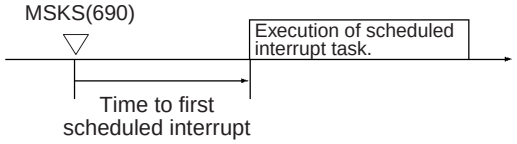
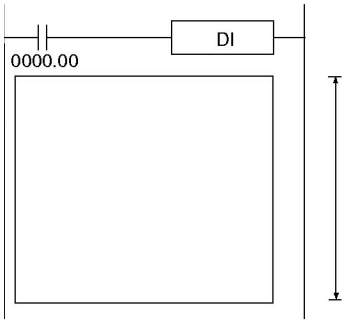
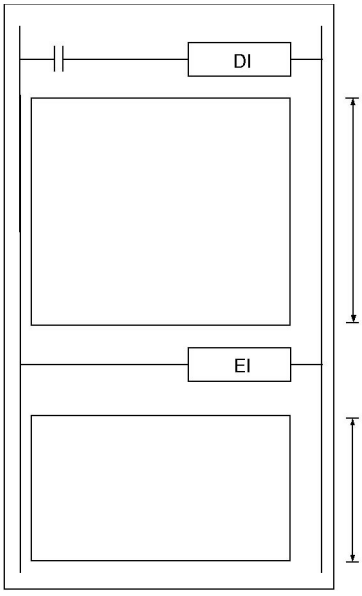
2-2-17 Subroutine Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SUBROUTINE CALL SBS @SBS 091	SBS(091) N **N**: Subroutine number	Calls the subroutine with the specified subroutine number and executes that program. 	Output Required	811
MACRO MCRO @MCRO 099	MCRO(099) N S D **N**: Subroutine number **S**: 1st input parameter word **D**: 1st output parameter word	Calls the subroutine with the specified subroutine number and executes that program using the input parameters in S to S+3 and the output parameters in D to D+3. 	Output Required	817
SUBROUTINE ENTRY SBN 092	SBN(092) N **N**: Subroutine number	Indicates the beginning of the subroutine program with the specified subroutine number. 	Output Not required	821
SUBROUTINE RETURN RET 093	RET(093)	Indicates the end of a subroutine program.	Output Not required	824

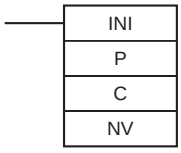
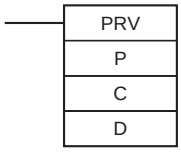
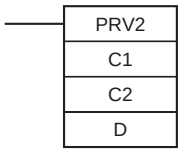
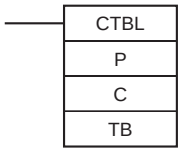
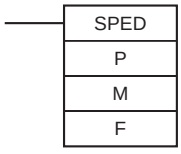
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
GLOBAL SUBROUTINE CALL (CS1-H, CJ1-H, CJ1M, or CS1D only) GSBS 750	GSBS(750) N **N**: Subroutine number	Calls the subroutine with the specified subroutine number and executes that program.	Output Not required	824
GLOBAL SUBROUTINE ENTRY (CS1-H, CJ1-H, CJ1M, or CS1D only) GSBN 751	GSBN(751) N **N**: Subroutine number	Indicates the beginning of the subroutine program with the specified subroutine number.	Output Not required	832
GLOBAL SUBROUTINE RETURN (CS1-H, CJ1-H, CJ1M, or CS1D only) GRET 752	GRET(752)	Indicates the end of a subroutine program.	Output Not required	835

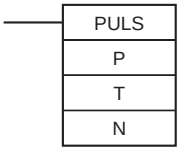
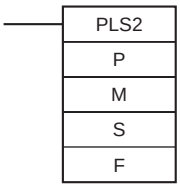
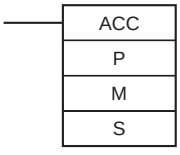
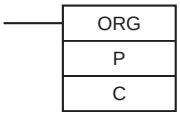
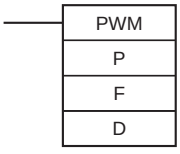
2-2-18 Interrupt Control Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SET INTERRUPT MASK (Not supported by CS1D CPU Units for Duplex-CPU Systems.) MSKS @MSKS 690	MSKS(690) N C **N**: Interrupt identifier **C**: Control data	Sets up interrupt processing for I/O interrupts or scheduled interrupts. Both I/O interrupt tasks and scheduled interrupt tasks are masked (disabled) when the PC is first turned on. MSKS(690) can be used to unmask or mask I/O interrupts and set the time intervals for scheduled interrupts. 	Output Required	839
READ INTERRUPT MASK (Not supported by CS1D CPU Units for Duplex-CPU Systems.) MSKR @MSKR 692	MSKR(692) N D **N**: Interrupt identifier **D**: Destination word	Reads the current interrupt processing settings that were set with MSKS(690).	Output Required	846

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
CLEAR INTERRUPT (Not supported by CS1D CPU Units for Duplex-CPU Systems.) CLI @CLI 691	CLI(691) N C **N**: Interrupt identifier **C**: Control data	Clears or retains recorded interrupt inputs for I/O interrupts or sets the time to the first scheduled interrupt for scheduled interrupts. N = 0 to 3  Recorded interrupt cleared Recorded interrupt retained N = 4 to 5 	Output Required	851
DISABLE INTERRUPTS DI @DI 693	DI(693)	Disables execution of all interrupt tasks except the power OFF interrupt. 	Output Required	855
ENABLE INTERRUPTS EI 694	EI(694)	Enables execution of all interrupt tasks that were disabled with DI(693). 	Output Not required	858

2-2-19 High-speed Counter and Pulse Output Instructions (CJ1M-CPU21/22/23 Only)

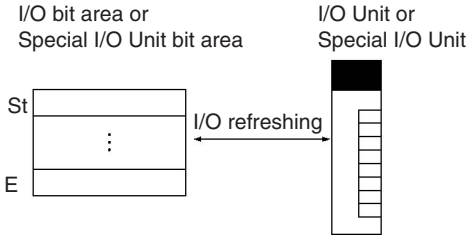
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
MODE CONTROL INI @INI 880	 **P:** Port specifier **C:** Control data **NV:** 1st word with new PV	INI(880) is used to start and stop target value comparison, to change the present value (PV) of a high-speed counter, to change the PV of an interrupt input (counter mode), to change the PV of a pulse output, or to stop pulse output.	Output Required	864
HIGH-SPEED COUNTER PV READ PRV @PRV 881	 **P:** Port specifier **C:** Control data **D:** 1st destination word	PRV(881) is used to read the present value (PV) of a high-speed counter, pulse output, or interrupt input (counter mode).	Output Required	868
COUNTER FREQUENCY CONVERT PRV2 883 (CJ1M CPU Unit Ver. 2.0 or later only)	 **C1:** Control data **C2:** Pulses/revolution **D:** 1st destination word	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. Pulses can be input from high-speed counter 0 only.	Output Required	874
COMPARISON TABLE LOAD CTBL @CTBL 882	 **P:** Port specifier **C:** Control data **TB:** 1st comparison table word	CTBL(882) is used to perform target value or range comparisons for the present value (PV) of a high-speed counter.	Output Required	878
SPEED OUTPUT SPED @SPED 885	 **P:** Port specifier **M:** Output mode **F:** 1st pulse frequency word	SPED(885) is used to specify the frequency and perform pulse output without acceleration or deceleration.	Output Required	882

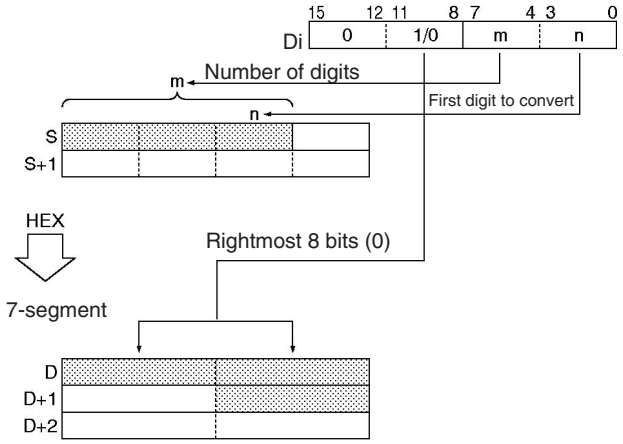
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SET PULSES PULS @PULS 886	 **P:** Port specifier **T:** Pulse type **N:** Number of pulses	PULS(886) is used to set the number of pulses for pulse output.	Output Required	887
PULSE OUTPUT PLS2 @PLS2 887	 **P:** Port specifier **M:** Output mode **S:** 1st word of settings table **F:** 1st word of starting frequency	PLS2(887) is used to set the pulse frequency and acceleration/deceleration rates, and to perform pulse output with acceleration/deceleration (with different acceleration/deceleration rates). Only positioning is possible.	Output Required	890
ACCELERATION CONTROL ACC @ACC 888	 **P:** Port specifier **M:** Output mode **S:** 1st word of settings table	ACC(888) is used to set the pulse frequency and acceleration/deceleration rates, and to perform pulse output with acceleration/deceleration (with the same acceleration/deceleration rate). Both positioning and speed control are possible.	Output Required	896
ORIGIN SEARCH ORG @ORG 889	 **P:** Port specifier **C:** Control data	ORG(889) is used to perform origin searches and returns.	Output Required	903
PULSE WITH VARIABLE DUTY FACTOR PWM @ 891	 **P:** Port specifier **F:** Frequency **D:** Duty factor	PWM(891) is used to output pulses with a variable duty factor.	Output Required	906

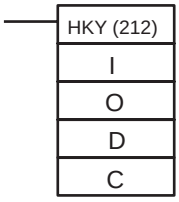
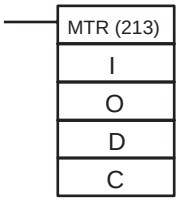
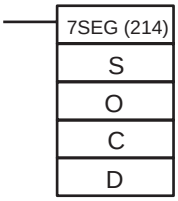
2-2-20 Step Instructions

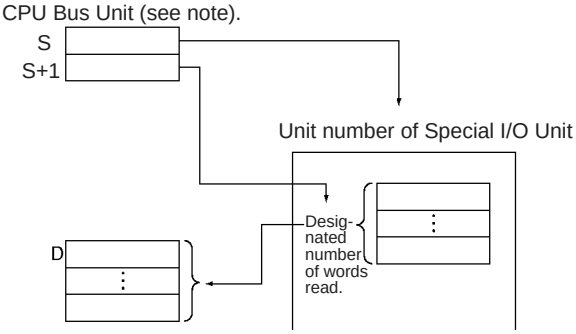
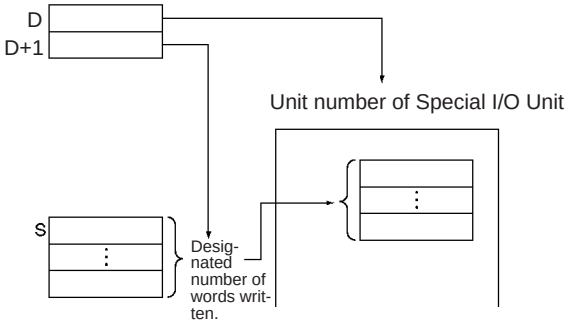
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
STEP DEFINE STEP 008	STEP(008) B **B**: Bit	STEP(008) functions in following 2 ways, depending on its position and whether or not a control bit has been specified. (1)Starts a specific step. (2)Ends the step programming area (i.e., step execution).	Output Required	909
STEP START SNXT 009	SNXT(009) B **B**: Bit	SNXT(009) is used in the following three ways: (1)To start step programming execution. (2)To proceed to the next step control bit. (3)To end step programming execution.	Output Required	909

2-2-21 Basic I/O Unit Instructions

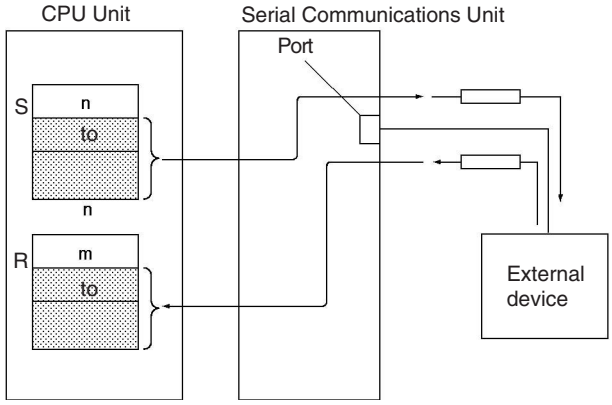
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
I/O REFRESH IORF @IORF 097	IORF(097) St E **St**: Starting word **E**: End word	Refreshes the specified I/O words. 	Output Required	926
SPECIAL I/O UNIT I/O REFRESH (CJ1-H-R only) FIORF @FIORF 225	FIORF(225) N **N**: Unit number	Performs I/O refreshing immediately for the specified Special I/O Unit's allocated CIO Area and DM Area words.t with the specified unit number.	Output Required	929
CPU BUS UNIT I/O REFRESH (CS1-H, CJ1-H, CJ1M, or CS1D only) DLNK @DLNK 226	DLNK(226) N **N**: Unit number	Immediately refreshes the I/O in the CPU Bus Unit with the specified unit number.	Output required	932

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
7-SEGMENT DECODER SDEC @SDEC 078	SDEC(078) S Di D **S**: Source word **Di**: Digit designator **D**: 1st destination word	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. 	Output Required	937
DIGITAL SWITCH INPUT DSW 210 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	DSW (210) I O D C1 C2 **I:** Data input word (D0 to D3) **O:** Output word **D:** 1st result word **C1:** Number of digits **C2:** System word	Reads the value set on an external digital switch (or thumbwheel switch) connected to an Input Unit or Output Unit and stores the 4-digit or 8-digit BCD data in the specified words.	Output Required	940
TEN KEY INPUT TKY 211 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	TKY (211) I D1 D2 **I**: Data input word **D1**: 1st register word **D2**: Key input word	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.	Output Required	945

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
HEXADECIMAL KEY INPUT HKY 212 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	 **I**: Data input word **O**: Output word **D**: 1st register word **C**: System word	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.	Output Required	948
MATRIX INPUT MTR 213 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	 **I**: Data input word **O**: Output word **D**: 1st destination word **C**: System word	Inputs up to 64 signals from an 8 × 8 matrix connected to an Input Unit and Output Unit (using 8 input points and 8 output points) and stores that 64-bit data in the 4 destination words.	Output Required	953
7-SEGMENT DISPLAY OUTPUT 7SEG 214 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	 **S**: 1st source word **O**: Output word **C**: Control data **D**: System word	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.	Output Required	957

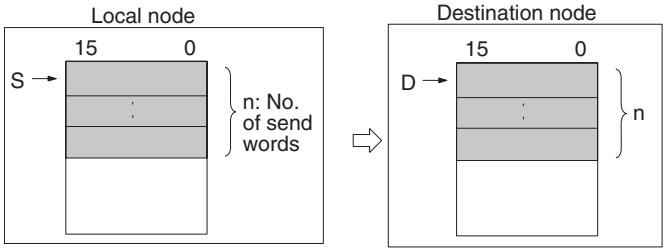
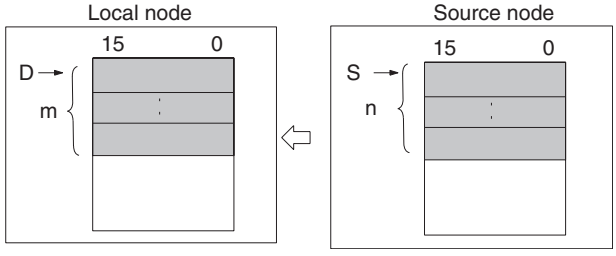
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
INTELLIGENT I/O READ IORD @IORD 222	IORD(222) C S D **C**: Control data **S**: Transfer source and number of words **D**: Transfer destination and number of words	Reads the contents of the memory area for the Special I/O Unit or CPU Bus Unit (see note).  **Note:** CS/CJ-series CPU Unit Ver. 2.0 or later (including CS1-H, CJ1-H, and CJ1M CPU Units from lot number 030418 or later) can read from CPU Bus Units.	Output Required	962
INTELLIGENT I/O WRITE IOWR @IOWR 223	IOWR(223) C S D **C**: Control data **S**: Transfer source and number of words **D**: Transfer destination and number of words	Outputs the contents of the CPU Unit's I/O memory area to the Special I/O Unit or the CPU Bus Unit (see note).  **Note:** CS/CJ-series CPU Unit Ver. 2.0 or later (including CS1-H, CJ1-H, and CJ1M CPU Units from lot number 030418 or later) can write to CPU Bus Units.	Output Required	967

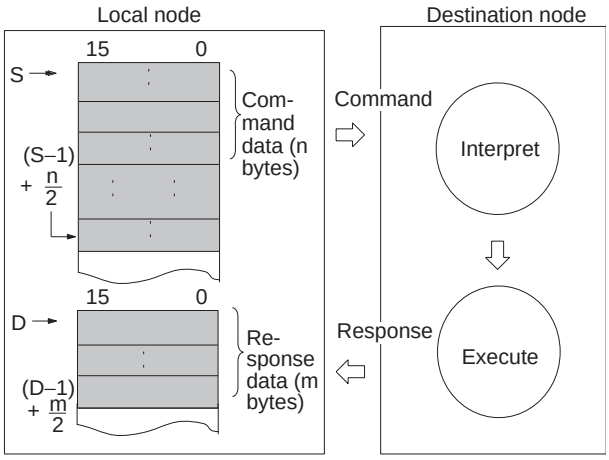
2-2-22 Serial Communications Instructions

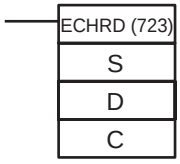
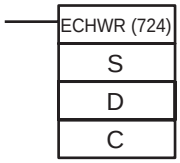
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
PROTOCOL MACRO PMCR @PMCR 260	PMCR(260) C1 C2 S R **C1**: Control word 1 **C2**: Control word 2 **S**: 1st send word **R**: 1st receive word	Calls and executes a communications sequence registered in a Serial Communications Board (CS Series only) or Serial Communications Unit. 	Output Required	974
TRANSMIT TXD @TXD 236	TXD(236) S C N **S**: 1st source word **C**: Control word **N**: Number of bytes 0000 to 0100 hex (0 to 256 decimal)	Outputs the specified number of bytes of data from the RS-232C port built into the CPU Unit or the serial port of a Serial Communications Board (version 1.2 or later).	Output Required	983
RECEIVE RXD @RXD 235	RXD(235) D C N **D**: 1st destination word **C**: Control word **N**: Number of bytes to store 0000 to 0100 hex (0 to 256 decimal)	Reads the specified number of bytes of data from the RS-232C port built into the CPU Unit or the serial port of a Serial Communications Board (version 1.2 or later).	Output Required	993
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT TXDU @TXDU 256	TXDU(256) S C N **S**: 1st source word **C**: 1st control word **N**: Number of bytes 0000 to 0256 BCD	Outputs the specified number of bytes of data from the serial port of a Serial Communications Unit (version 1.2 or later). The data is output in no-protocol mode with the start code and end code (if any) specified in the allocated DM Setup Area.	Output Required	1005

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
RECEIVE VIA SERIAL COMMUNICATIONS UNIT RXDU @RXDU 255	RXDU(255) D C N **D**: 1st destination word **C**: 1st control word **N**: Number of bytes to store 0000 to 0256 BCD	Reads the specified number of bytes of data from the serial port of a Serial Communications Unit (version 1.2 or later). The data is read in no-protocol mode with the start code and end code (if any) specified in the allocated DM Setup Area.	Output Required	1013
CHANGE SERIAL PORT SETUP STUP @STUP 237	STUP(237) C S **C**: Control word (port) **S**: First source word	Changes the communications parameters of a serial port on the CPU Unit, Serial Communications Unit (CPU Bus Unit), or Serial Communications Board. STUP(237) thus enables the protocol mode to be changed during PLC operation.	Output Required	1021

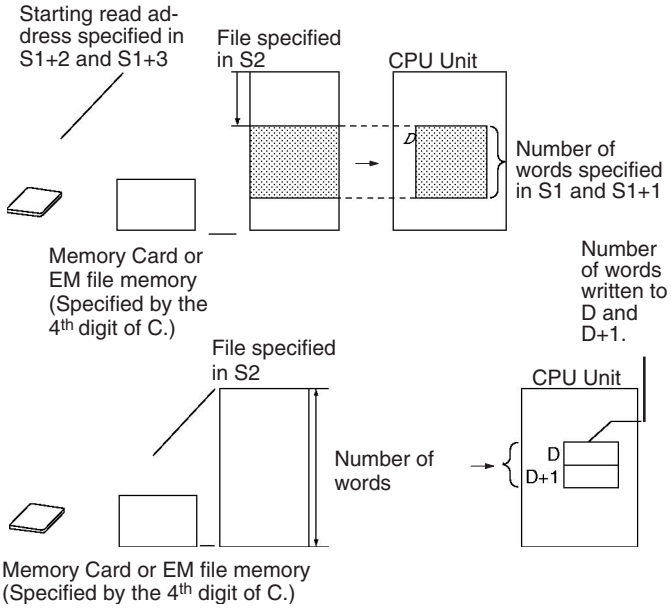
2-2-23 Network Instructions

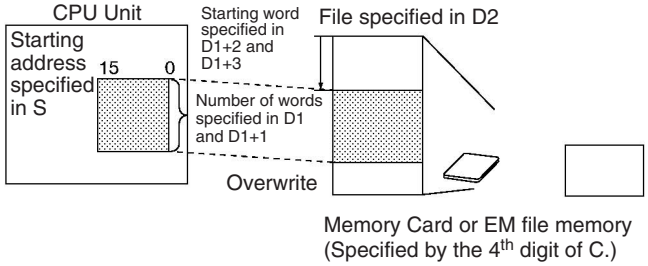
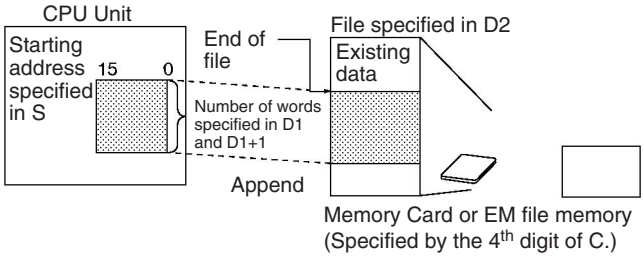
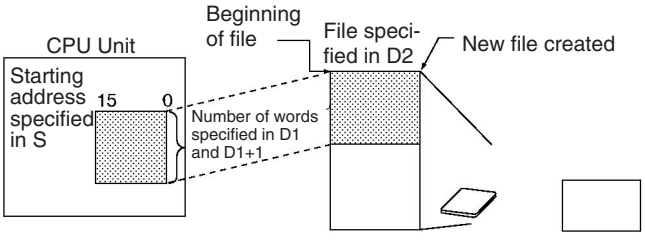
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
NETWORK SEND SEND @SEND 090	SEND(090) S D C **S**: 1st source word **D**: 1st destination word **C**: 1st control word	Transmits data to a node in the network. 	Output Required	1044
NETWORK RECEIVE RECV @RECV 098	RECV(098) S D C **S**: 1st source word **D**: 1st destination word **C**: 1st control word	Requests data to be transmitted from a node in the network and receives the data. 	Output Required	1050

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DELIVER COMMAND CMND @CMND 490	CMND(490) S D C **S**: 1st command word **D**: 1st response word **C**: 1st control word	Sends FINS commands and receives the response.  	Output Required	1056
EXPLICIT MESSAGE SEND EXPLT 720 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	EXPLT (720) S D C **S**: 1st word of send message **D**: 1st word of received message **C**: 1st control word	Sends an explicit message with any Service Code.	Output Required	1066
EXPLICIT GET ATTRIBUTE EGATR 721 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	EGATR (721) S D C **S**: 1st word of send message **D**: 1st word of received message **C**: 1st control word message	Reads status information with an explicit message (Get Attribute Single, Service Code: 0E hex).	Output Required	1074
EXPLICIT SET ATTRIBUTE ESATR 722 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	ESATR (722) S C **S**: First word of send message **C**: First control word	Writes status information with an explicit message (Set Attribute Single, Service Code: 0E hex)	Output Required	1081

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
EXPLICIT WORD READ ECHRD 723 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	 **S**: 1st source word in remote CPU Unit **D**: 1st destination word in local CPU Unit **C**: 1st control word	Reads data to the local CPU Unit from a remote CPU Unit in the network. (The remote CPU Unit must support explicit messages.)	Output Required	1087
EXPLICIT WORD WRITE ECHWR 724 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	 **S**: 1st source word in local CPU Unit **D**: 1st destination word in remote CPU Unit **C**: 1st control word	Writes data from the local CPU Unit to a remote CPU Unit in the network. (The remote CPU Unit must support explicit messages.)	Output Required	1091

2-2-24 File Memory Instructions

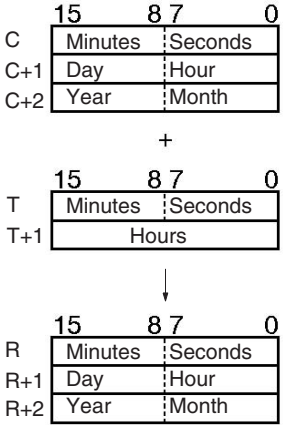
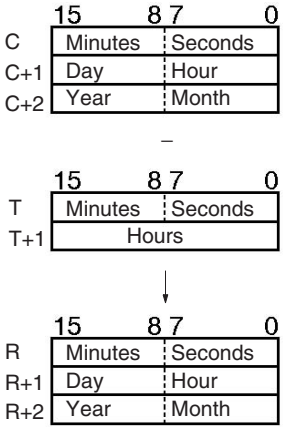
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
READ DATA FILE FREAD @FREAD 700	FREAD(700) C S1 S2 D **C**: Control word **S1**: 1st source word **S2**: Filename **D**: 1st destination word	Reads the specified data or amount of data from the specified data file in file memory to the specified data area in the CPU Unit. 	Output Required	1099

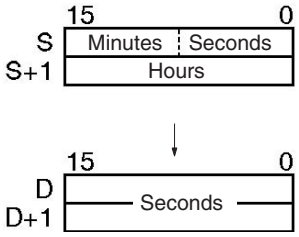
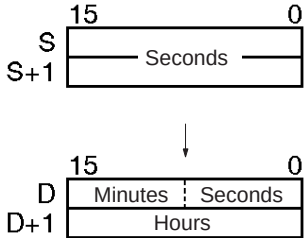
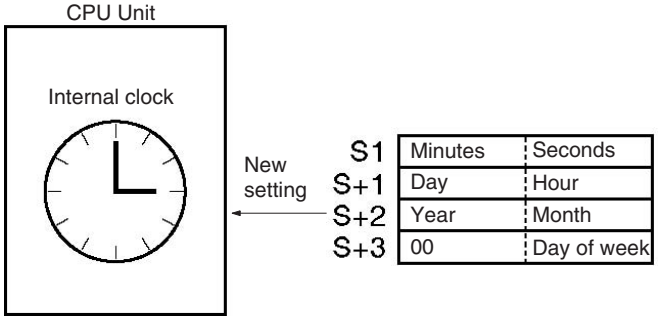
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
WRITE DATA FILE FWRIT @FWRIT 701	FWRIT(701) C D1 D2 S **C**: Control word **D1**: 1st destination word **D2**: Filename **S**: 1st source word	Overwrites or appends data in the specified data file in file memory with the specified data from the data area in the CPU Unit. If the specified file doesn't exist, a new file is created with that filename.   	Output Required	1106
WRITE TEXT FILE TWRIT @TWRIT 704 (CS/CJ-series CPU Units with unit version 4.0 or later only)	TWRIT C S1 S2 S3 S4 **C**: Control word **S1**: Number of bytes to write **S2**: Directory and file name **S3**: Write data **S4**: Delimiter	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). The data is stored in the TXT format.	Output Required	1113

2-2-25 Display Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DISPLAY MESSAGE MSG @MSG 046	MSG(046) N M **N**: Message number **M**: 1st message word	Reads the specified sixteen words of extended ASCII and displays the message on a Peripheral Device such as a Programming Console.	Output Required	1119

2-2-26 Clock Instructions

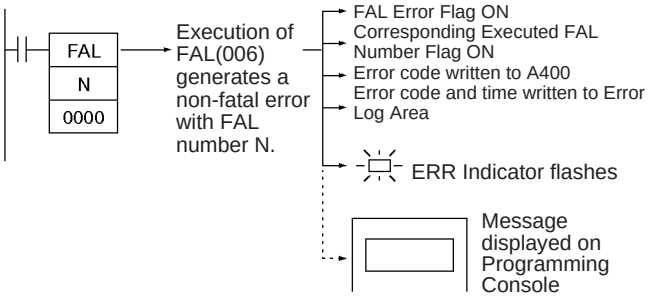
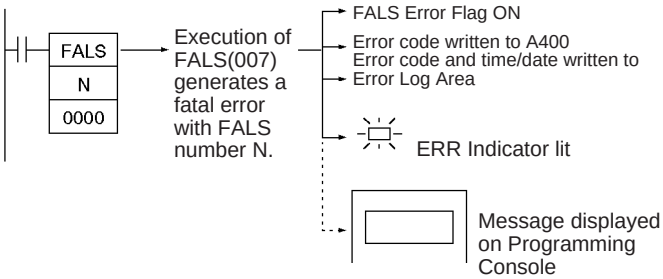
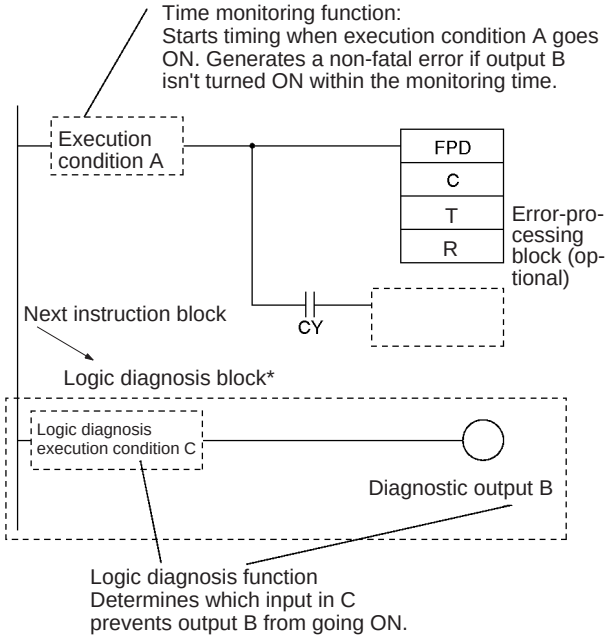
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
CALENDAR ADD CADD @CADD 730	CADD(730) C T R **C**: 1st calendar word **T**: 1st time word **R**: 1st result word	Adds time to the calendar data in the specified words. 	Output Required	1122
CALENDAR SUBTRACT CSUB @CSUB 731	CSUB(731) C T R **C**: 1st calendar word **T**: 1st time word **R**: 1st result word	Subtracts time from the calendar data in the specified words. 	Output Required	1126

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
HOURS TO SECONDS SEC @SEC 065	SEC(065) S D **S**: 1st source word **D**: 1st destination word	Converts time data in hours/minutes/seconds format to an equivalent time in seconds only. 	Output Required	1129
SECONDS TO HOURS HMS @HMS 066	HMS(066) S D **S**: 1st source word **D**: 1st destination word	Converts seconds data to an equivalent time in hours/minutes/seconds format. 	Output Required	1131
CLOCK ADJUSTMENT DATE @DATE 735	DATE(735) S **S**: 1st source word	Changes the internal clock setting to the setting in the specified source words. 	Output Required	1134

2-2-27 Debugging Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
TRACE MEMORY SAMPLING TRSM 045	TRSM(045)	When TRSM(045) is executed, the status of a preselected bit or word is sampled and stored in Trace Memory. TRSM(045) can be used anywhere in the program, any number of times.	Output Not required	1136

2-2-28 Failure Diagnosis Instructions

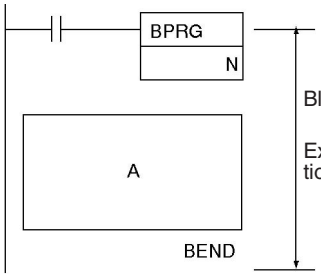
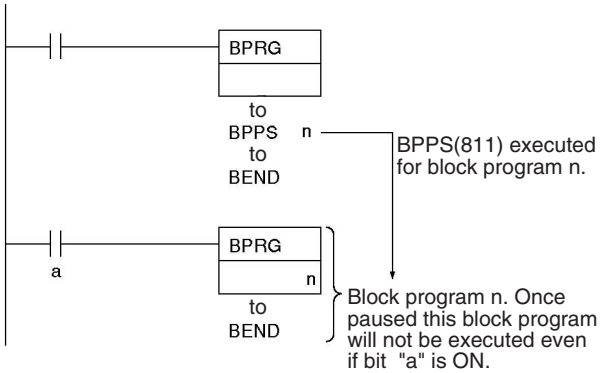
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FAILURE ALARM FAL @FAL 006	FAL(006) N S **N**: FAL number **S**: 1st message word or error code to generate	Generates or clears user-defined non-fatal errors. Non-fatal errors do not stop PC operation. Also generates non-fatal errors with the system. 	Output Required	1140
SEVERE FAILURE ALARM FALS 007	FALS(007) N S **N:** FALS number **S:** 1st message word or error code to generate	Generates user-defined fatal errors. Fatal errors stop PC operation. Also generates fatal errors with the system. 	Output Required	1148
FAILURE POINT DETECTION FPD 269	FPD(269) C T R **C**: Control word **T**: Monitoring time **R**: 1st register word	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. 	Output Required	1156

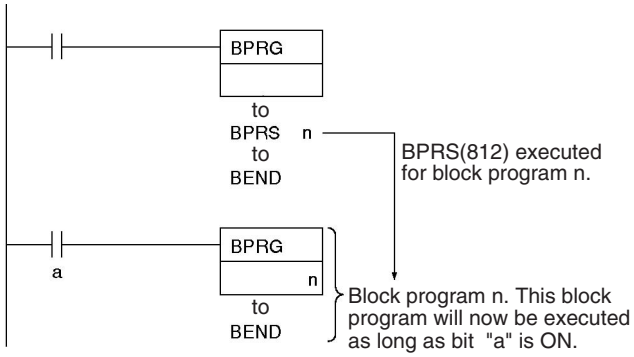
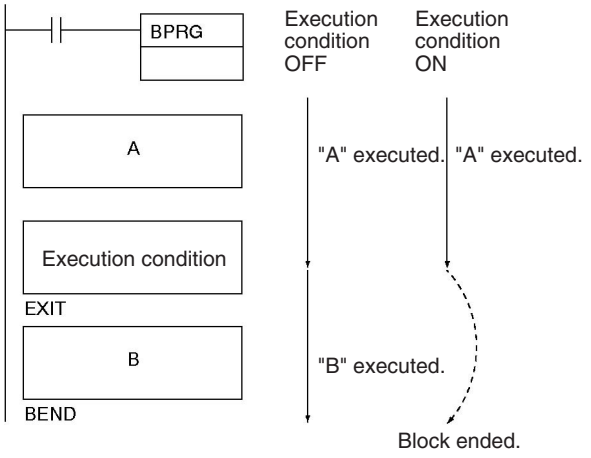
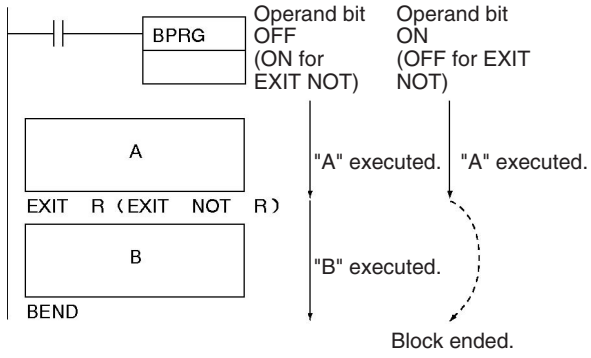
2-2-29 Other Instructions

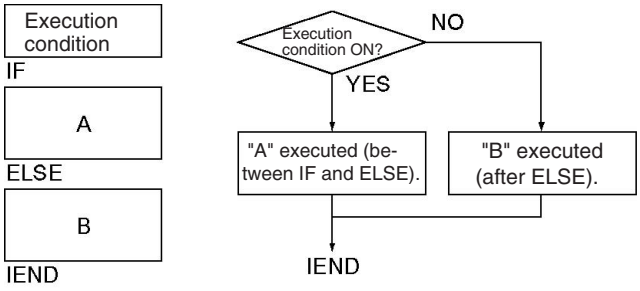
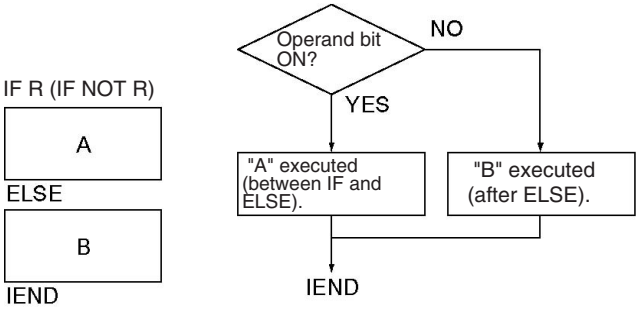
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SET CARRY STC @STC 040	STC(040)	Sets the Carry Flag (CY).	Output Required	1166
CLEAR CARRY CLC @CLC 041	CLC(041)	Turns OFF the Carry Flag (CY).	Output Required	1166
SELECT EM BANK EMBC @EMBC 281	EMBC(281) N **N**: EM bank number	Changes the current EM bank.	Output Required	1167
EXTEND MAXIMUM CYCLE TIME WDT @WDT 094	WDT(094) T **T**: Timer setting	Extends the maximum cycle time, but only for the cycle in which this instruction is executed.	Output Required	1169
SAVE CONDITION FLAGS (CS1-H, CJ1-H, CJ1M, or CS1D only) CCS @CCS 282	CCS(282)	Saves the status of the condition flags.	Output Required	1171
LOAD CONDITION FLAGS (CS1-H, CJ1-H, CJ1M, or CS1D only) CCL @CCL 283	CCL(283)	Reads the status of the condition flags that was saved.	Output Required	1173
CONVERT ADDRESS FROM CV (CS1-H, CJ1-H, CJ1M, or CS1D only) FRMCV @FRMCV 284	FRMCV(284) S D **S**: Word containing CV-series memory address **D**: Destination Index Register	Converts a CV-series PLC memory address to its equivalent CS/CJ-series PLC memory address.	Output Required	1174
CONVERT ADDRESS TO CV (CS1-H, CJ1-H, CJ1M, or CS1D only) TOCV @TOCV 285	TOCV(285) S D **S**: Index Register containing CS-series memory address **D**: Destination word	Converts a CS/CJ-series PLC memory address to its equivalent CV-series PLC memory address.	Output Required	1179

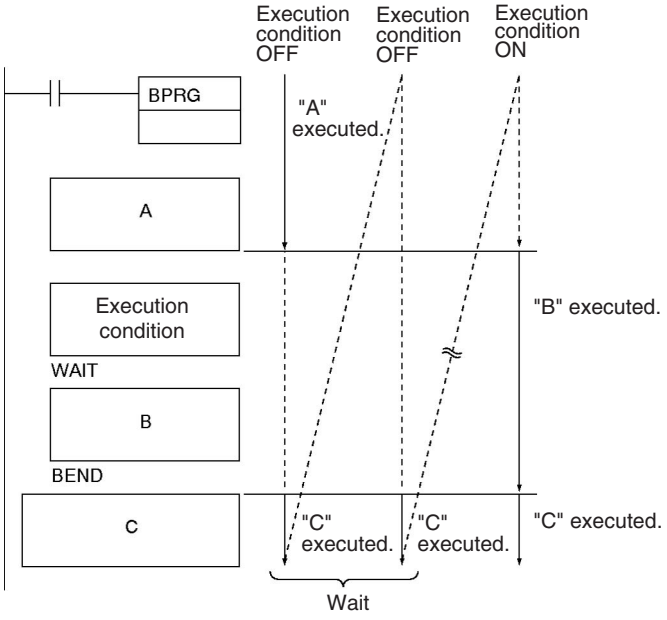
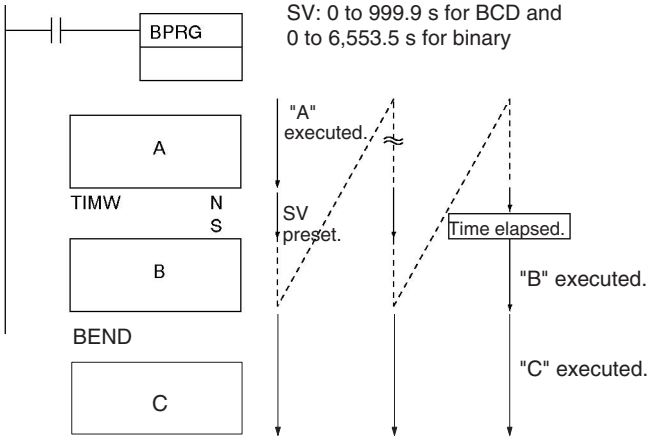
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DISABLE PERIPHERAL SERVICING (CS1D CPU Units for Single-CPU Systems, CS1-H, CJ1-H, or CJ1M only) IOSP @IOSP 287	IOSP(287)	Disables peripheral servicing during program execution in one of the Parallel Processing Modes or Peripheral Servicing Priority Mode.	Output Required	1183
ENABLE PERIPHERAL SERVICING (CS1D CPU Unit for Single-CPU Systems, CS1-H, CJ1-H, or CJ1M only) IORS 288	IORS(288)	Enables peripheral servicing that was disabled by IOSP(287) for program execution in one of the Parallel Processing Modes or Peripheral Servicing Priority Mode.	Output Not required	1185

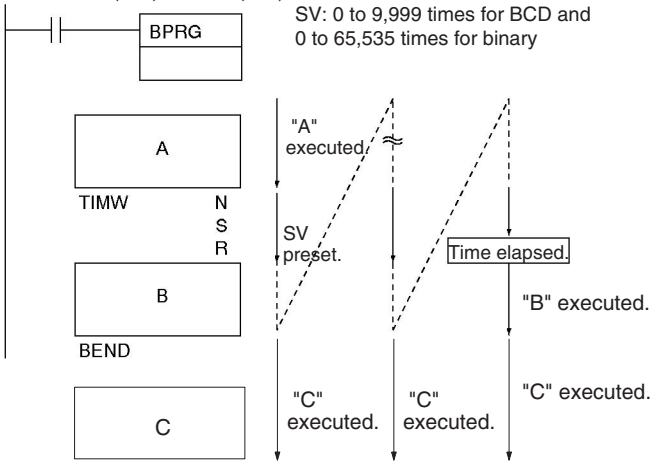
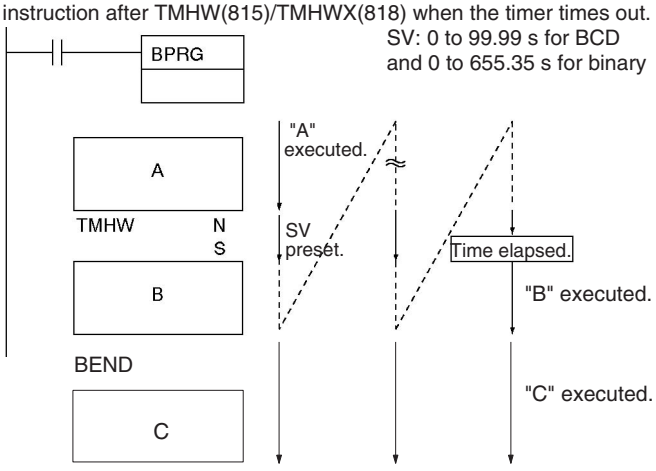
2-2-30 Block Programming Instructions

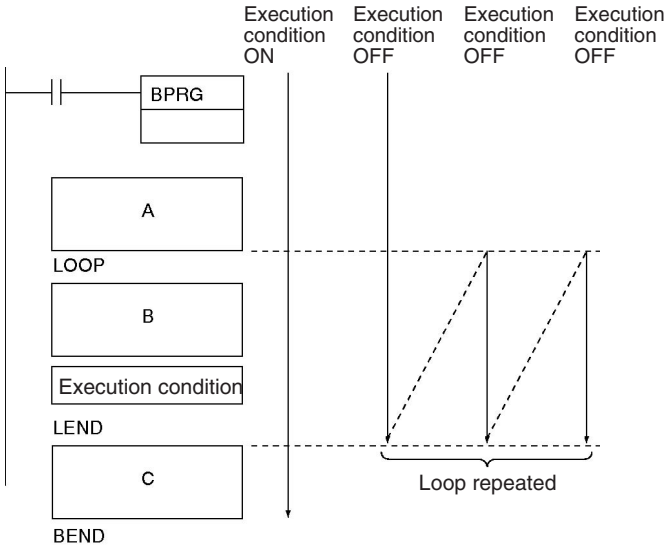
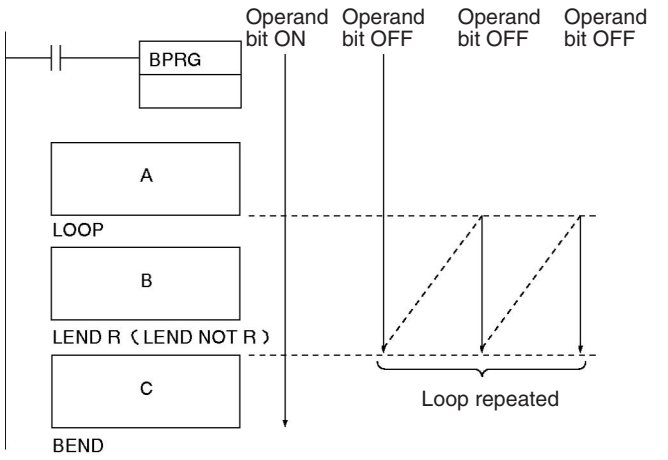
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BLOCK PROGRAM BEGIN BPRG 096	BPRG(096) N **N**: Block program number	Define a block programming area. For every BPRG(096) there must be a corresponding BEND(801). 	Output Required	1191
BLOCK PROGRAM END BEND 801		Define a block programming area. For every BPRG(096) there must be a corresponding BEND(801).	Block program Required	1191
BLOCK PROGRAM PAUSE BPPS 811	BPPS (811) N **N**: Block program number	Pause and restart the specified block program from another block program. 	Block program Required	1193

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BLOCK PROGRAM RESTART BPRS 812	BPRS (812) N **N**: Block program number	Pause and restart the specified block program from another block program. 	Block program Required	1193
CONDITIONAL BLOCK EXIT EXIT 806	EXIT(806) **B**: Bit operand	EXIT(806) without an operand bit exits the program if the execution condition is ON. 	Block program Required	1199
CONDITIONAL BLOCK EXIT EXIT 806	EXIT(806)B **B**: Bit operand	EXIT(806) without an operand bit exits the program if the execution condition is ON. 	Block program Required	1199
CONDITIONAL BLOCK EXIT NOT EXIT NOT 806	EXIT NOT(806) B **B**: Bit operand	EXIT(806) without an operand bit exits the program if the execution condition is OFF.	Block program Required	1199

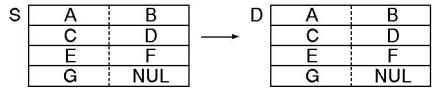
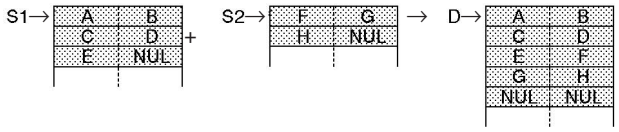
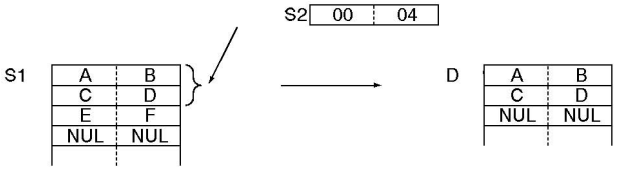
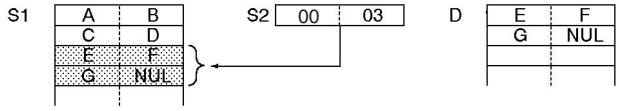
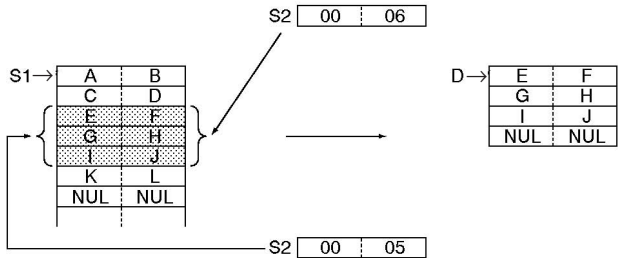
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
CONDITIONAL BLOCK BRANCHING IF 802	IF (802)	If the execution condition is ON, the instructions between IF(802) and ELSE(803) will be executed and if the execution condition is OFF, the instructions between ELSE(803) and IEND(804) will be executed. 	Block program Required	1196
CONDITIONAL BLOCK BRANCHING IF 802	IF (802) B **B**: Bit operand	If the operand bit is ON, the instructions between IF(802) and ELSE(803) will be executed. If the operand bit is OFF, the instructions between ELSE(803) and IEND(804) will be executed. 	Block program Required	1196
CONDITIONAL BLOCK BRANCHING (NOT) IF NOT 802	IF (802) NOT B **B**: Bit operand	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.	Block program Required	1196
CONDITIONAL BLOCK BRANCHING (ELSE) ELSE 803	---	If the ELSE(803) instruction is omitted and the operand bit is ON, the instructions between IF(802) and IEND(804) will be executed	Block program Required	1196
CONDITIONAL BLOCK BRANCHING END IEND 804	---	If the operand bit is OFF, only the instructions after IEND(804) will be executed.	Block program Required	1196

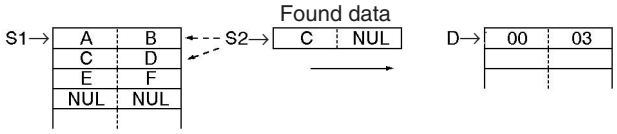
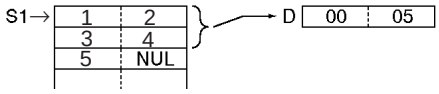
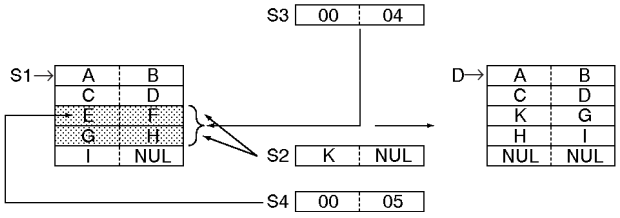
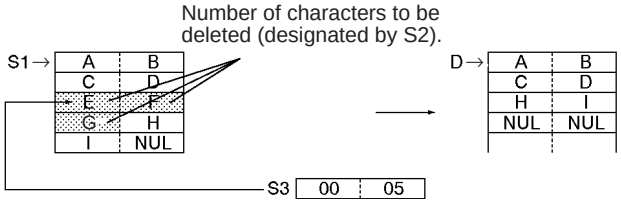
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ONE CYCLE AND WAIT WAIT 805	WAIT(805)	If the execution condition is ON for WAIT(805), the rest of the instruction in the block program will be skipped. 	Block program Required	1202
ONE CYCLE AND WAIT WAIT 805	WAIT(805) B **B**: Bit operand	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.	Block program Required	1202
ONE CYCLE AND WAIT (NOT) WAIT NOT 805	WAIT(805) NOT B **B**: Bit operand	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.	Block program Required	1202
HUNDRED-MS TIMER WAIT TIMW 813 (BCD) TIMWX 816 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	TIMW(813) N SV **N**: Timer number **SV**: Set value TIMWX(816) N SV **N**: Timer number **SV**: Set value	Delays execution of the block program until the specified time has elapsed. Execution continues from the next instruction after TIMW(813)/TIMWX(816) when the timer times out. SV: 0 to 999.9 s for BCD and 0 to 6,553.5 s for binary 	Block program Required	1206

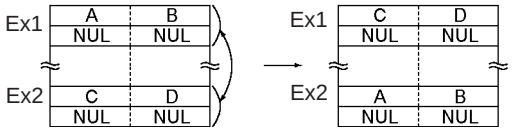
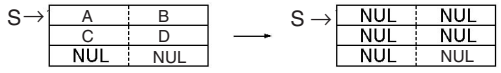
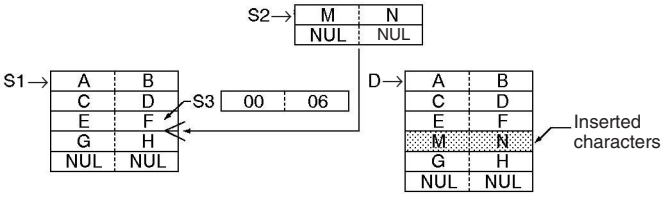
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
COUNTER WAIT CNTW 814 (BCD) CNTWX 818 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	CNTW(814) N SV **N**: Counter number **SV**: Set value **I**: Count input CNTWX(818) N SV **N**: Counter number **SV**: Set value **I**: Count input	Delays execution of the rest of the block program until the specified count has been achieved. Execution will be continued from the next instruction after CNTW(814)/CNTWX(818) when the counter counts out. SV: 0 to 9,999 times for BCD and 0 to 65,535 times for binary 	Block program Required	1209
TEN-MS TIMER WAIT TMHW 815 (BCD) TMHWX 817 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	TMHW(815) N SV **N**: Timer number **SV**: Set value TMHWX(817) N SV **N**: Timer number **SV**: Set value	Delays execution of the rest of the block program until the specified time has elapsed. Execution will be continued from the next instruction after TMHW(815)/TMHWX(818) when the timer times out. SV: 0 to 99.99 s for BCD and 0 to 655.35 s for binary 	Block program Required	1212

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
LOOP LOOP 809	---	LOOP(809) designates the beginning of the loop program. 	Block program Required	1215
LEND LEND 810	LEND (810)	LEND(810) or LEND(810) NOT specifies the end of the loop. When LEND(810) or LEND(810) NOT is reached, program execution will loop back to the next previous LOOP(809) until the operand bit for LEND(810) or LEND(810) NOT turns ON or OFF (respectively) or until the execution condition for LEND(810) turns ON.	Block program Required	1215
LEND LEND 810	LEND (810) B **B**: Bit operand	If the operand bit is OFF for LEND(810) (or ON for LEND(810) NOT), execution of the loop is repeated starting with the next instruction after LOOP(809). If the operand bit is ON for LEND(810) (or OFF for LEND(810) NOT), the loop is ended and execution continues to the next instruction after LEND(810) or LEND(810) NOT.  **Note** The status of the operand bit would be reversed for LEND(810) NOT.	Block program Required	1215
LEND NOT LEND NOT 810	LEND(810) NOT **B**: Bit operand	LEND(810) or LEND(810) NOT specifies the end of the loop. When LEND(810) or LEND(810) NOT is reached, program execution will loop back to the next previous LOOP(809) until the operand bit for LEND(810) or LEND(810) NOT turns ON or OFF (respectively) or until the execution condition for LEND(810) turns ON.	Block program Required	1215

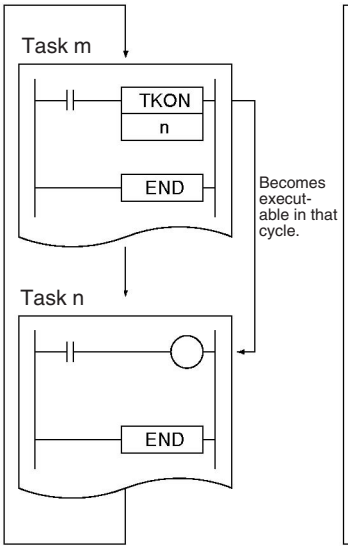
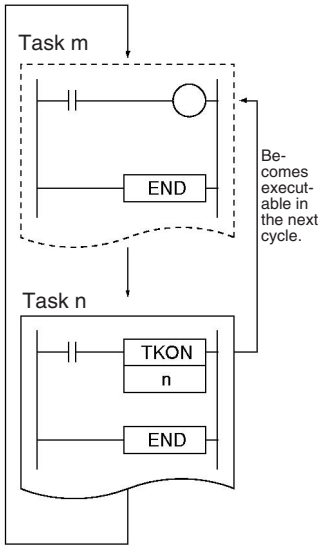
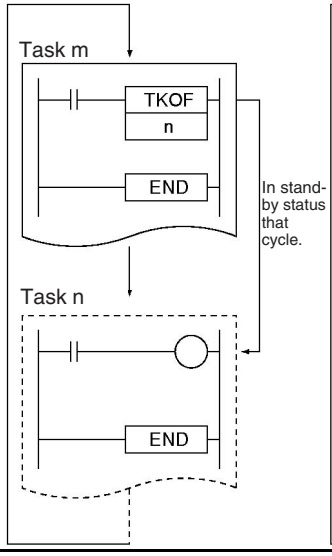
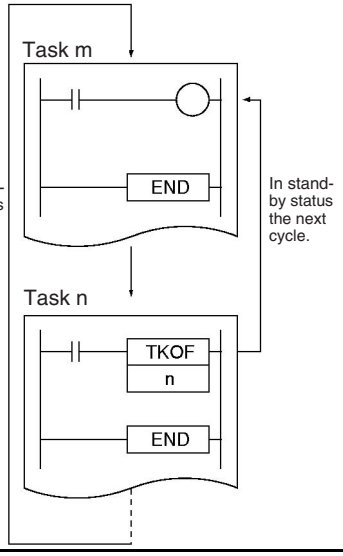
2-2-31 Text String Processing Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
MOV STRING MOV$ @MOV$ 664	MOV$(664) S D **S**: 1st source word **D**: 1st destination word	Transfers a text string. 	Output Required	1221
CONCATENATE STRING +$ @+$ 656	+$(656) S1 S2 D **S1**: Text string 1 **S2**: Text string 2 **D**: First destination word	Links one text string to another text string. 	Output Required	1223
GET STRING LEFT LEFT$ @LEFT$ 652	LEFT$(652) S1 S2 D **S1**: Text string first word **S2**: Number of characters **D**: First destination word	Fetches a designated number of characters from the left (beginning) of a text string. 	Output Required	1226
GET STRING RIGHT RGHT$ @RGHT$ 653	RGHT$(653) S1 S2 D **S1**: Text string first word **S2**: Number of characters **D**: First destination word	Reads a designated number of characters from the right (end) of a text string. 	Output Required	1228
GET STRING MIDDLE MID$ @MID$ 654	MID$(654) S1 S2 S3 D **S1**: Text string first word **S2**: Number of characters **S3**: Beginning position **D**: First destination word	Reads a designated number of characters from any position in the middle of a text string. 	Output Required	1230

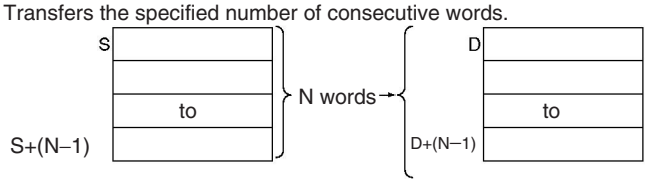
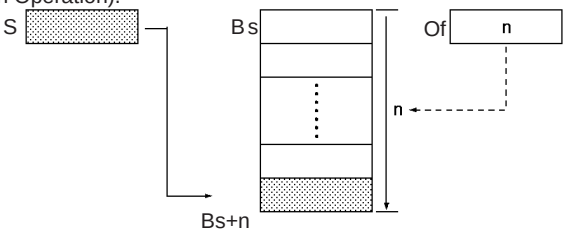
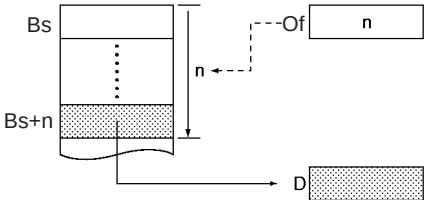
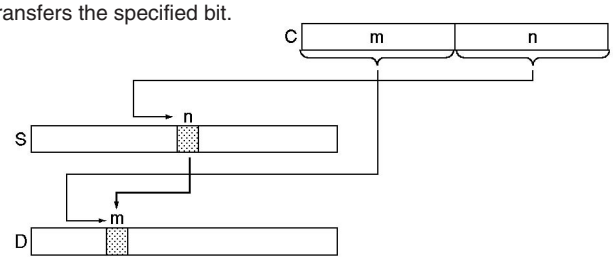
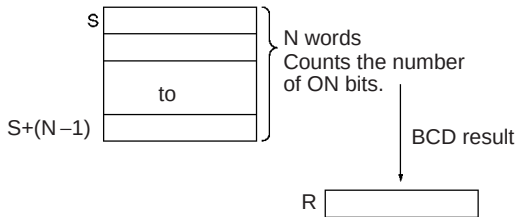
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FIND IN STRING FIND @FIND$ 660	FIND$(660) S1 S2 D **S1**: Source text string first word **S2**: Found text string first word **D**: First destination word	Finds a designated text string from within a text string. 	Output Required	1233
STRING LENGTH LEN$ @LEN$ 650	LEN$(650) S D **S**: Text string first word **D**: 1st destination word	Calculates the length of a text string. 	Output Required	1235
REPLACE IN STRING RPLC$ @RPLC$ 661	RPLC$(654) S1 S2 S3 S4 D **S1**: Text string first word **S2**: Replacement text string first word **S3**: Number of characters **S4**: Beginning position **D**: First destination word	Replaces a text string with a designated text string from a designated position. 	Output Required	1237
DELETE STRING DEL$ @DEL$ 658	DEL$(658) S1 S2 S3 D **S1**: Text string first word **S2**: Number of characters **S3**: Beginning position **D**: First destination word	Deletes a designated text string from the middle of a text string. 	Output Required	1240

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
EXCHANGE STRING XCHG$ @XCHG$ 665	XCHG$(665) Ex1 Ex2 **Ex1**: 1st exchange word 1 **Ex2**: 1st exchange word 2	Replaces a designated text string with another designated text string. 	Output Required	1242
CLEAR STRING CLR$ @CLR$ 666	CLR$(666) S **S**: Text string first word	Clears an entire text string with NUL (00 hex). 	Output Required	1245
INSERT INTO STRING INS$ @INS$ 657	INS$(657) S1 S2 S3 D **S1**: Base text string first word **S2**: Inserted text string first word **S3**: Beginning position **D**: First destination word	Deletes a designated text string from the middle of a text string. 	Output Required	1246
String Comparison LD, AND, OR + =$, <>$, <$, <=$, >$, >=$ 670 (=$) 671 (<>$) 672 (<$) 673 (<=$) 674 (>$) 675 (>=$)	**LD** Symbol S1 S2 **AND** Symbol S1 S2 **OR** Symbol S1 S2 **S1**: Text string 1 **S2**: Text string 2	Sting comparison instructions (=$, <>$, <$, <=$, >$, >=$) compare two text strings from the beginning, in terms of value of the ASCII codes. If the result of the comparison is true, an ON execution condition is created for a LOAD, AND, or OR.	LD: Not required AND, OR: Required	1250

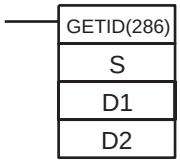
2-2-32 Task Control Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
TASK ON TKON @TKON 820	TKON(820) N **N**: Task number	Makes the specified task executable. The specified task's task number is higher than the local task's task number (m<n).  The specified task's task number is lower than the local task's task number (m>n). 	Output Required	1255
TASK OFF TKOF @TKOF 821	TKOF(821) N **N**: Task number	Puts the specified task into standby status. The specified task's task number is higher than the local task's task number (m<n).  The specified task's task number is lower than the local task's task number (m>n). 	Output Required	1258

2-2-33 Model Conversion Instructions (CPU Unit Ver. 3.0 or Later Only)

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BLOCK TRANSFER XFERC @XFERC 565	XFERC(565) N S D **N**: Number of words **S**: 1st source word **D**: 1st destination word	Transfers the specified number of consecutive words. 	Output Required	1263
SINGLE WORD DISTRIBUTE DISTC @DISTC 566	DISTC(566) S Bs Of **S**: Source word **Bs**: Destination base address **Of**: Offset	Transfers the source word to a destination word calculated by adding an offset value to the base address. Can also write to a stack (Stack Push Operation). 	Output Required	1266
DATA COLLECT COLLC @COLLC 567	COLLC(567) Bs Of D **Bs**: Source base address **Of**: Offset **D**: Destination word	Transfers the source word (calculated by adding an offset value to the base address) to the destination word. Can also read data from a stack in FIFO or LIFO order (Stack Read Operation). 	Output Required	1269
MOVE BIT MOVBC @MOVBC 568	MOVBC(568) S C D **S**: Source word or data **C**: Control word **D**: Destination word	Transfers the specified bit. 	Output Required	1273
BIT COUNTER BCNTC @BCNTC 621	BCNTC(621) N S R **N**: Number of words (BCD) **S**: 1st source word **R**: Result word	Counts the total number of ON bits in the specified word(s). 	Output Required	1275

2-2-34 Special Function Block Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
GET VARIABLE ID GETID @GETID 286	 **S**: Variable or address **D1**: ID code **D2**: Destination word	Outputs the FINS command variable type (data area) code and word address for the specified variable or address. This instruction is generally used to get the assigned address of a variable in a function block.	Output Required	1277

2-3 Alphabetical List of Instructions by Mnemonic

A

Mnemonic	Instruction	Function code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
ACC	ACCELERATION CONTROL	888	@ACC	---	---	896
ACOS	ARC COSINE	464	@ACOS	---	---	625
ACOSD	DOUBLE ARC COSINE	855	@ACOSD	---	---	682
AND	AND	---	@AND	%AND	!AND	165
AND <	AND LESS THAN	310	---	---	---	291
AND <$	AND STRING LESS THAN	672	---	---	---	1250
AND <>	AND NOT EQUAL	305	---	---	---	291
AND <>$	AND STRING NOT EQUAL	671	---	---	---	1250
AND <>D	AND DOUBLE FLOATING NOT EQUAL	336	---	---	---	694
AND <> DT	AND TIME NOT EQUAL	342	---	---	---	297
AND <>F	AND FLOATING NOT EQUAL	330	---	---	---	636
AND <>L	AND DOUBLE NOT EQUAL	306	---	---	---	291
AND <>S	AND SIGNED NOT EQUAL	307	---	---	---	291
AND <>SL	AND DOUBLE SIGNED NOT EQUAL	308	---	---	---	291
AND <D	AND DOUBLE FLOATING LESS THAN	337	---	---	---	694
AND <DT	AND TIME LESS THAN	343	---	---	---	297
AND <F	AND FLOATING LESS THAN	331	---	---	---	636
AND <L	AND DOUBLE LESS THAN	311	---	---	---	291
AND <S	AND SIGNED LESS THAN	312	---	---	---	291
AND <SL	AND DOUBLE SIGNED LESS THAN	313	---	---	---	291
AND =	AND EQUAL	300	---	---	---	291
AND =$	AND STRING EQUALS	670	---	---	---	1250
AND =D	AND DOUBLE FLOATING EQUAL	335	---	---	---	694
AND =DT	AND TIME EQUAL	341	---	---	---	297
AND =F	AND FLOATING EQUAL	329	---	---	---	636
AND =L	AND DOUBLE EQUAL	301	---	---	---	291
AND =S	AND SIGNED EQUAL	302	---	---	---	291
AND =SL	AND DOUBLE SIGNED EQUAL	303	---	---	---	291
AND >	AND GREATER THAN	320	---	---	---	291
AND >$	AND STRING GREATER THAN	674	---	---	---	1250
AND >D	AND DOUBLE FLOATING GREATER THAN	339	---	---	---	694
AND >DT	AND TIME GREATER THAN	345	---	---	---	297
AND >F	AND FLOATING GREATER THAN	333	---	---	---	636

Mnemonic	Instruction	Function code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
AND >L	AND DOUBLE GREATER THAN	321	---	---	---	291
AND >S	AND SIGNED GREATER THAN	322	---	---	---	291
AND >SL	AND DOUBLE SIGNED GREATER THAN	323	---	---	---	291
AND LD	AND LOAD	---	---	---	---	172
AND NOT	AND NOT	---	---	---	!AND NOT	167
AND TST	AND BIT TEST	350	---	---	---	182
AND TSTN	AND BIT TEST	351	---	---	---	182
AND <=	AND LESS THAN OR EQUAL	315	---	---	---	291
AND <=$	AND STRING LESS THAN OR EQUAL	673	---	---	---	1250
AND <=D	AND DOUBLE FLOATING LESS THAN OR EQUAL	338	---	---	---	694
AND <=DT	AND TIME LESS THAN OR EQUAL	344	---	---	---	297
AND <=F	AND FLOATING LESS THAN OR EQUAL	332	---	---	---	636
AND <=L	AND DOUBLE LESS THAN OR EQUAL	316	---	---	---	291
AND <=S	AND SIGNED LESS THAN OR EQUAL	317	---	---	---	291
AND <=SL	AND DOUBLE SIGNED LESS THAN OR EQUAL	318	---	---	---	291
AND >=	AND GREATER THAN OR EQUAL	325	---	---	---	291
AND >=$	AND STRING GREATER THAN OR EQUALS	675	---	---	---	1250
AND >=D	AND DOUBLE FLOATING GREATER THAN OR EQUAL	340	---	---	---	694
AND >=DT	AND TIME GREATER THAN OR EQUAL	346	---	---	---	297
AND >=F	AND FLOATING GREATER THAN OR EQUAL	334	---	---	---	636
AND >=L	AND DOUBLE GREATER THAN OR EQUAL	326	---	---	---	291
AND >=S	AND SIGNED GREATER THAN OR EQUAL	327	---	---	---	291
AND >=SL	AND DOUBLE SIGNED GREATER THAN OR EQUAL	328	---	---	---	291
ANDL	DOUBLE LOGICAL AND	610	@ANDL	---	---	550
ANDW	LOGICAL AND	034	@ANDW	---	---	548
APR	ARITHMETIC PROCESS	069	@APR	---	---	571
ASC	ASCII CONVERT	086	@ASC	---	---	504
ASFT	ASYNCHRONOUS SHIFT REGISTER	017	@ASFT	---	---	365
ASIN	ARC SINE	463	@ASIN	---	---	623
ASIND	DOUBLE ARC SINE	854	@ASIND	---	---	680
ASL	ARITHMETIC SHIFT LEFT	025	@ASL	---	---	370

Mnemonic	Instruction	Function code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
ASLL	DOUBLE SHIFT LEFT	570	@ASLL	---	---	371
ASR	ARITHMETIC SHIFT RIGHT	026	@ASR	---	---	373
ASRL	DOUBLE SHIFT RIGHT	571	@ASRL	---	---	374
ATAN	ARC TANGENT	465	@ATAN	---	---	627
ATAND	DOUBLE ARC TANGENT	856	@ATAND	---	---	684
AVG	AVERAGE	195	---	---	---	807

B

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
BAND	DEAD BAND CONTROL	681	@BAND	---	---	781
BCD	BINARY TO BCD	024	@BCD	---	---	487
BCDL	DOUBLE BINARY TO BCD	059	@BCDL	---	---	489
BCDS	SIGNED BINARY TO BCD	471	@BCDS	---	---	523
BCMP	UNSIGNED BLOCK COMPARE	068	@BCMP	---	---	320
BCMP2	EXPANDED BLOCK COMPARE	502	@BCMP2	---	---	322
BCNT	BIT COUNTER	067	@BCNT	---	---	587
BCNTC	BIT COUNTER	621	@BCNTC	---	---	1275
BDSL	DOUBLE SIGNED BINARY TO BCD	473	@BDSL	---	---	525
BEND	BLOCK PROGRAM END	801	---	---	---	1191
BIN	BCD TO BINARY	023	@BIN	---	---	483
BINL	DOUBLE BCD TO DOUBLE BINARY	058	@BINL	---	---	485
BINS	SIGNED BCD TO BINARY	470	@BINS	---	---	517
BISL	DOUBLE SIGNED BCD TO BINARY	472	@BISL	---	---	520
BPPS	BLOCK PROGRAM PAUSE	811	---	---	---	1193
BPRG	BLOCK PROGRAM BEGIN	096	---	---	---	1191
BPRS	BLOCK PROGRAM RESTART	812	---	---	---	1193
BREAK	BREAK LOOP	514	---	---	---	241
BSET	BLOCK SET	071	@BSET	---	---	347

C

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
CADD	CALENDAR ADD	730	@CADD	---	---	1122
CCL	LOAD CONDITION FLAGS	283	@CCL	---	---	1173
CCS	SAVE CONDITION FLAGS	282	@CCS	---	---	1171
CJP	CONDITIONAL JUMP	510	---	---	---	232
CJPN	CONDITIONAL JUMP	511	---	---	---	232
CLC	CLEAR CARRY	041	@CLC	---	---	1166

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
CLI	CLEAR INTERRUPT	691	@CLI	---	---	851
CLR$	CLEAR STRING	666	@CLR$	---	---	1245
CMND	DELIVER COMMAND	490	@CMND	---	---	1056
CMP	COMPARE	020	---	---	!CMP	303
CMPL	DOUBLE COMPARE	060	---	---	---	306
CNR	RESET TIMER/ COUNTER	545	@CNR	---	---	282
CNRX	RESET TIMER/ COUNTER	548	@CNRX	---	---	282
CNT	COUNTER	---	---	---	---	275
CNTX	COUNTER	546	---	---	---	275
CNTR	REVERSIBLE COUNTER	012	---	---	---	278
CNTRX	REVERSIBLE COUNTER	548	---	---	---	278
CNTW	COUNTER WAIT	814	---	---	---	1209
CNTWX	COUNTER WAIT	818	---	---	---	1209
COLL	DATA COLLECT	081	@COLL	---	---	354
COLLC	DATA COLLECT	567	@COLLC	---	---	1269
COLM	LINE TO COLUMN	064	@COLM	---	---	514
COM	COMPLEMENT	029	---	---	---	562
COML	DOUBLE COMPLEMENT	614	@COML	---	---	564
COS	COSINE	461	@COS	---	---	615
COSD	DOUBLE COSINE	852	@COSD	---	---	676
COSQ	HIGH-SPEED COSINE	476	@COSQ	---	---	617
CPS	SIGNED BINARY COMPARE	114	---	---	!CPS	309
CPSL	DOUBLE SIGNED BINARY COMPARE	115	---	---	---	312
CSUB	CALENDAR SUBTRACT	731	@CSUB	---	---	1126
CTBL	COMPARISON TABLE LOAD	882	@CTBL	---	---	878

D

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
DATE	CLOCK ADJUSTMENT	735	@DATE	---	---	1134
DBL	16-BIT BINARY TO DOUBLE FLOATING	843	@DBL	---	---	660
DBLL	32-BIT BINARY TO DOUBLE FLOATING	844	@DBLL	---	---	661
DEG	RADIANS-TO DEGREES	459	@DEG	---	---	610
DEGD	DOUBLE RADIANS TO DEGREES	850	@RADD	---	---	671
DEL$	DELETE STRING	658	@DEL$	---	---	1240
DI	DISABLE INTER-RUPTS	693	@DI	---	---	855
DIFD	DIFFERENTIATE DOWN	014	---	---	!DIFD	193
DIFU	DIFFERENTIATE UP	013	---	---	!DIFU	193
DIM	DIMENSION RECORD TABLE	631	@DIM	---	---	715
DIST	SINGLE WORD DISTRIBUTE	080	@DIST	---	---	352

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
DISTC	SINGLE WORD DISTRIBUTE	566	@DISTC	---	---	1266
DLNK	CPU BUS UNIT I/O REFRESH	226	@DLNK	---	---	932
DMPX	DATA ENCODER	077	@DMPX	---	---	500
DOWN	CONDITION OFF	522	---	---	---	181
DSW	DIGITAL SWITCH INPUT	210	---	---	---	940

E

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
ECHRD	EXPLICIT WORD READ	723	@ECHRD	---	---	1087
ECHWR	EXPLICIT WORD WRITE	724	@ECHWR	---	---	1091
EGATR	EXPLICIT GET ATTRIBUTE	721	@EGATR	---	---	1074
EI	ENABLE INTERRUPTS	694	---	---	---	858
ELSE	ELSE	803	---	---	---	1196
EMBC	SELECT EM BANK	281	@EMBC	---	---	1167
END	END	001	---	---	---	206
ESATR	EXPLICIT SET ATTRIBUTE	722	@ESATR	---	---	1081
EXIT NOT (operand)	CONDITIONAL BLOCK EXIT NOT	806	---	---	---	1199
EXIT (input condition)	CONDITIONAL BLOCK EXIT	806	---	---	---	1199
EXIT (operand)	CONDITIONAL BLOCK EXIT	806	---	---	---	1199
EXP	EXPONENT	467	@EXP	---	---	631
EXPD	DOUBLE EXPONENT	858	@EXPD	---	---	688
EXPLT	EXPLICIT MESSAGE SEND	720	@EXPLT	---	---	1066

F

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
FAL	FAILURE ALARM	006	@FAL	---	---	1140
FALS	SEVERE FAILURE ALARM	007	---	---	---	1148
FCS	FRAME CHECKSUM	180	@FCS	---	---	738
FDIV	FLOATING POINT DIVIDE	079	@FDIV	---	---	583
FIFO	FIRST IN FIRST OUT	633	@FIFO	---	---	709
FIND$	FIND IN STRING	660	@FIND$	---	---	1233
FIORF	SPECIAL I/O UNIT I/O REFRESH	225	@FIORF	---	---	929
FIX	FLOATING TO 16-BIT	450	@FIX	---	---	594
FIXD	DOUBLE FLOATING TO 16-BIT BINARY	841	@FIXD	---	---	657
FIXL	FLOATING TO 32-BIT	451	@FIXL	---	---	596
FIXLD	DOUBLE FLOATING TO 32-BIT BINARY	842	@FIXLD	---	---	658
FLT	16-BIT TO FLOATING	452	@FLT	---	---	597
FLTL	32-BIT TO FLOATING	453	@FLTL	---	---	599

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
FOR	FOR-NEXT LOOPS	512	---	---	---	238
FPD	FAILURE POINT DETECTION	269	---	---	---	1156
FREAD	READ DATA FILE	700	@FREAD	---	---	1099
FRMCV	CONVERT ADDRESS FROM CV	284	@FRMCV	---	---	1174
FSTR	FLOATING POINT TO ASCII	448	@FSTR	---	---	640
FWRIT	WRITE DATA FILE	701	@FWRIT	---	---	1106
FVAL	ASCII TO FLOATING POINT	449	@FVAL	---	---	645

G

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
GETID	GET VARIABLE ID	286	@GETID	---	---	1277
GETR	GET RECORD NUMBER	636	@GETR	---	---	720
GRET	GLOBAL SUBROUTINE RETURN	752	---	---	---	835
GRY	GRAY CODE CONVERSION	474	@GRY	---	---	529
GSBN	GLOBAL SUBROUTINE ENTRY	751	---	---	---	832
GSBS	GLOBAL SUBROUTINE CALL	750	@GSBS	---	---	824

H

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
HEX	ASCII TO HEX	162	@HEX	---	---	508
HKY	HEXADECIMAL KEY INPUT	212	---	---	---	948
HMS	SECONDS TO HOURS	066	@HMS	---	---	1131

I

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
IEND	IF END	804	---	---	---	1196
IF NOT (operand)	IF NOT	802	---	---	---	1196
IF (input condition)	IF	802	---	---	---	1196
IF (operand)	IF	802	---	---	---	1196
IL	INTERLOCK	002	---	---	---	210
ILC	INTERLOCK CLEAR	003	---	---	---	210
INI	MODE CONTROL	880	@INI	---	---	864
INS$	INS$	657	@INS$	---	---	1246
IORD	INTELLIGENT I/O READ	222	@IORD	---	---	962
IORF	I/O REFRESH	097	@IORF	---	---	926
IORS	ENABLE PERIPHERAL SERVICING	288	---	---	---	1185

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
IOSP	DISABLE PERIPHERAL SERVICING	287	@IOSP	---	---	1183
IOWR	INTELLIGENT I/O WRITE	223	@IOWR	---	---	967

J

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
JME	JUMP END	005	---	---	---	228
JME0	MULTIPLE JUMP END	516	---	---	---	236
JMP	JUMP	004	---	---	---	228
JMP0	MULTIPLE JUMP	515	---	---	---	236

K

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
KEEP	KEEP	011	---	---	!KEEP	188

L

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
LD	LOAD	---	@LD	%LD	!LD	161
LD <	LOAD LESS THAN	310	---	---	---	291
LD <$	LOAD STRING LESS THAN	672	---	---	---	1250
LD <D	LOAD DOUBLE FLOATING LESS THAN	337	---	---	---	694
LD <DT	LOAD TIME LESS THAN	343	---	---	---	297
LD <F	LOAD FLOATING LESS THAN	331	---	---	---	636
LD <>	LOAD NOT EQUAL	305	---	---	---	291
LD <>$	LOAD STRING NOT EQUAL	671	---	---	---	1250
LD <>D	LOAD DOUBLE FLOATING NOT EQUAL	336	---	---	---	694
LD <>DT	LOAD TIME NOT EQUAL	342	---	---	---	297
LD <>F	LOAD FLOATING NOT EQUAL	330	---	---	---	636
LD <>L	LOAD DOUBLE NOT EQUAL	306	---	---	---	291
LD <>S	LOAD SIGNED NOT EQUAL	307	---	---	---	291
LD <>SL	LOAD DOUBLE SIGNED NOT EQUAL	308	---	---	---	291
LD <L	LOAD DOUBLE LESS THAN	311	---	---	---	291
LD <S	LOAD SIGNED LESS THAN	312	---	---	---	291
LD <SL	LOAD DOUBLE SIGNED LESS THAN	313	---	---	---	291
LD =	LOAD EQUAL	300	---	---	---	291
LD =$	LOAD STRING EQUALS	670	---	---	---	1250

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
LD =D	LOAD DOUBLE FLOATING EQUAL	335	---	---	---	694
LD =DT	LOAD TIME EQUAL	341	---	---	---	297
LD =F	LOAD FLOATING EQUAL	329	---	---	---	636
LD =L	LOAD DOUBLE EQUAL	301	---	---	---	291
LD =S	LOAD SIGNED EQUAL	302	---	---	---	291
LD =SL	LOAD DOUBLE SIGNED EQUAL	303	---	---	---	291
LD >	LOAD GREATER THAN	320	---	---	---	291
LD >$	LOAD STRING GREATER THAN	674	---	---	---	1250
LD >D	LOAD DOUBLE FLOATING GREATER THAN	339	---	---	---	694
LD >DT	LOAD TIME GREATER THAN	345	---	---	---	297
LD >F	LOAD FLOATING GREATER THAN	333	---	---	---	636
LD >L	LOAD DOUBLE GREATER THAN	321	---	---	---	291
LD >S	LOAD SIGNED GREATER THAN	322	---	---	---	291
LD >SL	LOAD DOUBLE SIGNED GREATER THAN	323	---	---	---	291
LD NOT	LOAD NOT	---	---	---	!LD NOT	163
LD TST	LOAD BIT TEST	350	---	---	---	182
LD TSTN	LOAD BIT TEST	351	---	---	---	182
LD <=	LOAD LESS THAN OR EQUAL	315	---	---	---	291
LD <=$	LOAD STRING LESS THAN OR EQUAL	673	---	---	---	1250
LD <=D	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	338	---	---	---	694
LD <=DT	LOAD TIME LESS THAN OR EQUAL	344	---	---	---	297
LD <=F	LOAD FLOATING LESS THAN OR EQUAL	332	---	---	---	636
LD <=L	LOAD DOUBLE LESS THAN OR EQUAL	316	---	---	---	291
LD <=S	LOAD SIGNED LESS THAN OR EQUAL	317	---	---	---	291
LD <=SL	LOAD DOUBLE SIGNED LESS THAN OR EQUAL	318	---	---	---	291
LD >=	LOAD GREATER THAN OR EQUAL	325	---	---	---	291
LD >=$	LOAD STRING GREATER THAN OR EQUALS	675	---	---	---	1250
LD >=D	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	340	---	---	---	694
LD >=DT	LOAD TIME GREATER THAN OR EQUAL	346	---	---	---	297
LD >=F	LOAD FLOATING GREATER THAN OR EQUAL	334	---	---	---	636

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
LD >=L	LOAD DOUBLE GREATER THAN OR EQUAL	326	---	---	---	291
LD >=S	LOAD SIGNED GREATER THAN OR EQUAL	327	---	---	---	291
LD >=SL	LOAD DOUBLE SIGNED GREATER THAN OR EQUAL	328	---	---	---	291
LEFT$	GET STRING LEFT	652	@LEFT$	---	---	1226
LEN$	STRING LENGTH	650	@LEN$	---	---	1235
LEND NOT (operand)	LOOP END NOT	810	---	---	---	1215
LEND (input condition)	LOOP END	810	---	---	---	1215
LEND (operand)	LOOP END	810	---	---	---	1215
LIFO	LAST IN FIRST OUT	634	@LIFO	---	---	712
LINE	COLUMN TO LINE	063	@LINE	---	---	512
LMT	LIMIT CONTROL	680	@LMT	---	---	779
LOG	LOGARITHM	468	@LOG	---	---	633
LOGD	DOUBLE LOGARITHM	859	@LOGD	---	---	690
LOOP	LOOP	809	---	---	---	1215

M

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
MAX	FIND MAXIMUM	182	@MAX	---	---	727
MCMP	MULTIPLE COMPARE	019	@MCMP	---	---	315
MCRO	MACRO	099	@MCRO	---	---	817
MID$	GET STRING MIDDLE	654	@MID$	---	---	1230
MILC	MULTI-INTERLOCK CLEAR	519	---	---	---	214
MILH	MULTI-INTERLOCK DIFFERENTIATION HOLD	517	---	---	---	214
MILR	MULTI-INTERLOCK DIFFERENTIATION RELEASE	518	---	---	---	214
MIN	FIND MINIMUM	183	@MIN	---	---	731
MLPX	DATA DECODER	076	@MLPX	---	---	496
MOV	MOVE	021	@MOV	---	!MOV	331
MOV$	MOVE STRING	664	@MOV$	---	---	1221
MOVB	MOVE BIT	082	@MOVB	---	---	337
MOVBC	MOVE BIT	568	@MOVBC	---	---	1273
MOVD	MOVE DIGIT	083	@MOVD	---	---	339
MOVF	MOVE FLOATING-POINT (SINGLE)	469	@MOVF	---	---	649
MOVL	DOUBLE MOVE	498	@MOVL	---	---	334
MOVR	MOVE TO REGISTER	560	@MOVR	---	---	356
MOVRW	MOVE TIMER/ COUNTER PV TO REGISTER	561	---	---	---	358
MSG	DISPLAY MESSAGE	046	@MSG	---	---	1119
MSKR	READ INTERRUPT MASK	692	@MSKR	---	---	846
MSKS	SET INTERRUPT MASK	690	@MSKS	---	---	839

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
MTIM	MULTI-OUTPUT TIMER	543	---	---	---	269
MTIMX	MULTI-OUTPUT TIMER	554	---	---	---	269
MTR	MATRIX INPUT	213	---	---	---	953
MVN	MOVE NOT	022	@MVN	---	---	333
MVNL	DOUBLE MOVE NOT	499	@MVNL	---	---	336

N

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
NASL	SHIFT N-BITS LEFT	580	@NASL	---	---	397
NASR	SHIFT N-BITS RIGHT	581	@NASR	---	---	403
NEG	2'S COMPLEMENT	160	@NEG	---	---	491
NEGL	DOUBLE 2'S COMPLEMENT	161	@NEGL	---	---	493
NEXT	FOR-NEXT LOOPS	513	---	---	---	238
NOP	NO OPERATION	000	---	---	---	207
NOT	NOT	520	---	---	---	180
NSFL	SHIFT N-BIT DATA LEFT	578	@NSFL	---	---	393
NSFR	SHIFT N-BIT DATA RIGHT	579	@NSFR	---	---	395
NSLL	DOUBLE SHIFT N-BITS LEFT	582	@NSLL	---	---	400
NSRL	DOUBLE SHIFT N-BITS RIGHT	583	@NSRL	---	---	405
NUM4	ASCII TO FOUR-DIGIT NUMBER	604	@NUM4	---	---	534
NUM8	ASCII TO EIGHT-DIGIT NUMBER	605	@NUM8	---	---	537
NUM16	ASCII TO SIXTEEN-DIGIT NUMBER	606	@NUM16	---	---	539

O

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
OR	OR	---	@OR	%OR	!OR	169
OR <	OR LESS THAN	310	---	---	---	291
OR <$	OR STRING LESS THAN	672	---	---	---	1250
OR <>	OR NOT EQUAL	305	---	---	---	291
OR <>$	OR STRING NOT EQUAL	671	---	---	---	1250
OR <>D	OR DOUBLE FLOATING NOT EQUAL	336	---	---	---	694
OR <>DT	OR TIME NOT EQUAL	342	---	---	---	297
OR <>F	OR FLOATING NOT EQUAL	330	---	---	---	636
OR <>L	OR DOUBLE NOT EQUAL	306	---	---	---	291
OR <>S	OR SIGNED NOT EQUAL	307	---	---	---	291
OR <>SL	OR DOUBLE SIGNED NOT EQUAL	308	---	---	---	291
OR <D	OR DOUBLE FLOATING LESS THAN	337	---	---	---	694

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
OR <DT	OR TIME LESS THAN	343	---	---	---	297
OR <F	OR FLOATING LESS THAN	331	---	---	---	636
OR <L	OR DOUBLE LESS THAN	311	---	---	---	291
OR <S	OR SIGNED LESS THAN	312	---	---	---	291
OR <SL	OR DOUBLE SIGNED LESS THAN	313	---	---	---	291
OR =	OR EQUAL	300	---	---	---	291
OR =$	OR STRING EQUALS	670	---	---	---	1250
OR =D	OR DOUBLE FLOATING EQUAL	335	---	---	---	694
OR =DT	OR TIME EQUAL	341	---	---	---	297
OR =F	OR FLOATING EQUAL	329	---	---	---	636
OR =L	OR DOUBLE EQUAL	301	---	---	---	291
OR =S	OR SIGNED EQUAL	302	---	---	---	291
OR =SL	OR DOUBLE SIGNED EQUAL	303	---	---	---	291
OR >	OR GREATER THAN	320	---	---	---	291
OR >$	OR STRING GREATER THAN	674	---	---	---	1250
OR >D	OR DOUBLE FLOATING GREATER THAN	339	---	---	---	694
OR >DT	OR TIME GREATER THAN	345	---	---	---	297
OR >F	OR FLOATING GREATER THAN	333	---	---	---	636
OR >L	OR DOUBLE GREATER THAN	321	---	---	---	291
OR >S	OR SIGNED GREATER THAN	322	---	---	---	291
OR >SL	OR DOUBLE SIGNED GREATER THAN	323	---	---	---	291
OR LD	OR LOAD	---	---	---	---	174
OR NOT	OR NOT	---	---	---	!OR NOT	171
OR TST	OR BIT TEST	350	---	---	---	182
OR TSTN	OR BIT TEST	351	---	---	---	182
OR <=	OR LESS THAN OR EQUAL	315	---	---	---	291
OR <=$	OR STRING LESS THAN OR EQUALS	673	---	---	---	1250
OR <=D	OR DOUBLE FLOATING LESS THAN OR EQUAL	338	---	---	---	694
OR <=DT	OR TIME LESS THAN OR EQUAL	344	---	---	---	297
OR <=F	OR FLOATING LESS THAN OR EQUAL	332	---	---	---	636
OR <=L	OR DOUBLE LESS THAN OR EQUAL	316	---	---	---	291
OR <=S	OR SIGNED LESS THAN OR EQUAL	317	---	---	---	291
OR <=SL	OR DOUBLE SIGNED LESS THAN OR EQUAL	318	---	---	---	291
OR >=	OR GREATER THAN OR EQUAL	325	---	---	---	291
OR >=$	OR STRING GREATER THAN OR EQUALS	675	---	---	---	1250

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
OR >=D	OR DOUBLE FLOATING GREATER THAN OR EQUAL	340	---	---	---	694
OR >=DT	OR TIME GREATER THAN OR EQUAL	346	---	---	---	297
OR >=F	OR FLOATING GREATER THAN OR EQUAL	334	---	---	---	636
OR >=L	OR DOUBLE GREATER THAN OR EQUAL	326	---	---	---	291
OR >=S	OR SIGNED GREATER THAN OR EQUAL	327	---	---	---	291
OR >=SL	OR DOUBLE SIGNED GREATER THAN OR EQUAL	328	---	---	---	291
ORG	ORIGIN SEARCH	889	@ORG	---	---	903
ORW	LOGICAL OR	035	@ORW	---	---	551
ORWL	DOUBLE LOGICAL OR	611	@ORWL	---	---	553
OUT	OUTPUT	---	---	---	!OUT	185
OUTB	SINGLE BIT OUTPUT	534	@OUTB	---	!OUTB	204
OUT NOT	OUTPUT NOT	---	---	---	!OUT NOT	187

P

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
PID	PID CONTROL	190	---	---	---	757
PIDAT	PID CONTROL WITH AUTOTUNING	191	---	---	---	769
PMCR	PROTOCOL MACRO	260	@PMCR	---	---	974
PRV	HIGH-SPEED COUNTER PV READ	881	@PRV	---	---	868
PRV2	COUNTER FREQUENCY CONVERT	883	@PRV2	---	---	874
PULS	SET PULSES	886	@PULS	---	---	887
PLS2	PULSE OUTPUT	887	@PLS2	---	---	890
PUSH	PUSH ONTO STACK	632	@PUSH	---	---	706
PWM	PULSE WITH VARIABLE DUTY FACTOR	891	@PWM	---	---	906
PWR	EXPONENTIAL POWER	840	@PWR	---	---	635
PWRD	DOUBLE EXPONENTIAL POWER	860	@PWRD	---	---	692

R

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
RAD	DEGREES TO RADIANS	458	@RAD	---	---	633
RADD	DOUBLE DEGREES TO RADIANS	849	@RADD	---	---	671
RECV	NETWORK RECEIVE	098	@RECV	---	---	1050
RET	SUBROUTINE RETURN	093	---	---	---	824
RGHT$	GET STRING RIGHT	653	@RGHT$	---	---	1228
RLNC	ROTATE LEFT WITHOUT CARRY	574	@RLNC	---	---	383

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
RLNL	DOUBLE ROTATE LEFT WITHOUT CARRY	576	@RLNL	---	---	385
ROL	ROTATE LEFT	027	@ROL	---	---	376
ROLL	DOUBLE ROTATE LEFT	572	@ROLL	---	---	378
ROOT	BCD SQUARE ROOT	072	@ROOT	---	---	567
ROR	ROTATE RIGHT	028	@ROR	---	---	380
RORL	DOUBLE ROTATE RIGHT	573	@RORL	---	---	381
ROTB	BINARY ROOT	620	@ROTB	---	---	565
RPLC$	REPLACE IN STRING	661	@RPLC$	---	---	1237
RRNC	ROTATE RIGHT WITHOUT CARRY	575	@RRNC	---	---	387
RRNL	DOUBLE ROTATE RIGHT WITHOUT CARRY	577	@RRNL	---	---	388
RSET	RESET	---	@RSET	%RSET	!RSET	195
RSTA	MULTIPLE BIT RESET	531	@RSTA	---	---	198
RSTB	SINGLE BIT RESET	533	@RSTB	---	!RSTB	201
RXD	RECEIVE	235	@RXD	---	---	993
RXDU	RECEIVE VIA SERIAL COMMUNICATIONS UNIT	255	@RXDU	---	---	1013

S

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
SBN	SUBROUTINE ENTRY	092	---	---	---	821
SBS	SUBROUTINE CALL	091	@SBS	---	---	811
SCL	SCALING	194	@SCL	---	---	795
SCL2	SCALING 2	486	@SCL2	---	---	800
SCL3	SCALING 3	487	@SCL3	---	---	804
SDEC	7-SEGMENT DECODER	078	@SDEC	---	---	974
SDEL	STACK DATA DELETE	642	@SDEL	---	---	753
SEC	HOURS TO SECONDS	065	@SEC	---	---	1129
SEND	NETWORK SEND	090	@SEND	---	---	1044
SET	SET	---	@SET	%SET	!SET	195
SETA	MULTIPLE BIT SET	530	@SETA	---	---	198
SETB	SINGLE BIT SET	532	@SETB	---	!SETB	201
SETR	SET RECORD LOCATION	635	@SETR	---	---	718
SFT	SHIFT REGISTER	010	---	---	---	361
SFTR	REVERSIBLE SHIFT REGISTER	084	@SFTR	---	---	362
SIGN	16-BIT TO 32-BIT SIGNED BINARY	600	@SIGN	---	---	494
SIN	SINE	460	@SIN	---	---	612
SIND	DOUBLE SINE	851	@SIND	---	---	674
SINQ	HIGH-SPEED SINE	475	@SINQ	---	---	614
SINS	STACK DATA INSERT	641	@SINS	---	---	750
SLD	ONE DIGIT SHIFT LEFT	074	@SLD	---	---	390
SNUM	STACK SIZE READ	638	@SNUM	---	---	742
SNXT	STEP START	009	---	---	---	909

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
SPED	SPEED OUTPUT	885	@SPED	---	---	882
SQRT	SQUARE ROOT	466	@SQRT	---	---	629
SQRTD	DOUBLE SQUARE ROOT	857	@SQRTD	---	---	686
SRCH	DATA SEARCH	181	@SRCH	---	---	722
SRD	ONE DIGIT SHIFT RIGHT	075	@SRD	---	---	392
SREAD	STACK DATA READ	639	@SREAD	---	---	744
SSET	SET STACK	630	@SSET	---	---	703
STC	SET CARRY	040	@STC	---	---	1166
STEP	STEP DEFINE	008	---	---	---	909
STR4	FOUR-DIGIT NUMBER TO ASCII	601	@STR4	---	---	541
STR8	EIGHT-DIGIT NUMBER TO ASCII	602	@STR8	---	---	544
STR16	SIXTEEN-DIGIT NUMBER TO ASCII	603	@STR16	---	---	545
STUP	CHANGE SERIAL PORT SETUP	237	@STUP	---	---	1021
SUM	SUM	184	@SUM	---	---	735
SWAP	SWAP BYTES	637	@SWAP	---	---	725
SWRIT	STACK DATA WRITE	640	@SWRIT	---	---	747

T

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
TAN	TANGENT	462	@TAN	---	---	619
TAND	DOUBLE TANGENT	853	@TAND	---	---	678
TANQ	HIGH-SPEED TANGENT	477	@TANQ	---	---	621
TCMP	TABLE COMPARE	085	@TCMP	---	---	317
TIM	HUNDRED-MS TIMER	---	---	---	---	245
TIMH	TEN-MS TIMER	015	---	---	---	249
TIMHX	TEN-MS TIMER	551	---	---	---	249
TIML	LONG TIMER	542	---	---	---	266
TIMLX	LONG TIMER	553	---	---	---	266
TIMU	TENTH-MS TIMER	541	---	---	---	256
TIMUX	TENTH-MS TIMER	556	---	---	---	256
TIMW	HUNDRED-MS TIMER WAIT	813	---	---	---	1206
TIMWX	HUNDRED-MS TIMER WAIT	816	---	---	---	1206
TIMX	HUNDRED-MS TIMER	550	---	---	---	245
TKOF	TASK OFF	821	@TKOF	---	---	1258
TKON	TASK ON	820	@TKON	---	---	1255
TKY	TEN KEY INPUT	211	@TKY	---	---	945
TMHH	ONE-MS TIMER	540	---	---	---	253
TMHHX	ONE-MS TIMER	552	---	---	---	253
TMHW	TEN-MS TIMER WAIT	815	---	---	---	1212
TMHWX	TEN-MS TIMER WAIT	817	---	---	---	1212
TMUH	HUNDREDTH-MS TIMER	544	---	---	---	259
TMUHX	HUNDREDTH-MS TIMER	557	---	---	---	259

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
TOCV	CONVERT ADDRESS TO CV	285	@TOCV	---	---	1179
TPO	TIME-PROPOR-TIONAL OUTPUT	685	---	---	---	787
TRSM	TRACE MEMORY SAMPLING	045	---	---	---	1136
TTIM	ACCUMULATIVE TIMER	087	---	---	---	262
TTIMX	ACCUMULATIVE TIMER	555	---	---	---	262
TWRIT	WRITE TEXT FILE	704	@TWRIT	---	---	1113
TXD	TRANSMIT	236	@TXD	---	---	983
TXDU	TRANSMIT VIA SERIAL COMMUNICA-TIONS UNIT	256	@TXDU	---	---	1005

U

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
UP	CONDITION ON	521	---	---	---	181

W

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
WAIT NOT (operand)	ONE CYCLE AND WAIT NOT	805	---	---	---	1202
WAIT (input condition)	ONE CYCLE AND WAIT	805	---	---	---	1202
WAIT (operand)	ONE CYCLE AND WAIT	805	---	---	---	1202
WDT	EXTEND MAXIMUM CYCLE TIME	094	@WDT	---	---	1169
WSFT	WORD SHIFT	016	@WSFT	---	---	368

X

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
XCGL	DOUBLE DATA EXCHANGE	562	@XCGL	---	---	350
XCHG	DATA EXCHANGE	073	@XCHG	---	---	349
XCHG$	EXCHANGE STRING	665	@XCHG$	---	---	1242
XFER	BLOCK TRANSFER	070	@XFER	---	---	344
XFERC	BLOCK TRANSFER	565	@XFERC	---	---	1263
XFRB	MULTIPLE BIT TRANSFER	062	@XFRB	---	---	342
XNRL	DOUBLE EXCLUSIVE NOR	613	@XNRL	---	---	560
XNRW	EXCLUSIVE NOR	037	@XNRW	---	---	559
XORL	DOUBLE EXCLUSIVE OR	612	@XORL	---	---	557
XORW	EXCLUSIVE OR	036	@XORW	---	---	555

Z

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
ZCP	AREA RANGE COMPARE	088	---	---	---	326
ZCPL	DOUBLE AREA RANGE COMPARE	116	---	---	---	329
ZONE	DEAD ZONE CONTROL	682	@ZONE	---	---	784

Symbols

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
7SEG	7-SEGMENT DISPLAY OUTPUT	214	---	---	---	957
+	SIGNED BINARY ADD WITHOUT CARRY	400	@+	---	---	426
+$	CONCATENATE STRING	656	@+$	---	---	1223
++	INCREMENT BINARY	590	@++	---	---	409
++B	INCREMENT BCD	594	@++B	---	---	417
++BL	DOUBLE INCREMENT BCD	595	@++BL	---	---	419
++L	DOUBLE INCREMENT BINARY	591	@++L	---	---	411
+B	BCD ADD WITHOUT CARRY	404	@+B	---	---	434
+BC	BCD ADD WITH CARRY	406	@+BC	---	---	437
+BCL	DOUBLE BCD ADD WITH CARRY	407	@+BCL	---	---	439
+BL	DOUBLE BCD ADD WITHOUT CARRY	405	@+BL	---	---	435
+C	SIGNED BINARY ADD WITH CARRY	402	@+C	---	---	430
+CL	DOUBLE SIGNED BINARY ADD WITH CARRY	403	@+CL	---	---	432
+D	DOUBLE FLOATING-POINT ADD	845	@+D	---	---	663
+F	FLOATING-POINT ADD	454	@+F	---	---	601
+L	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	401	@+L	---	---	428
–	SIGNED BINARY SUBTRACT WITHOUT CARRY	410	@–	---	---	440
– –	DECREMENT BINARY	592	@– –	---	---	413
– –B	DECREMENT BCD	596	@– –B	---	---	421
– –BL	DOUBLE DECREMENT BCD	597	@– –BL	---	---	423
– –L	DOUBLE DECREMENT BINARY	593	@– –L	---	---	415
–B	BCD SUBTRACT WITHOUT CARRY	414	@–B	---	---	451
–BC	BCD SUBTRACT WITH CARRY	416	@–BC	---	---	456
–BCL	DOUBLE BCD SUBTRACT WITH CARRY	417	@–BCL	---	---	457

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
−BL	DOUBLE BCD SUBTRACT WITHOUT CARRY	415	@−BL	---	---	452
−C	SIGNED BINARY SUBTRACT WITH CARRY	412	@−C	---	---	446
−CL	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	413	@−CL	---	---	448
−D	DOUBLE FLOATING-POINT SUBTRACT	846	@−D	---	---	665
−F	FLOATING-POINT SUBTRACT	455	@−F	---	---	603
*	SIGNED BINARY MULTIPLY	420	@*	---	---	459
*B	BCD MULTIPLY	424	@*B	---	---	467
*BL	DOUBLE BCD MULTIPLY	425	@*BL	---	---	469
*D	DOUBLE FLOATING-POINT MULTIPLY	847	@*D	---	---	667
*F	FLOATING-POINT MULTIPLY	456	@*F	---	---	605
*L	DOUBLE SIGNED BINARY MULTIPLY	421	@*L	---	---	461
*U	UNSIGNED BINARY MULTIPLY	422	@*U	---	---	463
*UL	DOUBLE UNSIGNED BINARY MULTIPLY	423	@*UL	---	---	465
−L	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	411	@−L	---	---	442
/	SIGNED BINARY DIVIDE	430	@/	---	---	471
/B	BCD DIVIDE	434	@/B	---	---	479
/BL	DOUBLE BCD DIVIDE	435	@/BL	---	---	481
/D	DOUBLE FLOATING-POINT DIVIDE	848	@/D	---	---	669
/F	FLOATING-POINT DIVIDE	457	@/F	---	---	607
/L	DOUBLE SIGNED BINARY DIVIDE	431	@/L	---	---	473
/U	UNSIGNED BINARY DIVIDE	432	@/U	---	---	475
/UL	DOUBLE UNSIGNED BINARY DIVIDE	433	@/UL	---	---	477

2-4 List of Instructions by Function Code

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
---	LD	LOAD	@LD	%LD	!LD	161
---	LD NOT	LOAD NOT	---	---	!LD NOT	163
---	AND	AND	@AND	%AND	!AND	165
---	AND NOT	AND NOT	---	---	!AND NOT	167
---	OR	OR	@OR	%OR	!OR	169
---	OR NOT	OR NOT	---	---	!OR NOT	171
---	AND LD	AND LOAD	---	---	---	172
---	OR LD	OR LOAD	---	---	---	174
---	OUT	OUTPUT	---	---	!OUT	185
---	OUT NOT	OUTPUT NOT	---	---	!OUT NOT	187
---	SET	SET	@SET	%SET	!SET	195
---	RSET	RESET	@RSET	%RSET	!RSET	195
---	TIM	HUNDRED-MS TIMER	---	---	---	245
---	CNT	COUNTER	---	---	---	275
000	NOP	NO OPERATION	---	---	---	207
001	END	END	---	---	---	206
002	IL	INTERLOCK	---	---	---	210
003	ILC	INTERLOCK CLEAR	---	---	---	210
004	JMP	JUMP	---	---	---	228
005	JME	JUMP END	---	---	---	228
006	FAL	FAILURE ALARM	@FAL	---	---	1140
007	FALS	SEVERE FAILURE ALARM	---	---	---	1148
008	STEP	STEP DEFINE	---	---	---	909
009	SNXT	STEP START	---	---	---	909
010	SFT	SHIFT REGISTER	---	---	---	361
011	KEEP	KEEP	---	---	!KEEP	188
012	CNTR	REVERSIBLE COUNTER	---	---	---	278
013	DIFU	DIFFERENTIATE UP	---	---	!DIFU	193
014	DIFD	DIFFERENTIATE DOWN	---	---	!DIFD	193
015	TIMH	TEN-MS TIMER	---	---	---	249
016	WSFT	WORD SHIFT	@WSFT	---	---	368
017	ASFT	ASYNCHRONOUS SHIFT REGISTER	@ASFT	---	---	365
019	MCMP	MULTIPLE COMPARE	@MCMP	---	---	315
020	CMP	UNSIGNED COMPARE	---	---	!CMP	303
021	MOV	MOVE	@MOV	---	!MOV	331
022	MVN	MOVE NOT	@MVN	---	---	333
023	BIN	BCD TO BINARY	@BIN	---	---	483
024	BCD	BINARY TO BCD	@BCD	---	---	487
025	ASL	ARITHMETIC SHIFT LEFT	@ASL	---	---	370
026	ASR	ARITHMETIC SHIFT RIGHT	@ASR	---	---	373
027	ROL	ROTATE LEFT	@ROL	---	---	376
028	ROR	ROTATE RIGHT	@ROR	---	---	380
029	COM	COMPLEMENT	@COM	---	---	562
034	ANDW	LOGICAL AND	@ANDW	---	---	548
035	ORW	LOGICAL OR	@ORW	---	---	551
036	XORW	EXCLUSIVE OR	@XORW	---	---	555

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
037	XNRW	EXCLUSIVE NOR	@XNRW	---	---	559
040	STC	SET CARRY	@STC	---	---	1166
041	CLC	CLEAR CARRY	@CLC	---	---	1166
045	TRSM	TRACE MEMORY SAMPLING	---	---	---	1136
046	MSG	DISPLAY MESSAGE	@MSG	---	---	1119
058	BINL	DOUBLE BCD TO DOUBLE BINARY	@BINL	---	---	485
059	BCDL	DOUBLE BINARY TO BCD	@BCDL	---	---	489
060	CMPL	DOUBLE UNSIGNED COMPARE	---	---	---	306
062	XFRB	MULTIPLE BIT TRANSFER	@XFRB	---	---	342
063	LINE	COLUMN TO LINE	@LINE	---	---	512
064	COLM	LINE TO COLUMN	@COLM	---	---	514
065	SEC	HOURS TO SECONDS	@SEC	---	---	1129
066	HMS	SECONDS TO HOURS	@HMS	---	---	1131
067	BCNT	BIT COUNTER	@BCNT	---	---	587
068	BCMP	UNSIGNED BLOCK COMPARE	@BCMP	---	---	320
069	APR	ARITHMETIC PROCESS	@APR	---	---	571
070	XFER	BLOCK TRANSFER	@XFER	---	---	344
071	BSET	BLOCK SET	@BSET	---	---	347
072	ROOT	BCD SQUARE ROOT	@ROOT	---	---	567
073	XCHG	DATA EXCHANGE	@XCHG	---	---	349
074	SLD	ONE DIGIT SHIFT LEFT	@SLD	---	---	390
075	SRD	ONE DIGIT SHIFT RIGHT	@SRD	---	---	392
076	MLPX	DATA DECODER	@MLPX	---	---	496
077	DMPX	DATA ENCODER	@DMPX	---	---	500
078	SDEC	7-SEGMENT DECODER	@SDEC	---	---	974
079	FDIV	FLOATING POINT DIVIDE	@FDIV	---	---	583
080	DIST	SINGLE WORD DISTRIBUTE	@DIST	---	---	352
081	COLL	DATA COLLECT	@COLL	---	---	354
082	MOVB	MOVE BIT	@MOVB	---	---	337
083	MOVD	MOVE DIGIT	@MOVD	---	---	339
084	SFTR	REVERSIBLE SHIFT REGISTER	@SFTR	---	---	362
085	TCMP	TABLE COMPARE	@TCMP	---	---	317
086	ASC	ASCII CONVERT	@ASC	---	---	504
087	TTIM	ACCUMULATIVE TIMER	---	---	---	262
088	ZCP	AREA RANGE COMPARE	---	---	---	326
090	SEND	NETWORK SEND	@SEND	---	---	1044
091	SBS	SUBROUTINE CALL	@SBS	---	---	811
092	SBN	SUBROUTINE ENTRY	---	---	---	821
093	RET	SUBROUTINE RETURN	---	---	---	824
094	WDT	EXTEND MAXIMUM CYCLE TIME	@WDT	---	---	1169

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
096	BPRG	BLOCK PROGRAM BEGIN	---	---	---	1191
097	IORF	I/O REFRESH	@IORF	---	---	926
098	RECV	NETWORK RECEIVE	@RECV	---	---	1050
099	MCRO	MACRO	@MCRO	---	---	817
114	CPS	SIGNED BINARY COMPARE	---	---	!CPS	309
115	CPSL	DOUBLE SIGNED BINARY COMPARE	---	---	---	312
116	ZCPL	DOUBLE AREA RANGE COMPARE	---	---	---	329
160	NEG	2'S COMPLEMENT	@NEG	---	---	491
161	NEGL	DOUBLE 2'S COMPLEMENT	@NEGL	---	---	493
162	HEX	ASCII TO HEX	@HEX	---	---	508
180	FCS	FRAME CHECKSUM	@FCS	---	---	738
181	SRCH	DATA SEARCH	@SRCH	---	---	722
182	MAX	FIND MAXIMUM	@MAX	---	---	727
183	MIN	FIND MINIMUM	@MIN	---	---	731
184	SUM	SUM	@SUM	---	---	735
190	PID	PID CONTROL	---	---	---	757
191	PIDAT	PID CONTROL WITH AUTOTUNING	---	---	---	769
194	SCL	SCALING	@SCL	---	---	795
195	AVG	AVERAGE	---	---	---	807
210	DSW	DIGITAL SWITCH INPUT	---	---	---	940
211	TKY	TEN KEY INPUT	@TKY	---	---	945
212	HKY	HEXADECIMAL KEY INPUT	---	---	---	948
213	MTR	MATRIX INPUT	---	---	---	953
214	7SEG	7-SEGMENT DISPLAY OUTPUT	---	---	---	957
222	IORD	INTELLIGENT I/O READ	@IORD	---	---	962
223	IOWR	INTELLIGENT I/O WRITE	@IOWR	---	---	967
225	FIORF	SPECIAL I/O UNIT I/O REFRESH	@FIORF	---	---	929
226	DLNK	CPU BUS UNIT I/O REFRESH	@DLNK	---	---	932
235	RXD	RECEIVE	@RXD	---	---	993
236	TXD	TRANSMIT	@TXD	---	---	983
255	RXDU	RECEIVE VIA SERIAL COMMUNICATIONS UNIT	@RXDU	---	---	1013
256	TXDU	TRANSMIT VIA SERIAL COMMUNICA-TIONS UNIT	@TXDU	---	---	1005
237	STUP	CHANGE SERIAL PORT SETUP	@STUP	---	---	1021
260	PMCR	PROTOCOL MACRO	@PMCR	---	---	974
269	FPD	FAILURE POINT DETECTION	---	---	---	1156
281	EMBC	SELECT EM BANK	@EMBC	---	---	1167
282	CCS	SAVE CONDITION FLAGS	@CCS	---	---	1171
283	CCL	LOAD CONDITION FLAGS	@CCL	---	---	1173

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
284	FRMCV	CONVERT ADDRESS FROM CV	@FRMCV	---	---	1174
285	TOCV	CONVERT ADDRESS TO CV	@TOCV	---	---	1179
286	GETID	GET VARIABLE ID	@GETID	---	---	1277
287	IOSP	DISABLE PERIPH-ERAL SERVICING	@IOSP	---	---	1183
288	IORS	ENABLE PERIPH-ERAL SERVICING	---	---	---	1185
300	AND =	AND EQUAL	---	---	---	291
300	LD =	LOAD EQUAL	---	---	---	291
300	OR =	OR EQUAL	---	---	---	291
301	AND =L	AND DOUBLE EQUAL	---	---	---	291
301	LD =L	LOAD DOUBLE EQUAL	---	---	---	291
301	OR =L	OR DOUBLE EQUAL	---	---	---	291
302	AND =S	AND SIGNED EQUAL	---	---	---	291
302	LD =S	LOAD SIGNED EQUAL	---	---	---	291
302	OR =S	OR SIGNED EQUAL	---	---	---	291
303	AND =SL	AND DOUBLE SIGNED EQUAL	---	---	---	291
303	LD =SL	LOAD DOUBLE SIGNED EQUAL	---	---	---	291
303	OR =SL	OR DOUBLE SIGNED EQUAL	---	---	---	291
305	AND <>	AND NOT EQUAL	---	---	---	291
305	LD <>	LOAD NOT EQUAL	---	---	---	291
305	OR <>	OR NOT EQUAL	---	---	---	291
306	AND <>L	AND DOUBLE NOT EQUAL	---	---	---	291
306	LD <>L	LOAD DOUBLE NOT EQUAL	---	---	---	291
306	OR <>L	OR DOUBLE NOT EQUAL	---	---	---	291
307	AND <>S	AND SIGNED NOT EQUAL	---	---	---	291
307	LD <>S	LOAD SIGNED NOT EQUAL	---	---	---	291
307	OR <>S	OR SIGNED NOT EQUAL	---	---	---	291
308	AND <>SL	AND DOUBLE SIGNED NOT EQUAL	---	---	---	291
308	LD <>SL	LOAD DOUBLE SIGNED NOT EQUAL	---	---	---	291
308	OR <>SL	OR DOUBLE SIGNED NOT EQUAL	---	---	---	291
310	AND <	AND LESS THAN	---	---	---	291
310	LD <	LOAD LESS THAN	---	---	---	291
310	OR <	OR LESS THAN	---	---	---	291
311	AND <L	AND DOUBLE LESS THAN	---	---	---	291
311	LD <L	LOAD DOUBLE LESS THAN	---	---	---	291
311	OR <L	OR DOUBLE LESS THAN	---	---	---	291
312	AND <S	AND SIGNED LESS THAN	---	---	---	291
312	LD <S	LOAD SIGNED LESS THAN	---	---	---	291

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
312	OR <S	OR SIGNED LESS THAN	---	---	---	291
313	AND <SL	AND DOUBLE SIGNED LESS THAN	---	---	---	291
313	LD <SL	LOAD DOUBLE SIGNED LESS THAN	---	---	---	291
313	OR <SL	OR DOUBLE SIGNED LESS THAN	---	---	---	291
315	AND <=	AND LESS THAN OR EQUAL	---	---	---	291
315	LD <=	LOAD LESS THAN OR EQUAL	---	---	---	291
315	OR <=	OR LESS THAN OR EQUAL	---	---	---	291
316	AND <=L	AND DOUBLE LESS THAN OR EQUAL	---	---	---	291
316	LD <=L	LOAD DOUBLE LESS THAN OR EQUAL	---	---	---	291
316	OR <=L	OR DOUBLE LESS THAN OR EQUAL	---	---	---	291
317	AND <=S	AND SIGNED LESS THAN OR EQUAL	---	---	---	291
317	LD <=S	LOAD SIGNED LESS THAN OR EQUAL	---	---	---	291
317	OR <=S	OR SIGNED LESS THAN OR EQUAL	---	---	---	291
318	AND <=SL	AND DOUBLE SIGNED LESS THAN OR EQUAL	---	---	---	291
318	LD <=SL	LOAD DOUBLE SIGNED LESS THAN OR EQUAL	---	---	---	291
318	OR <=SL	OR DOUBLE SIGNED LESS THAN OR EQUAL	---	---	---	291
320	AND >	AND GREATER THAN	---	---	---	291
320	LD >	LOAD GREATER THAN	---	---	---	291
320	OR >	OR GREATER THAN	---	---	---	291
321	AND >L	AND DOUBLE GREATER THAN	---	---	---	291
321	LD >L	LOAD DOUBLE GREATER THAN	---	---	---	291
321	OR >L	OR DOUBLE GREATER THAN	---	---	---	291
322	AND >S	AND SIGNED GREATER THAN	---	---	---	291
322	LD >S	LOAD SIGNED GREATER THAN	---	---	---	291
322	OR >S	OR SIGNED GREATER THAN	---	---	---	291
323	AND >SL	AND DOUBLE SIGNED GREATER THAN	---	---	---	291
323	LD >SL	LOAD DOUBLE SIGNED GREATER THAN	---	---	---	291
323	OR >SL	OR DOUBLE SIGNED GREATER THAN	---	---	---	291
325	AND >=	AND GREATER THAN OR EQUAL	---	---	---	291
325	LD >=	LOAD GREATER THAN OR EQUAL	---	---	---	291

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
325	OR >=	OR GREATER THAN OR EQUAL	---	---	---	291
326	AND >=L	AND DOUBLE GREATER THAN OR EQUAL	---	---	---	291
326	LD >=L	LOAD DOUBLE GREATER THAN OR EQUAL	---	---	---	291
326	OR >=L	OR DOUBLE GREATER THAN OR EQUAL	---	---	---	291
327	AND >=S	AND SIGNED GREATER THAN OR EQUAL	---	---	---	291
327	LD >=S	LOAD SIGNED GREATER THAN OR EQUAL	---	---	---	291
327	OR >=S	OR SIGNED GREATER THAN OR EQUAL	---	---	---	291
328	AND >=SL	AND DOUBLE SIGNED GREATER THAN OR EQUAL	---	---	---	291
328	LD >=SL	LOAD DOUBLE SIGNED GREATER THAN OR EQUAL	---	---	---	291
328	OR >=SL	OR DOUBLE SIGNED GREATER THAN OR EQUAL	---	---	---	291
329	AND =F	AND FLOATING EQUAL	---	---	---	636
329	LD =F	LOAD FLOATING EQUAL	---	---	---	636
329	OR =F	OR FLOATING EQUAL	---	---	---	636
330	AND <>F	AND FLOATING NOT EQUAL	---	---	---	636
330	LD <>F	LOAD FLOATING NOT EQUAL	---	---	---	636
330	OR <>F	OR FLOATING NOT EQUAL	---	---	---	636
331	AND <F	AND FLOATING LESS THAN	---	---	---	636
331	LD <F	LOAD FLOATING LESS THAN	---	---	---	636
331	OR <F	OR FLOATING LESS THAN	---	---	---	636
332	AND <=F	AND FLOATING LESS THAN OR EQUAL	---	---	---	636
332	LD <=F	LOAD FLOATING LESS THAN OR EQUAL	---	---	---	636
332	OR <=F	OR FLOATING LESS THAN OR EQUAL	---	---	---	636
333	AND >F	AND FLOATING GREATER THAN	---	---	---	636
333	LD >F	LOAD FLOATING GREATER THAN	---	---	---	636
333	OR >F	OR FLOATING GREATER THAN	---	---	---	636
334	AND >=F	AND FLOATING GREATER THAN OR EQUAL	---	---	---	636
334	LD >=F	LOAD FLOATING GREATER THAN OR EQUAL	---	---	---	636

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
334	OR >=F	OR FLOATING GREATER THAN OR EQUAL	---	---	---	636
335	AND =D	AND DOUBLE FLOATING EQUAL	---	---	---	694
335	LD =D	LOAD DOUBLE FLOATING EQUAL	---	---	---	694
335	OR =D	OR DOUBLE FLOATING EQUAL	---	---	---	694
336	AND <>D	AND DOUBLE FLOATING NOT EQUAL	---	---	---	694
336	LD <>D	LOAD DOUBLE FLOATING NOT EQUAL	---	---	---	694
336	OR <>D	OR DOUBLE FLOATING NOT EQUAL	---	---	---	694
337	AND <D	AND DOUBLE FLOATING LESS THAN	---	---	---	694
337	LD <D	LOAD DOUBLE FLOATING LESS THAN	---	---	---	694
337	OR <D	OR DOUBLE FLOATING LESS THAN	---	---	---	694
338	AND <=D	AND DOUBLE FLOATING LESS THAN OR EQUAL	---	---	---	694
338	LD <=D	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	---	---	---	694
338	OR <=D	OR DOUBLE FLOATING LESS THAN OR EQUAL	---	---	---	694
339	AND >D	AND DOUBLE FLOATING GREATER THAN	---	---	---	694
339	LD >D	LOAD DOUBLE FLOATING GREATER THAN	---	---	---	694
339	OR >D	OR DOUBLE FLOATING GREATER THAN	---	---	---	694
340	AND >=D	AND DOUBLE FLOATING GREATER THAN OR EQUAL	---	---	---	694
340	LD >=D	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	---	---	---	694
340	OR >=D	OR DOUBLE FLOATING GREATER THAN OR EQUAL	---	---	---	694
341	AND = DT	AND TIME EQUAL	---	---	---	297
341	LD = DT	LOAD TIME EQUAL	---	---	---	297
341	OR = DT	OR TIME EQUAL	---	---	---	297
342	AND <> DT	AND TIME NOT EQUAL	---	---	---	297
342	LD <> DT	LOAD TIME NOT EQUAL	---	---	---	297
342	OR <> DT	OR TIME NOT EQUAL	---	---	---	297
343	AND < DT	AND TIME LESS THAN	---	---	---	297
343	LD < DT	LOAD TIME LESS THAN	---	---	---	297
343	OR < DT	OR TIME LESS THAN	---	---	---	297
344	AND <= DT	AND TIME LESS THAN OR EQUAL	---	---	---	297

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
344	LD <= DT	LD TIME LESS THAN OR EQUAL	---	---	---	297
344	OR <= DT	OR TIME LESS THAN OR EQUAL	---	---	---	297
345	AND > DT	AND TIME GREATER THAN	---	---	---	297
345	LD > DT	LOAD TIME GREATER THAN	---	---	---	297
345	OR > DT	OR TIME GREATER THAN	---	---	---	297
346	AND >= DT	AND TIME GREATER THAN OR EQUAL	---	---	---	297
346	LD >= DT	LOAD TIME GREATER THAN OR EQUAL	---	---	---	297
346	OR >= DT	OR TIME GREATER THAN OR EQUAL	---	---	---	297
350	AND TST	AND BIT TEST	---	---	---	182
350	LD TST	LOAD BIT TEST	---	---	---	182
350	OR TST	OR BIT TEST	---	---	---	182
351	AND TSTN	AND BIT TEST NOT	---	---	---	182
351	LD TSTN	LOAD BIT TEST NOT	---	---	---	182
351	OR TSTN	OR BIT TEST NOT	---	---	---	182
400	+	SIGNED BINARY ADD WITHOUT CARRY	@+	---	---	426
401	+L	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	@+L	---	---	428
402	+C	SIGNED BINARY ADD WITH CARRY	@+C	---	---	430
403	+CL	DOUBLE SIGNED BINARY ADD WITH CARRY	@+CL	---	---	432
404	+B	BCD ADD WITHOUT CARRY	@+B	---	---	437
405	+BL	DOUBLE BCD ADD WITHOUT CARRY	@+BL	---	---	435
406	+BC	BCD ADD WITH CARRY	@+BC	---	---	437
407	+BCL	DOUBLE BCD ADD WITH CARRY	@+BCL	---	---	439
410	–	SIGNED BINARY SUBTRACT WITHOUT CARRY	@–	---	---	440
411	–L	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	@–L	---	---	442
412	–C	SIGNED BINARY SUBTRACT WITH CARRY	@–C	---	---	446
413	–CL	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	@–CL	---	---	448
414	–B	BCD SUBTRACT WITHOUT CARRY	@–B	---	---	451
415	–BL	DOUBLE BCD SUBTRACT WITHOUT CARRY	@–BL	---	---	452
416	–BC	BCD SUBTRACT WITH CARRY	@–BC	---	---	456
417	–BCL	DOUBLE BCD SUBTRACT WITH CARRY	@–BCL	---	---	457

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
420	*	SIGNED BINARY MULTIPLY	@*	---	---	459
421	*L	DOUBLE SIGNED BINARY MULTIPLY	@*L	---	---	461
422	*U	UNSIGNED BINARY MULTIPLY	@*U	---	---	463
423	*UL	DOUBLE UNSIGNED BINARY MULTIPLY	@*UL	---	---	465
424	*B	BCD MULTIPLY	@*B	---	---	467
425	*BL	DOUBLE BCD MULTIPLY	@*BL	---	---	469
430	/	SIGNED BINARY DIVIDE	@/	---	---	471
431	/L	DOUBLE SIGNED BINARY DIVIDE	@/L	---	---	473
432	/U	UNSIGNED BINARY DIVIDE	@/U	---	---	475
433	/UL	DOUBLE UNSIGNED BINARY DIVIDE	@/UL	---	---	477
434	/B	BCD DIVIDE	@/B	---	---	479
435	/BL	DOUBLE BCD DIVIDE	@/BL	---	---	481
448	FSTR	FLOATING POINT TO ASCII	@FSTR	---	---	640
449	FVAL	ASCII TO FLOATING POINT	@FVAL	---	---	645
450	FIX	FLOATING TO 16-BIT	@FIX	---	---	594
451	FIXL	FLOATING TO 32-BIT	@FIXL	---	---	596
452	FLT	16-BIT TO FLOATING	@FLT	---	---	597
453	FLTL	32-BIT TO FLOATING	@FLTL	---	---	599
454	+F	FLOATING-POINT ADD	@+F	---	---	601
455	–F	FLOATING-POINT SUBTRACT	@–F	---	---	603
456	*F	FLOATING-POINT MULTIPLY	@*F	---	---	605
457	/F	FLOATING-POINT DIVIDE	@/F	---	---	607
458	RAD	DEGREES TO RADIANS	@RAD	---	---	633
459	DEG	RADIANS-TO DEGREES	@DEG	---	---	610
460	SIN	SINE	@SIN	---	---	612
461	COS	COSINE	@COS	---	---	615
462	TAN	TANGENT	@TAN	---	---	619
463	ASIN	ARC SINE	@ASIN	---	---	623
464	ACOS	ARC COSINE	@ACOS	---	---	625
465	ATAN	ARC TANGENT	@ATAN	---	---	627
466	SQRT	SQUARE ROOT	@SQRT	---	---	629
467	EXP	EXPONENT	@EXP	---	---	631
468	LOG	LOGARITHM	@LOG	---	---	633
469	MOVF	MOVE FLOATING-POINT (SINGLE)	@MOVF	---	---	649
470	BINS	SIGNED BCD TO BINARY	@BINS	---	---	517
471	BCDS	SIGNED BINARY TO BCD	@BCDS	---	---	523
472	BISL	DOUBLE SIGNED BCD TO BINARY	@BISL	---	---	520

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
473	BDSL	DOUBLE SIGNED BINARY TO BCD	@BDSL	---	---	525
474	GRY	GRAY CODE CONVERSION	@GRY	---	---	529
475	SINQ	HIGH-SPEED SINE	@SINQ	---	---	614
476	COSQ	HIGH-SPEED COSINE	@COSQ	---	---	617
477	TANQ	HIGH-SPEED TANGENT	@TANQ	---	---	621
486	SCL2	SCALING 2	@SCL2	---	---	800
487	SCL3	SCALING 3	@SCL3	---	---	804
490	CMND	DELIVER COMMAND	@CMND	---	---	1056
498	MOVL	DOUBLE MOVE	@MOVL	---	---	334
499	MVNL	DOUBLE MOVE NOT	@MVNL	---	---	336
502	BCMP2	EXPANDED BLOCK COMPARE	@BCMP2	---	---	322
510	CJP	CONDITIONAL JUMP	---	---	---	232
511	CJPN	CONDITIONAL JUMP	---	---	---	232
512	FOR	FOR-NEXT LOOPS	---	---	---	238
513	NEXT	FOR-NEXT LOOPS	---	---	---	238
514	BREAK	BREAK LOOP	---	---	---	241
515	JMP0	MULTIPLE JUMP	---	---	---	236
516	JME0	MULTIPLE JUMP END	---	---	---	236
517	MILH	MULTI-INTERLOCK DIFFERENTIATION HOLD	---	---	---	214
518	MILR	MULTI-INTERLOCK DIFFERENTIATION RELEASE	---	---	---	214
519	MILC	MULTI-INTERLOCK CLEAR	---	---	---	214
520	NOT	NOT	---	---	---	180
521	UP	CONDITION ON	---	---	---	181
522	DOWN	CONDITION OFF	---	---	---	181
530	SETA	MULTIPLE BIT SET	@SETA	---	---	198
531	RSTA	MULTIPLE BIT RESET	@RSTA	---	---	198
532	SETB	SINGLE BIT SET	@SETB	---	!SETB	201
533	RSTB	SINGLE BIT RESET	@RSTB	---	!RSTB	201
534	OUTB	SINGLE BIT OUTPUT	@OUTB	---	!OUTB	204
540	TMHH	ONE-MS TIMER	---	---	---	253
541	TIMU	TENTH-MS TIMER	---	---	---	256
542	TIML	LONG TIMER	---	---	---	266
543	MTIM	MULTI-OUTPUT TIMER	---	---	---	269
544	TMUH	HUNDREDTH-MS TIMER	---	---	---	259
545	CNR	RESET TIMER/ COUNTER	@CNR	---	---	282
546	CNTX	COUNTER	---	---	---	275
547	CNRX	RESET TIMER/ COUNTER	---	---	---	282
548	CNTRX	REVERSIBLE COUNTER	---	---	---	278
550	TIMX	HUNDRED-MS TIMER	---	---	---	245
551	TIMHX	TEN-MS TIMER	---	---	---	249
552	TMHHX	ONE-MS TIMER	---	---	---	253
553	TIMLX	LONG TIMER	---	---	---	266

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
554	MTIMX	MULTI-OUTPUT TIMER	---	---	---	269
555	TTIMX	ACCUMULATIVE TIMER	---	---	---	262
556	TIMUX	TENTH-MS TIMER	---	---	---	256
557	TMUHX	HUNDREDTH-MS TIMER	---	---	---	259
560	MOVR	MOVE TO REGISTER	@MOVR	---	---	356
561	MOVRW	MOVE TIMER/ COUNTER PV TO REGISTER	@MOVRW	---	---	358
562	XCGL	DOUBLE DATA EXCHANGE	@XCGL	---	---	350
565	XFERC	BLOCK TRANSFER	@XFERC	---	---	1263
566	DISTC	SINGLE WORD DISTRIBUTE	@DISTC	---	---	1266
567	COLLC	DATA COLLECT	@COLLC	---	---	1269
568	MOVBC	MOVE BIT	@MOVBC	---	---	1273
570	ASLL	DOUBLE SHIFT LEFT	@ASLL	---	---	371
571	ASRL	DOUBLE SHIFT RIGHT	@ASRL	---	---	374
572	ROLL	DOUBLE ROTATE LEFT	@ROLL	---	---	378
573	RORL	DOUBLE ROTATE RIGHT	@RORL	---	---	381
574	RLNC	ROTATE LEFT WITHOUT CARRY	@RLNC	---	---	383
575	RRNC	ROTATE RIGHT WITHOUT CARRY	@RRNC	---	---	387
576	RLNL	DOUBLE ROTATE LEFT WITHOUT CARRY	@RLNL	---	---	385
577	RRNL	DOUBLE ROTATE RIGHT WITHOUT CARRY	@RRNL	---	---	388
578	NSFL	SHIFT N-BIT DATA LEFT	@NSFL	---	---	393
579	NSFR	SHIFT N-BIT DATA RIGHT	@NSFR	---	---	395
580	NASL	SHIFT N-BITS LEFT	@NASL	---	---	397
581	NASR	SHIFT N-BITS RIGHT	@NASR	---	---	403
582	NSLL	DOUBLE SHIFT N-BITS LEFT	@NSLL	---	---	400
583	NSRL	DOUBLE SHIFT N-BITS RIGHT	@NSRL	---	---	405
590	++	INCREMENT BINARY	@++	---	---	409
591	++L	DOUBLE INCREMENT BINARY	@++L	---	---	411
592	– –	DECREMENT BINARY	@– –	---	---	413
593	– –L	DOUBLE DECREMENT BINARY	@– –L	---	---	415
594	++B	INCREMENT BCD	@++B	---	---	417
595	++BL	DOUBLE INCREMENT BCD	@++BL	---	---	419
596	– –B	DECREMENT BCD	@– –B	---	---	421
597	– –BL	DOUBLE DECREMENT BCD	@– –BL	---	---	423
600	SIGN	16-BIT TO 32-BIT SIGNED BINARY	@SIGN	---	---	494
601	STR4	FOUR-DIGIT NUMBER TO ASCII	@STR4	---	---	534

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
602	STR8	EIGHT-DIGIT NUMBER TO ASCII	@STR8	---	---	537
603	STR16	SIXTEEN-DIGIT NUMBER TO ASCII	@STR16	---	---	539
604	NUM4	ASCII TO FOUR-DIGIT NUMBER	@NUM4	---	---	541
605	NUM8	ASCII TO EIGHT-DIGIT NUMBER	@NUM8	---	---	544
606	NUM16	ASCII TO SIXTEEN-DIGIT NUMBER	@NUM16	---	---	545
610	ANDL	DOUBLE LOGICAL AND	@ANDL	---	---	550
611	ORWL	DOUBLE LOGICAL OR	@ORWL	---	---	553
612	XORL	DOUBLE EXCLUSIVE OR	@XORL	---	---	557
613	XNRL	DOUBLE EXCLUSIVE NOR	@XNRL	---	---	560
614	COML	DOUBLE COMPLEMENT	@COML	---	---	564
620	ROTB	BINARY ROOT	@ROTB	---	---	565
621	BCNTC	BIT COUNTER	@BCNTC	---	---	1275
630	SSET	SET STACK	@SSET	---	---	703
631	DIM	DIMENSION RECORD TABLE	@DIM	---	---	715
632	PUSH	PUSH ONTO STACK	@PUSH	---	---	706
633	FIFO	FIRST IN FIRST OUT	@FIFO	---	---	709
634	LIFO	LAST IN FIRST OUT	@LIFO	---	---	712
635	SETR	SET RECORD LOCATION	@SETR	---	---	718
636	GETR	GET RECORD NUMBER	@GETR	---	---	720
637	SWAP	SWAP BYTES	@SWAP	---	---	725
638	SNUM	STACK SIZE READ	@SNUM	---	---	742
639	SREAD	STACK DATA READ	@SREAD	---	---	744
640	SWRIT	STACK DATA WRITE	@SWRIT	---	---	747
641	SINS	STACK DATA INSERT	@SINS	---	---	750
642	SDEL	STACK DATA DELETE	@SDEL	---	---	753
650	LEN$	STRING LENGTH	@LEN$	---	---	1235
652	LEFT$	GET STRING LEFT	@LEFT$	---	---	1226
653	RGHT$	GET STRING RIGHT	@RGHT$	---	---	1228
654	MID$	GET STRING MIDDLE	@MID$	---	---	1230
656	+$	CONCATENATE STRING	@+$	---	---	1223
657	INS$	INS$	@INS$	---	---	1246
658	DEL$	DELETE STRING	@DEL$	---	---	1240
660	FIND$	FIND IN STRING	@FIND$	---	---	1233
661	RPLC$	REPLACE IN STRING	@RPLC$	---	---	1237
664	MOV$	MOV STRING	@MOV$	---	---	1221
665	XCHG$	EXCHANGE STRING	@XCHG$	---	---	1242
666	CLR$	CLEAR STRING	@CLR$	---	---	1245
670	AND =$	AND STRING EQUALS	---	---	---	1250
670	LD =$	LOAD STRING EQUALS	---	---	---	1250
670	OR =$	OR STRING EQUALS	---	---	---	1250
671	AND <>$	AND STRING NOT EQUAL	---	---	---	1250

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
671	LD <>$	LOAD STRING NOT EQUAL	---	---	---	1250
671	OR <>$	OR STRING NOT EQUAL	---	---	---	1250
672	AND <$	AND STRING LESS THAN	---	---	---	1250
672	LD <$	LOAD STRING LESS THAN	---	---	---	1250
672	OR <$	OR STRING LESS THAN	---	---	---	1250
673	AND <=$	AND STRING LESS THAN OR EQUALS	---	---	---	1250
673	LD <=$	LOAD STRING LESS THAN OR EQUAL	---	---	---	1250
673	OR <=$	OR STRING LESS THAN OR EQUALS	---	---	---	1250
674	AND >$	AND STRING GREATER THAN	---	---	---	1250
674	LD >$	LOAD STRING GREATER THAN	---	---	---	1250
674	OR >$	OR STRING GREATER THAN	---	---	---	1250
675	AND >=$	AND STRING GREATER THAN OR EQUALS	---	---	---	1250
675	LD >=$	LOAD STRING GREATER THAN OR EQUALS	---	---	---	1250
675	OR >=$	OR STRING GREATER THAN OR EQUALS	---	---	---	1250
680	LMT	LIMIT CONTROL	@LMT	---	---	779
681	BAND	DEAD BAND CONTROL	@BAND	---	---	781
682	ZONE	DEAD ZONE CONTROL	@ZONE	---	---	784
685	TPO	TIME-PROPOR-TIONAL OUTPUT	---	---	---	787
690	MSKS	SET INTERRUPT MASK	@MSKS	---	---	839
691	CLI	CLEAR INTERRUPT	@CLI	---	---	851
692	MSKR	READ INTERRUPT MASK	@MSKR	---	---	846
693	DI	DISABLE INTERRUPTS	@DI	---	---	855
694	EI	ENABLE INTERRUPTS	---	---	---	858
700	FREAD	READ DATA FILE	@FREAD	---	---	1099
701	FWRIT	WRITE DATA FILE	@FWRIT	---	---	1106
704	TWRIT	WRITE TEXT TILE	@TWRIT	---	---	1113
720	EXPLT	EXPLICIT MESSAGE SEND	@EXPLT	---	---	1066
721	EGATR	EXPLICIT GET ATTRIBUTE	@EGATR	---	---	1074
722	ESATR	EXPLICIT SET ATTRIBUTE	@ESATR	---	---	1081
723	ECHRD	EXPLICIT WORD READ	@ECHRD	---	---	1087
724	ECHWR	EXPLICIT WORD CLEAR	@ECHWR	---	---	1091
730	CADD	CALENDAR ADD	@CADD	---	---	1122
731	CSUB	CALENDAR SUBTRACT	@CSUB	---	---	1126

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
735	DATE	CLOCK ADJUSTMENT	@DATE	---	---	1134
750	GSBS	GLOBAL SUBROUTINE CALL	@GSBS	---	---	824
751	GSBN	GLOBAL SUBROUTINE ENTRY	---	---	---	832
752	GRET	GLOBAL SUBROUTINE RETURN	---	---	---	835
801	BEND	BLOCK PROGRAM END	---	---	---	1191
802	IF	CONDITIONAL BRANCHING BLOCK	---	---	---	1196
802	IF	CONDITIONAL BRANCHING BLOCK	---	---	---	1196
802	IF NOT	CONDITIONAL BRANCHING BLOCK NOT	---	---	---	1196
803	ELSE	ELSE	---	---	---	1196
804	IEND	IF END	---	---	---	1196
805	WAIT	ONE CYCLE AND WAIT	---	---	---	1202
805	WAIT	ONE CYCLE AND WAIT	---	---	---	1202
805	WAIT NOT	ONE CYCLE AND WAIT NOT	---	---	---	1202
806	EXIT	CONDITIONAL BLOCK EXIT	---	---	---	1199
806	EXIT	CONDITIONAL BLOCK EXIT	---	---	---	1199
806	EXIT NOT	CONDITIONAL BLOCK EXIT NOT	---	---	---	1199
809	LOOP	LOOP	---	---	---	1215
810	LEND	LOOP END	---	---	---	1215
810	LEND	LOOP END	---	---	---	1215
810	LEND NOT	LOOP END NOT	---	---	---	1215
811	BPPS	BLOCK PROGRAM PAUSE	---	---	---	1193
812	BPRS	BLOCK PROGRAM RESTART	---	---	---	1193
813	TIMW	HUNDRED-MS TIMER WAIT	---	---	---	1206
814	CNTW	COUNTER WAIT	---	---	---	1209
815	TMHW	TEN-MS TIMER WAIT	---	---	---	1212
816	TIMWX	HUNDRED-MS TIMER WAIT	---	---	---	1206
817	TMHWX	TEN-MS TIMER WAIT	---	---	---	1212
818	CNTWX	COUNTER WAIT	---	---	---	1209
820	TKON	TASK ON	@TKON	---	---	1255
821	TKOF	TASK OFF	@TKOF	---	---	1258
840	PWR	EXPONENTIAL POWER	@PWR	---	---	635
841	FIXD	DOUBLE FLOATING TO 16-BIT BINARY	@FIXD	---	---	657
842	FIXLD	DOUBLE FLOATING TO 32-BIT BINARY	@FIXLD	---	---	658
843	DBL	16-BIT BINARY TO DOUBLE FLOATING	@DBL	---	---	660
844	DBLL	32-BIT BINARY TO DOUBLE FLOATING	@DBLL	---	---	661
845	+D	DOUBLE FLOATING-POINT ADD	@+D	---	---	663

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Immediate Refreshing Specification	Page
846	–D	DOUBLE FLOATING-POINT SUBTRACT	@–D	---	---	665
847	*D	DOUBLE FLOATING-POINT MULTIPLY	@*D	---	---	667
848	/D	DOUBLE FLOATING-POINT DIVIDE	@/D	---	---	669
849	RADD	DOUBLE DEGREES TO RADIANS	@RADD	---	---	671
850	DEGD	DOUBLE RADIANS TO DEGREES	@RADD	---	---	673
851	SIND	DOUBLE SINE	@SIND	---	---	674
852	COSD	DOUBLE COSINE	@COSD	---	---	676
853	TAND	DOUBLE TANGENT	@TAND	---	---	678
854	ASIND	DOUBLE ARC SINE	@ASIND	---	---	680
855	ACOSD	DOUBLE ARC COSINE	@ACOSD	---	---	682
856	ATAND	DOUBLE ARC TANGENT	@ATAND	---	---	684
857	SQRTD	DOUBLE SQUARE ROOT	@SQRTD	---	---	686
858	EXPD	DOUBLE EXPONENT	@EXPD	---	---	688
859	LOGD	DOUBLE LOGARITHM	@LOGD	---	---	690
860	PWRD	DOUBLE EXPONENTIAL POWER	@PWRD	---	---	692
880	INI	MODE CONTROL	@INI	---	---	864
881	PRV	HIGH-SPEED COUNTER PV READ	@PRV	---	---	868
882	CTBL	COMPARISON TABLE LOAD	@CTBL	---	---	878
883	PRV2	COUNTER FREQUENCY CONVERT	@PRV2	---	---	874
885	SPED	SPEED OUTPUT	@SPED	---	---	882
886	PULS	SET PULSES	@PULS	---	---	887
887	PLS2	PULSE OUTPUT	@PLS2	---	---	890
888	ACC	ACCELERATION CONTROL	@ACC	---	---	896
889	ORG	ORIGIN SEARCH	@ORG	---	---	903
891	PWN	PULSE WITH VARIABLE DUTY FACTOR	@PWN	---	---	906

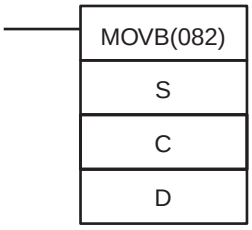
SECTION 3
Instructions

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

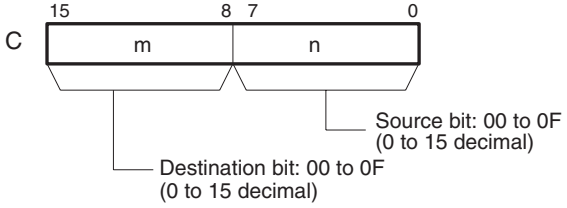
3-1 Notation and Layout of Instruction Descriptions

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

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

<table>
<tr><th colspan="2">Item</th><th>Contents</th></tr>
<tr><td colspan="2">Name and Mnemonic</td><td>The heading of each section consists of the name of the instruction followed by the mnemonic with the function code in parentheses. Example: MOVE BIT: MOVB(082)</td></tr>
<tr><td colspan="2">Purpose</td><td>The basic purpose of the instruction is described after the section heading.</td></tr>
<tr><td colspan="2">Ladder Symbol and Operand Names</td><td>The ladder symbol used to represent the instruction on the CX-Programmer is shown, as in the example for the MOVE BIT instruction given below. The name of each operand is also provided with the ladder symbol.
 

S: Source word or data
C: Control word
D: Destination word</td></tr>
<tr><td rowspan="2">Variations</td><td>Variations</td><td>The variations that can be used to control execution of the instruction under special conditions are given using the mnemonic form. Any variation that is not supported by an instruction is given as “Not supported.”
• Executed Each Cycle for ON Condition: The instruction is executed as long as it receives an ON execution condition.
• Executed Once for Upward Differentiation: The instruction is executed during the next cycle only after the execution condition changes from OFF to ON.
• Executed Once for Downward Differentiation: The instruction is executed during the next cycle only after the execution condition changes from ON to OFF.
• Always Executed: The instruction does not require an execution condition and is executed each cycle.
• Creates ON Condition....: The instruction is executed each cycle to create an execution condition for the next instruction.
<table><tr><td rowspan="3">Variations</td><td>Executed Each Cycle for ON Condition</td><td>MOVB(082)</td></tr><tr><td>Executed Once for Upward Differentiation</td><td>@MOVB(082)</td></tr><tr><td>Executed Once for Downward Differentiation</td><td>Not supported.</td></tr></table></td></tr>
<tr><td>Variations</td><td></td></tr>
<tr><td rowspan="2"></td><td>Immediate Refreshing Specification</td><td>Immediate refreshing can be specified for some instructions to refresh I/O when the instruction is executed. If immediate refreshing is supported, the specification is given using the mnemonic form. If immediate refreshing is not support by an instruction “Not supported” is given.
<table><tr><td>Immediate Refreshing Specification</td><td>Not supported.</td></tr></table></td></tr>
<tr><td colspan="2">Applicable Program Areas</td><td>The program areas in which the instruction can be used are specified. “OK” indicates the areas in which the instruction can be used.
<table><tr><th>Block program areas</th><th>Step program areas</th><th>Subroutines</th><th>Interrupt tasks</th></tr><tr><td>OK</td><td>OK</td><td>OK</td><td>OK</td></tr></table></td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operands</td><td>Where necessary, the meaning of words and bits used in specific operands, such as control words, is given.
 
</td></tr>
<tr><td>Operand Specifications</td><td>The memory areas addresses that can be used each operand are listed in a table like the following one. The letters used in the column headings on the left are the same as those used in the ladder symbol. “---” is used to indicate when an area cannot be specific for an operand.
<table>
<tr><th>Area</th><th>S</th><th>C</th><th>D</th></tr>
<tr><td>CIO Area</td><td colspan="3">CIO 0000 to CIO 6143</td></tr>
<tr><td>Work Area</td><td colspan="3">W000 to W511</td></tr>
<tr><td>Holding Bit Area</td><td colspan="3">H000 to H511</td></tr>
<tr><td>Auxiliary Bit Area</td><td colspan="2">A000 to A959</td><td>A448 to A959</td></tr>
<tr><td>Timer Area</td><td colspan="3">T0000 to T4095</td></tr>
<tr><td>Counter Area</td><td colspan="3">C0000 to C4095</td></tr>
<tr><td>DM Area</td><td colspan="3">D00000 to D32767</td></tr>
<tr><td>EM Area without bank</td><td colspan="3">E00000 to E32767</td></tr>
</table></td></tr>
<tr><td>Description</td><td>The function of the instruction and the operands used in the instruction are described.</td></tr>
<tr><td>Flags</td><td>The flags table indicates the status of the condition flags immediately after execution of the instruction. Any flags that are not listed are not affected by the instruction. “OFF” indicates that a flag is turned OFF immediately after execution of the instruction regardless of the results of executing the instruction.
<table>
<tr><th>Name</th><th>Label</th><th>Operation</th></tr>
<tr><td>Error Flag</td><td>ER</td><td>ON if control data is within ranges.
OFF in all other cases.</td></tr>
<tr><td>Equals Flag</td><td>=</td><td>OFF</td></tr>
<tr><td>Negative Flag</td><td>N</td><td>OFF</td></tr>
</table></td></tr>
<tr><td>Precautions</td><td>Special precautions required in using the instruction are provided. Be sure to read and follow these precautions.</td></tr>
<tr><td>Example</td><td>An example of using the instruction with specific operands is provided to further explain the function of the instruction.</td></tr>
</table>

Constants

Constants input for operands are given as listed below.

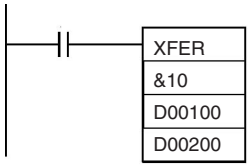
Operand Descriptions and Operand Specifications

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

- Operands Indicating Control Numbers (Except for Jump Numbers):
  The decimal form is given for control numbers, e.g., “0 to 1023” is given for the N operand for the SBS(091) instruction.

Examples

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



The input methods for constants for the 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. The CHG Key can then be pressed to rotate between hexadecimal (with # prefix), signed decimal (with +/–), and unsigned decimal (with & prefix).
Operands specifying numeric values (normally input as decimal)		
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	<>

Flag	CX-Programmer label	Programming Console label
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

3-2 Instruction Upgrades and New Instructions

This section lists the instruction upgrades for CS1 CPU Units with the -EV1 suffix and CS1-H/CJ1-H CPU Units.

3-2-1 Upgrades for CS1-H/CJ1-H CPU Units

New Instructions

The following instructions have been added to the CS1-H and CJ1-H CPU Units.

Sequence Output Instructions

SINGLE BIT SET, SETB(532)
SINGLE BIT RESET, RSTB(533)
SINGLE BIT OUTPUT, OUTB(534)

Data Comparison Instructions

AREA RANGE COMPARE, ZCP(088)
DOUBLE AREA RANGE COMPARE, ZCPL(116)

Floating Point Calculation and Conversion Instructions

Floating Point Data Comparison Instructions: =F, <>F, <F, <=F, >F, and >=F (329 to 334)

FLOATING POINT TO ASCII, FSTR(448)
ASCII TO FLOATING POINT, VAL(449)

Double-precision Floating Point Calculation and Conversion Instructions

Double-precision Comparison Instructions: =D, <>D, <D, <=D, >D, and >=D (335 to 340)

DOUBLE FLOATING TO 16-BIT BINARY, FIXD(841)
DOUBLE FLOATING TO 32-BIT BINARY, FIXLD(8420)
16-BIT BINARY TO DOUBLE FLOATING, DBL(843)
32-BIT BINARY TO DOUBLE FLOATING, DBLL(844)
DOUBLE FLOATING-POINT ADD, +D(845)
DOUBLE FLOATING-POINT SUBTRACT, –D(846)
DOUBLE FLOATING-POINT MULTIPLY, ∗D(847)
DOUBLE FLOATING-POINT DIVIDE, /D(848)
DOUBLE DEGREES TO RADIANS, RADD(849)
DOUBLE RADIANS TO DEGREES, DEGD(850)
DOUBLE SINE, SIND(851)
DOUBLE COSINE, COSD(852)
DOUBLE TANGENT, TAND(853)
DOUBLE ARC SINE, ASIND(854)
DOUBLE ARC COSINE, ACOSD(855)
DOUBLE ARC TANGENT, ATAND(856)
DOUBLE SQUARE ROOT, SQRTD(857)
DOUBLE EXPONENT, EXPD(858)
DOUBLE LOGARITHM, LOGD(859)
DOUBLE EXPONENTIAL POWER, PWRD(860)

Table Data Processing Instructions

STACK SIZE READ, SNUM(638)
STACK DATA READ, SREAD(639)
STACK DATA WRITE, SWRIT(640)
STACK DATA INSERT, SINS(641)
STACK DATA DELETE, SDEL(642)

Data Control Instructions

PID CONTROL WITH AUTOTUNING, PIDAT(191)

Subroutine Instructions

GLOBAL SUBROUTINE CALL, GSBS(750)
GLOBAL SUBROUTINE ENTRY, GSBN(751)
GLOBAL SUBROUTINE RETURN, GRET(752)

I/O Unit Instructions

CPU BUS UNIT I/O REFRESH, DLNK(226)

Other Instructions

SAVE CONDITION FLAGS, CCS(282)
LOAD CONDITION FLAGS, CCL(283)
CONVERT ADDRESS FROM CV, FRMCV(284)
CONVERT ADDRESS TO CV, TOCV(285)
DISABLE PERIPHERAL SERVICING, IOSP(287)
ENABLE PERIPHERAL SERVICING, IORS(288)

New Instructions

The following instructions have been upgraded for the CS1-H and CJ1-H CPU Units.

Special Math Instructions

ARITHMETIC PROCESS, APR(069)

Failure Diagnosis Instructions

FAILURE ALARM, FAL(006)
SEVERE FAILURE ALARM, FALS(007)

3-3 Sequence Input Instructions

3-3-1 LOAD: LD

Purpose

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

Ladder Symbol

Variations

Variations	**Restarts Logic and Creates ON Each Cycle Operand Bit is ON**	LD
	Restarts Logic and Creates ON Once for Upward Differentiation	@LD
	Restarts Logic and Creates ON Once for Downward Differentiation	%LD
Immediate Refreshing Specification (See note.)		!LD
Combined Variations	**Refreshes Input Bit, Restarts Logic, and Creates ON Once for Upward Differentiation (See note.)**	!@LD
	Refreshes Input Bit, Restarts Logic, and Creates ON Once for Downward Differentiation (See note.)	!%LD

Note Immediate refreshing is not supported by CS1D CPU Units for Duplex-CPU Systems.

Applicable Program Areas

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

Operand Specifications

Area	LD operand bit
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK0000 to TK0031
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, A1, A0
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	TR0 to TR15
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---

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

Description

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

LD is used in the following circumstances as an instruction for indicating a logical start.

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

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

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

Flags

There are no flags affected by this instruction.

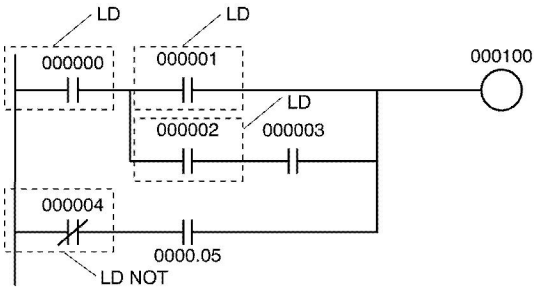
Precautions

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

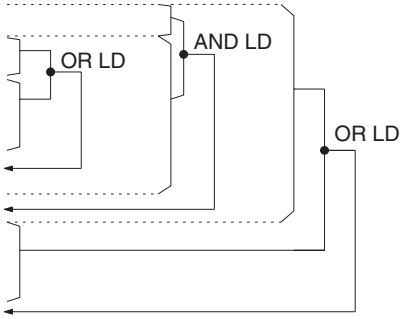
Immediate refreshing (!) can be specified for LD. An immediate refresh instruction updates the status of the input bit just before the instruction is executed for Basic Input Units (but not Basic Input Units on Slave Racks or for C200H Group 2 Multi-point Input Units).

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.

Example



Instruction	Operand
LD	000000
LD	000001
LD	000002
AND	000003
OR LD	---
AND LD	---
LD NOT	000004
AND	000005
OR LD	---
OUT	000100



3-3-2 LOAD NOT: LD NOT

Purpose

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

Ladder Symbol

Variations

Variations	**Restarts Logic and Creates ON Each Cycle Operand Bit is OFF**	LD NOT
	Restarts Logic and Creates ON Once for Upward Differentiation (See note 1.)	@LD NOT
	Restarts Logic and Creates ON Once for Downward Differentiation (See note 1.)	%LD NOT
Immediate Refreshing Specification (See note 2.)		!LD NOT
Combined Variations	**Refreshes Input Bit, Restarts Logic, and Creates ON Once for Upward Differentiation (See note 3.)**	!@LD NOT
	Refreshes Input Bit, Restarts Logic, and Creates ON Once for Downward Differentiation (See note 3.)	!%LD NOT

Note

1. The following variations are supported by only the CS1-H, CJ1-H, CJ1M, or CS1D CPU Units: @LD NOT, %LD NOT, !@LD NOT, and !%LD NOT.
2. Immediate refreshing is not supported by CS1D CPU Units for Duplex-CPU Systems.
3. Combined variations are supported by CS1D CPU Units for Single-CPU Systems and CS1-H, CJ1-H, and CJ1M CPU Units only.

Applicable Program Areas

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

Operand Specifications

Area	LD NOT bit operand
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK0000 to TK0031
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

LD NOT is used for the first normally closed bit from the bus bar, or for the first normally closed bit of a logic block. If there is no immediate refreshing specification, the specified bit in I/O memory is read and reversed. If there is an immediate refreshing specification, the status of the Basic Input Unit's input terminal is read, reversed, and used.

LD NOT is used in the following circumstances as an instruction for indicating a logical start.

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

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

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

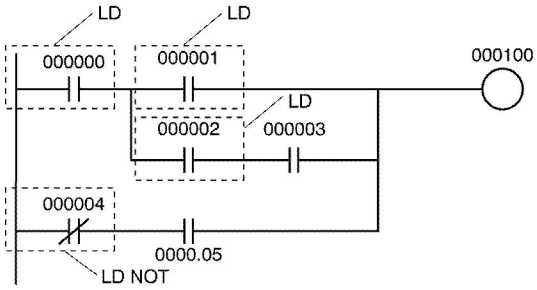
Flags

There are no flags affected by this instruction.

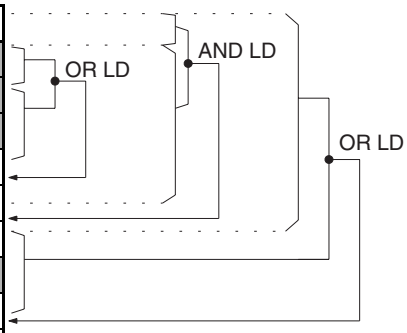
Precautions

Immediate refreshing (!) can be specified for LD NOT. An immediate refresh instruction updates the status of the input bit 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).

Example



Instruction	Operand
LD	000000
LD	000001
LD	000002
AND	000003
OR LD	---
AND LD	---
LD NOT	000004
AND	000005
OR LD	---
OUT	000100



3-3-3 AND: AND

Purpose

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

Ladder Symbol

Variations

Variations	**Creates ON Each Cycle AND Result is ON**	AND
	Creates ON Once for Upward Differentiation	@AND
	Creates ON Once for Downward Differentiation	%AND
Immediate Refreshing Specification (See note.)		!AND
Combined Variations	**Refreshes Input Bit and Creates ON Once for Upward Differentiation (See note.)**	!@AND
	Refreshes Input Bit and Creates ON Once for Downward Differentiation (See note.)	!%AND

Note Immediate refreshing is not supported by CS1D CPU Units for Duplex-CPU Systems.

Applicable Program Areas

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

Operand Specifications

Area	AND bit operand
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK0000 to TK0031
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

AND is used for a normally open bit connected in series. AND cannot be directly connected to the bus bar, and cannot be used at the beginning of a logic block. If there is no immediate refreshing specification, the specified bit in I/O memory is read. If there is an immediate refreshing specification, the status of the Basic Input Unit's input terminal is read.

Flags

There are no flags affected by this instruction.

Precautions

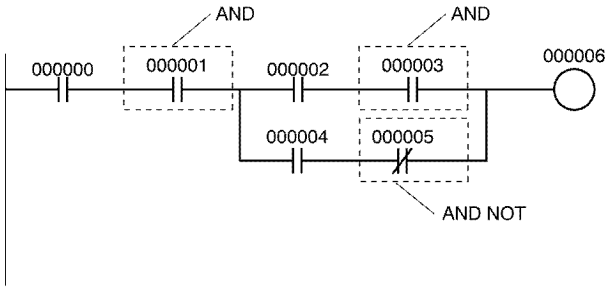
Differentiate up (@) or differentiate down (%) can be specified for AND. If differentiate up (@) is specified, the execution condition is turned ON for one cycle only after the status of the operand bit goes from OFF to ON. If differentiate down (%) is specified, the execution condition is turned ON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

Immediate refreshing (!) can be specified for AND. An immediate refresh instruction updates the status of the input bit 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.

AND cannot be used for addresses in the DM and EM Areas. Use AND TST(350) instead.

Example



Instruction	Operand
LD	000000
AND	000001
LD	000002
AND	000003
LD	000004
AND NOT	000005
OR LD	---
AND LD	---
OUT	000006

3-3-4 AND NOT: AND NOT

Purpose

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

Ladder Symbol

Variations

Variations	**Creates ON Each Cycle AND NOT Result is ON**	AND NOT
	Creates ON Once for Upward Differentiation (See note 1.)	@AND NOT
	Creates ON Once for Downward Differentiation (See note 1.)	%AND NOT
Immediate Refreshing Specification (See note 2.)		!AND NOT
Combined Variations	**Refreshes Input Bit and Creates ON Once for Upward Differentiation (See note 3.)**	!@AND NOT
	Refreshes Input Bit and Creates ON Once for Downward Differentiation (See note 3.)	!%AND NOT

Note

1. The following variations are supported by only the CS1-H, CJ1-H, CJ1M, or CS1D CPU Units: @AND NOT, %AND NOT, !@AND NOT, and !%AND NOT.
2. Immediate refreshing is not supported by CS1D CPU Units for Duplex-CPU Systems.
3. Combined variations are supported by CS1D CPU Units for Single-CPU Systems and CS1-H, CJ1-H, and CJ1M CPU Units only.

Applicable Program Areas

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

Operand Specifications

Area	AND NOT bit operand
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115

Area	AND NOT bit operand
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK0000 to TK0031
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

AND NOT is used for a normally closed bit connected in series. AND NOT cannot be directly connected to the bus bar, and cannot be used at the beginning of a logic block. If there is no immediate refreshing specification, the specified bit in I/O memory is read. If there is an immediate refreshing specification, the status the Basic Input Unit's input terminals is read.

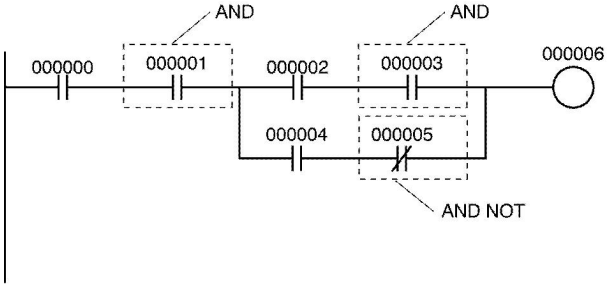
Flags

There are no flags affected by this instruction.

Precautions

Immediate refreshing (!) can be specified for AND NOT. An immediate refresh instruction updates the status of input bit just before the instruction is executed from Basic Input Units (but not for Basic Input Units on Slave Racks or for C200H Group 2 Multi-point Input Units).

Example



Instruction	Operand
LD	000000
AND	000001
LD	000002
AND	000003
LD	000004
AND NOT	000005

Instruction	Operand
OR LD	---
AND LD	---
OUT	000006

3-3-5 OR: OR

Purpose

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

Ladder Symbol

Variations

Variations	**Creates ON Each Cycle OR Result is ON**	OR
	Creates ON Once for Upward Differentiation	@OR
	Creates ON Once for Downward Differentiation	%OR
Immediate Refreshing Specification (See note.)		!OR
Combined Variations	**Refreshes Input Bit and Creates ON Once for Upward Differentiation (See note.)**	!@OR
	Refreshes Input Bit and Creates ON Once for Downward Differentiation (See note.)	!%OR

Note Immediate refreshing is not supported by CS1D CPU Units for Duplex-CPU Systems.

Applicable Program Areas

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

Operand Specifications

Area	OR bit operand
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK0000 to TK0031
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---

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

Description

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.

Flags

There are no flags affected by this instruction.

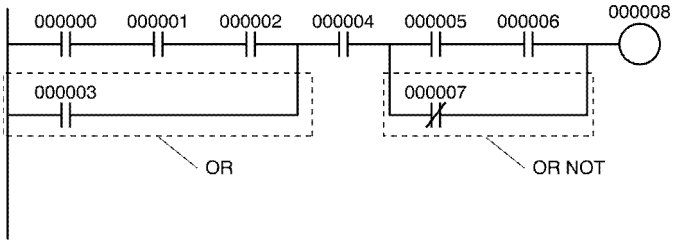
Precautions

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

Immediate refreshing (!) can be specified for OR. An immediate refresh instruction updates the status of the input bit 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.

Example



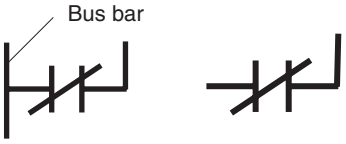
Instruction	Operand
LD	000000
AND	000001
AND	000002
OR	000003
AND	000004
LD	000005
AND	000006
OR NOT	000007
AND LD	---
OUT	000008

3-3-6 OR NOT: OR NOT

Purpose

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

Ladder Symbol



Variations

Variations	**Creates ON Each Cycle OR NOT Result is ON**	OR NOT
	Creates ON Once for Upward Differentiation (See note 1.)	@OR NOT
	Creates ON Once for Downward Differentiation (See note 1.)	%OR NOT
Immediate Refreshing Specification (See note 2.)		!OR NOT
Combined Variations	**Refreshes Input Bit and Creates ON Once for Upward Differentiation (See note 3.)**	!@OR NOT
	Refreshes Input Bit and Creates ON Once for Downward Differentiation (See note 3.)	!%OR NOT

Note

1. The following variations are supported by only the CS1-H, CJ1-H, CJ1M, or CS1D CPU Units: @OR NOT, %OR NOT, !@OR NOT, and !%OR NOT.
2. Immediate refreshing is not supported by CS1D CPU Units for Duplex-CPU Systems.
3. Combined variations are supported by CS1D CPU Units for Single-CPU Systems and CS1-H, CJ1-H, and CJ1M CPU Units only.

Applicable Program Areas

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

Operand Specifications

Area	OR NOT bit operand
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flag Area	TK0000 to TK0031
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, A1, A0
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	---
DM Area	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---

Area	OR NOT bit operand
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

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.

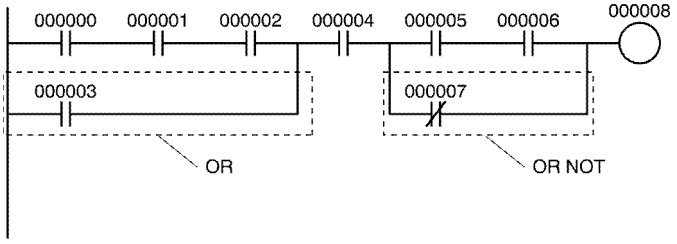
Flags

There are no flags affected by this instruction.

Precautions

Immediate refresh (!) can be specified for OR NOT. An immediate refresh instruction updates the status of the input bit just before the instruction is executed from a Basic Input Unit (but not Basic Input Units on Slave Racks or for C200H Group 2 Multi-point Input Units).

Example



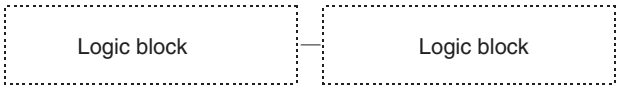
Instruction	Operand
LD	000000
AND	000001
AND	000002
OR	000003
AND	000004
LD	000005
AND	000006
OR NOT	000007
AND LD	---
OUT	000008

3-3-7 AND LOAD: AND LD

Purpose

Takes a logical AND between logic blocks.

Ladder Symbol



Variations

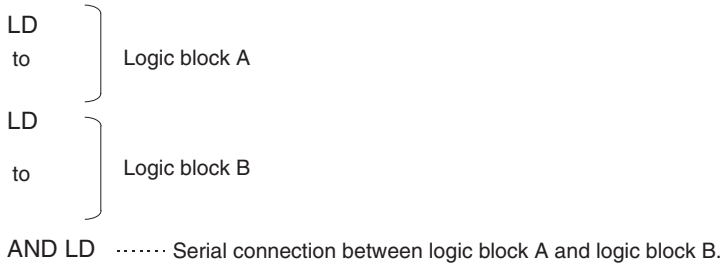
Variations	Creates ON Each Cycle AND Result is ON	AND LD
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

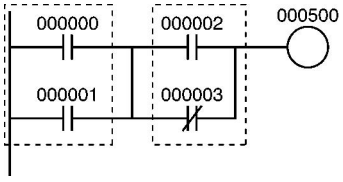
Description

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



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

In the following diagram, the two logic blocks are indicated by dotted lines. Studying this example shows that an ON execution condition will be produced when either of the execution conditions in the left logic block is ON (i.e., when either CIO 000000 or CIO 000001 is ON) **and** either of the execution conditions in the right logic block is ON (i.e., when either CIO 000002 is ON or CIO 000003 is OFF).



Flags

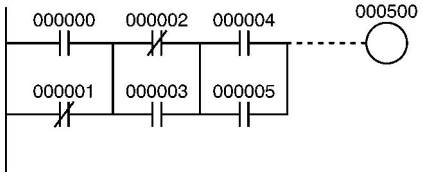
There are no flags affected by this instruction.

Precautions

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

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

Example



Coding Example (1)

Instruction	Operand
LD	000000
OR NOT	000001
LD NOT	000002
OR	000003
AND LD	---
LD	000004
OR	000005

Instruction	Operand
AND LD	---
. .	. .
OUT	000500

Coding Example (2)

Instruction	Operand
LD	000000
OR NOT	000001
LD NOT	000002
OR	000003
LD	000004
OR	000005
. .	. .
AND LD	---
AND LD	---
. .	. .
OUT	000500

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.

Coding

Address	Instruction	Operand
000000	LD	000000
000001	OR	000001
000002	LD	000002
000003	OR NOT	000003
000004	AND LD	---
000005	OUT	000500

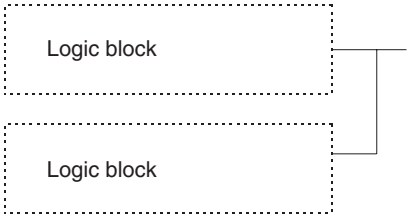
Second LD: Used for first bit of next block connected in series to previous block.

3-3-8 OR LOAD: OR LD

Purpose

Takes a logical OR between logic blocks.

Ladder Symbol



Variations

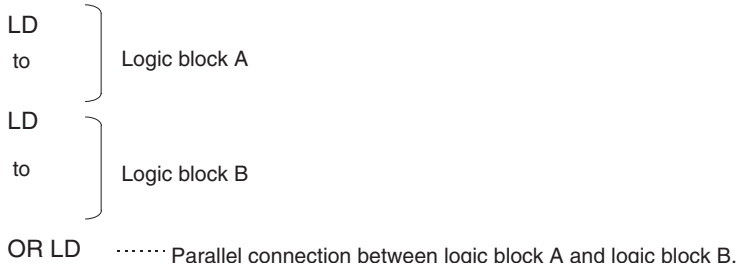
Variations	Creates ON Each Cycle AND Result is ON	OR LD
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

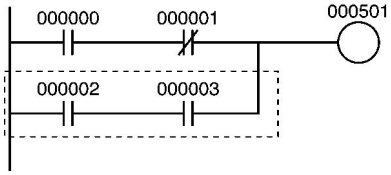
Description

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



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

The following diagram requires an OR LOAD instruction between the top logic block and the bottom logic block. An ON execution condition would be produced either when CIO 000000 is ON and CIO 000001 is OFF or when CIO 000002 and CIO 000003 are both ON. The operation of and mnemonic code for the OR LOAD instruction is exactly the same as those for a AND LOAD instruction except that the current execution condition is ORed with the last unused execution condition.



Flags

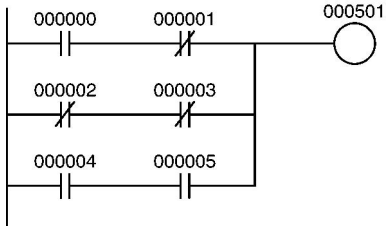
There are no flags affected by this instruction.

Precautions

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

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

Example



Coding Example (1)

Instruction	Operand
LD	000000
AND NOT	000001
LD NOT	000002
AND NOT	000003
OR LD	---
LD	000004
AND	000005
OR LD	---
. .	. .
OUT	000501

Coding Example (2)

Instruction	Operand
LD	000000
AND NOT	000001
LD NOT	000002
AND NOT	000003
LD	000004
AND	000005
. .	. .
OR LD	---
OR LD	---
. .	. .
OUT	000501

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.

Coding

Address	Instruction	Operand
000100	LD	000000
000101	AND NOT	000001
000102	LD	000002
000103	AND	000003
000104	OR LD	---
000105	OUT	000501

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

3-3-9 Differentiated and Immediate Refreshing Instructions

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

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

The I/O timing for data handled by instructions differs for ordinary and differentiated instructions, immediate refreshing instructions, and immediate refreshing differentiated instructions.

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

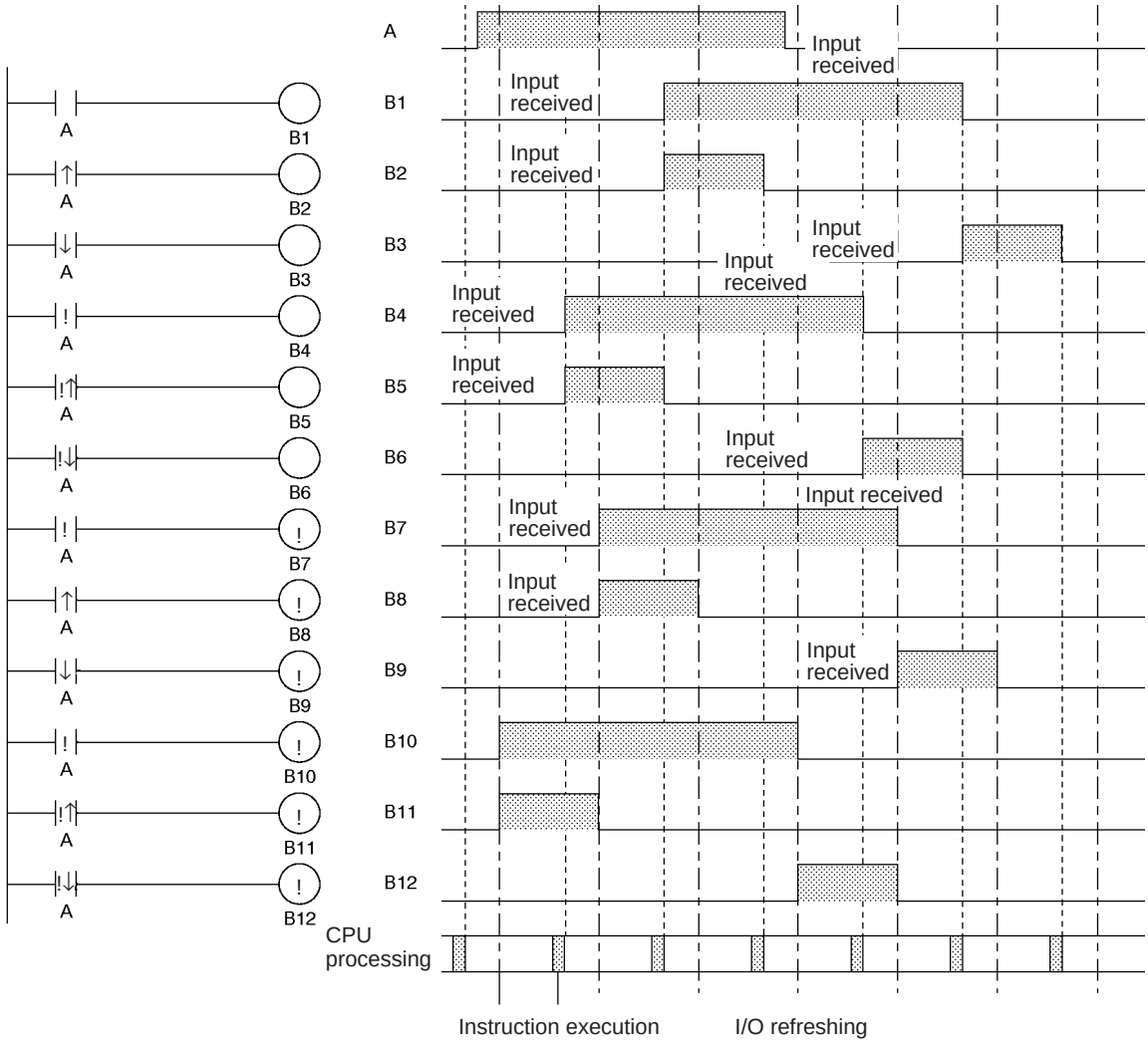
In addition to the above I/O refreshing, an immediate refresh instruction exchanges data with the I/O Unit for those words that are accessed by the instruction. An immediate refresh instruction refreshes eight bits simultaneously (leftmost or rightmost eight bits) in addition to the specified bit.

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.	

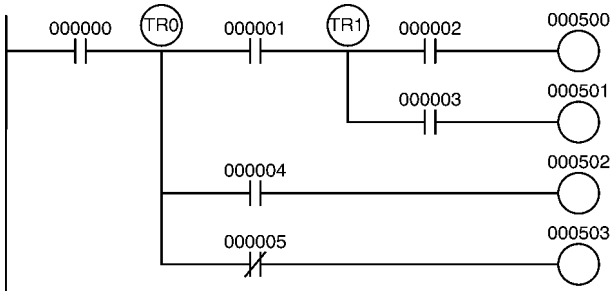
3-3-10 Operation Timing for I/O Instructions

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



3-3-11 TR Bits

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

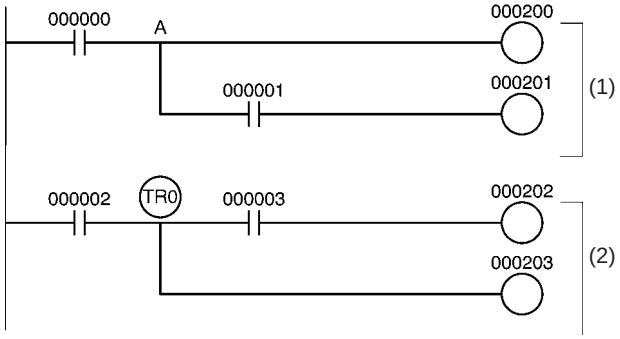


Address	Instruction	Operands
000000	LD	000000
000001	OUT	TR0
000002	AND	000001
000003	OUT	TR1
000004	AND	000002
000005	OUT	000500
000006	LD	TR1
000007	AND	000003
000008	OUT	000501
000009	LD	TR0
000010	AND	000004
000011	OUT	000502
000012	LD	TR0
000013	AND NOT	000005
000014	OUT	000503

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.



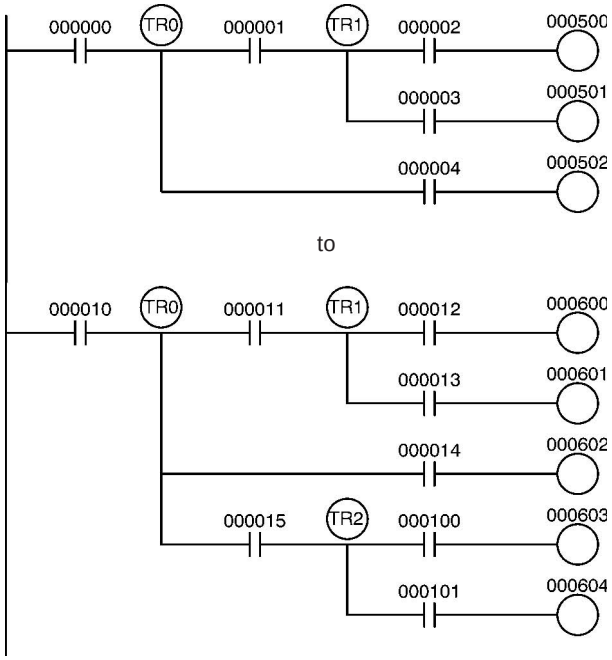
In instruction block (1), the ON/OFF status at point A is the same as for output CIO 00200, so AND 000001 and OUT 000201 can be coded without requiring a TR bit. In instruction block (2), the status of the branching point and that of output CIO 000202 are not necessarily the same, so a TR bit must be used. In this case, the number of steps in the program could be reduced by using instruction block (1) in place of instruction block (2).

TR0 to TR15 Considerations

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

TR0 to TR15 output Duplication

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



3-3-12 NOT: NOT(520)

Purpose

Reverses the execution condition.

Ladder Symbol

Variations

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

Applicable Program Areas

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

Description

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

Flags

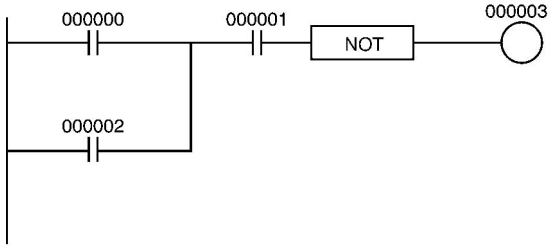
There are no flags affected by NOT(520).

Precautions

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

Example

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



The following table shows the operation of this program section.

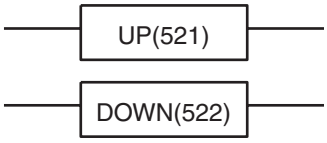
Input bit status			Output bit status
CIO 000000	**CIO 000001**	**CIO 000002**	**CIO 000003**
1	1	1	0
1	1	0	0
1	0	1	1
0	1	1	0
1	0	0	1
0	1	0	1
0	0	1	1
0	0	0	1

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

Purpose

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

Ladder Symbols



Variations

Variations	Creates ON Once for Upward Differentiation	UP(521)
Immediate Refreshing Specification		Not supported

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

Applicable Program Areas

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

Description

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

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

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

Flags

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

Precautions

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

The operation of UP(521) and DOWN(522) depends on the execution condition for the instruction as well as the execution condition for the program section when it is programmed in an interlocked program section, a jumped

program section, or a subroutine. Refer to *3-5-4 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003), 3-5-6 JUMP and JUMP END: JMP(004) and JME(005)*, and *3-20 Interrupt Control Instructions* for details.

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

The operation of UP(521) 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) 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.*

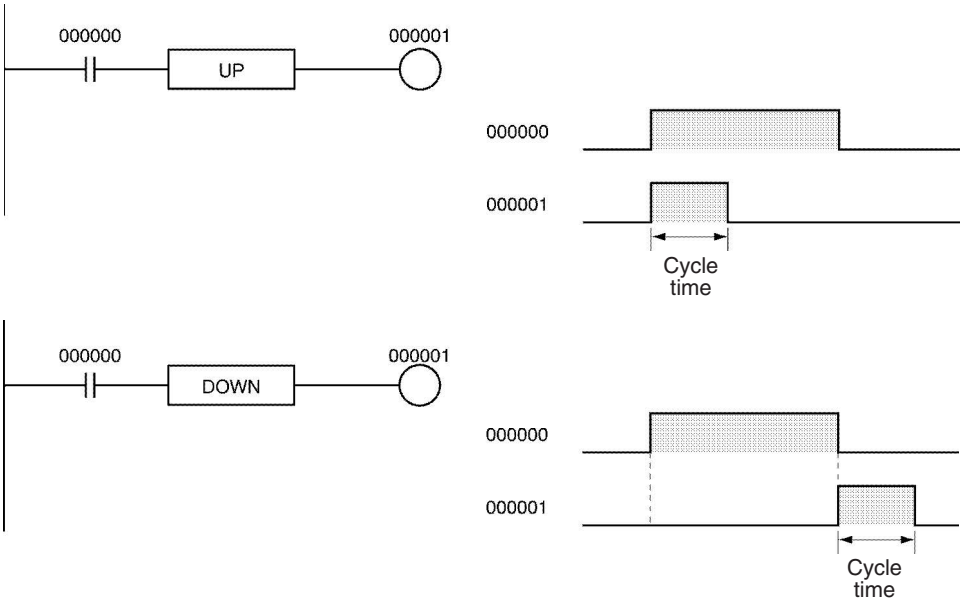
Observe the following precaution when using UP(521) in a subroutine.

The operation of UP(521) 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) in a function block definition. For details, refer to information on SBS(091).

Examples

When CIO 000000 goes from OFF to ON in the following example, CIO 000001 is turned ON for just one cycle.



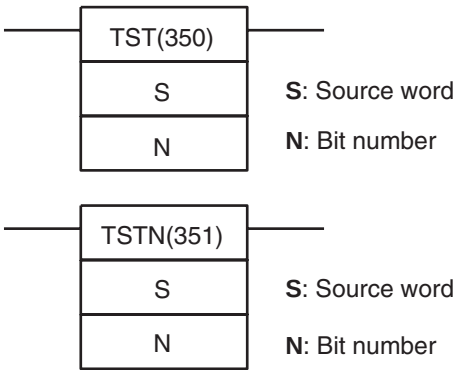
3-3-14 BIT TEST: TST(350) and TSTN(351)

Purpose

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

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

Ladder Symbols



Variations

Variations	Executed Each Cycle	TST(350)
Immediate Refreshing Specification		Not supported

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

Applicable Program Areas

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

Operands

N: Bit number

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

Operand Specifications

Area	S	N
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	#0000 to #000F (binary) or &0 to &15
Data Registers	DR0 to DR15	

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

Description

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

LD TSTN(351), AND TSTN(351), and OR TSTN(351) can be used in the program like LD NOT, AND NOT, and OR NOT; the execution condition is OFF when the specified bit in the specified word is ON and ON when the bit is OFF. Unlike LD NOT, AND NOT, and OR NOT, bits in the DM and EM areas can be used as operands in TSTN(351).

Flags

Name	Label	Operation
Error Flag	ER	OFF or unchanged (See note.)
Equals Flag	=	OFF or unchanged (See note.)
Negative Flag	N	OFF or unchanged (See note.)

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

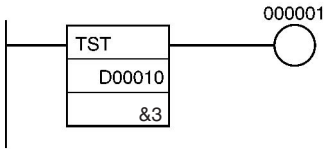
Precautions

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

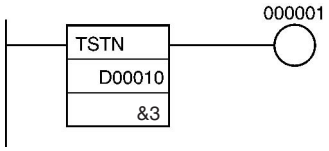
Examples

LD TST(350) and LD TSTN(351)

In the following example, CIO 000001 is turned ON when bit 3 of D00010 is ON.

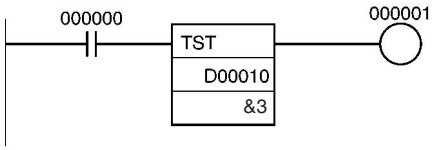


In the following example, CIO 000001 is turned ON when bit 3 of D00010 is OFF.

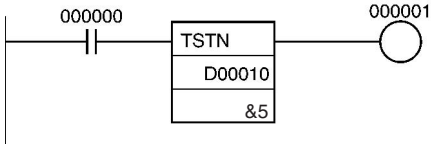


AND TST(350) and AND TSTN(351)

In the following example, CIO 000001 is turned ON when CIO 000000 and bit 3 of D00010 are both ON.

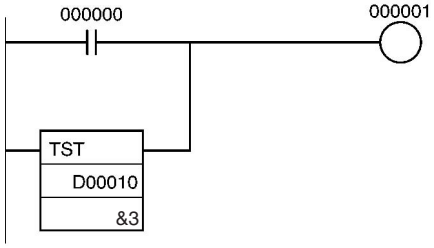


In the following example, CIO 000001 is turned ON when CIO 000000 is ON and bit 5 of D00010 is OFF.

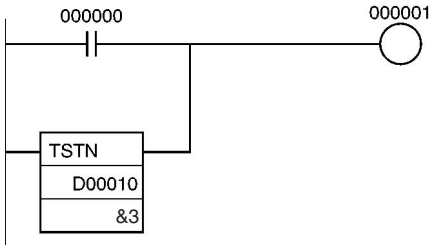


OR TST(350) and OR TSTN(351)

In the following example, CIO 000001 is turned ON when CIO 000000 or bit 3 of D00010 is ON.



In the following example, CIO 000001 is turned ON when CIO 000000 is ON or bit 3 of D00010 is OFF.



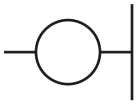
3-4 Sequence Output Instructions

3-4-1 OUTPUT: OUT

Purpose

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

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	OUT
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification (See note.)		!OUT

Note Immediate refreshing is not supported by CS1D CPU Units.

Applicable Program Areas

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

Operand Specifications

Area	OUT bit operand
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A44800 to A95915
Timer Area	---
Counter Area	---
TR Area	TR0 to TR15
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to ,IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

If there is no immediate refreshing specification, the status of the execution condition (power flow) is 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.

Flags

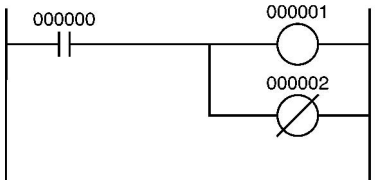
There are no flags affected by this instruction.

Precautions

Immediate refreshing (!) can be specified for OUT and OUT NOT. An immediate refresh instruction updates the status of the output terminal 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.

OUT cannot be used for addresses in the DM and EM Areas. Use OUTB(534) instead.

Example



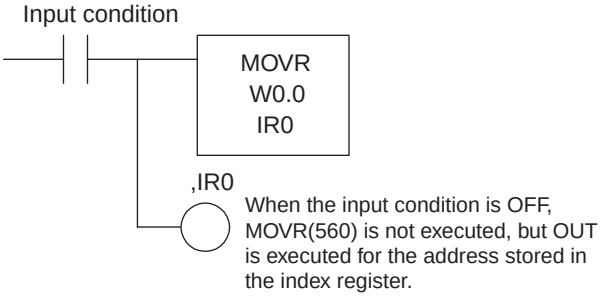
Instruction	Operand
LD	000000
OUT	000001
OUT NOT	000002

Note 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.

Note Precaution for Index Registers

OUT is executed even when the input condition turns OFF. Be particularly careful when programming OUT using an indirect index register address.

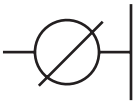


3-4-2 OUTPUT NOT: OUT NOT

Purpose

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

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	OUT NOT
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification (See note.)		!OUT NOT

Note Immediate refreshing is not supported by CS1D CPU Units.

Applicable Program Areas

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

Operand Specifications

Area	OUT bit operand
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115

Area	OUT bit operand
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A44800 to A95915
Timer Area	---
Counter Area	---
TR Area	TR0 to TR15
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to ,IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

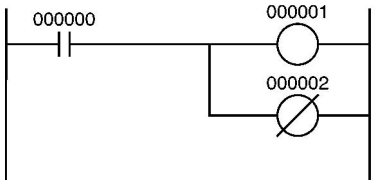
Description

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

Flags

There are no flags affected by this instruction.

Example



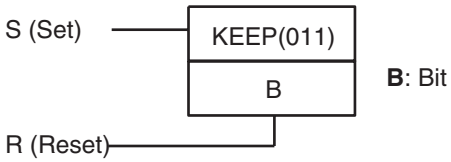
Instruction	Operand
LD	000000
OUT	000001
OUT NOT	000002

3-4-3 KEEP: KEEP(011)

Purpose

Operates as a latching relay.

Ladder Symbol



Variations

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

Note Immediate refreshing is not supported by CS1D CPU Units.

Applicable Program Areas

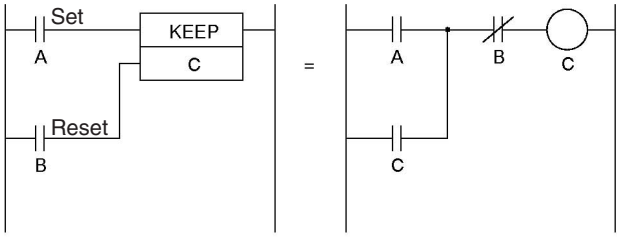
Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

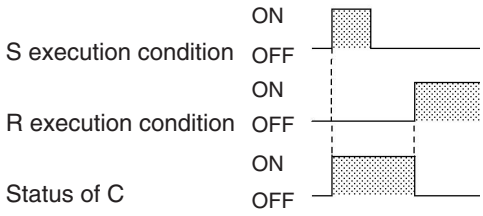
Operand Specifications

Area	B
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A44800 to A95915
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15

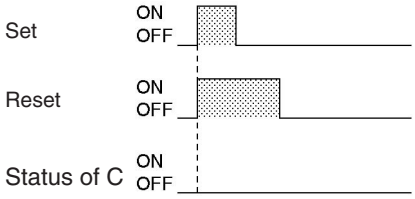
Description

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

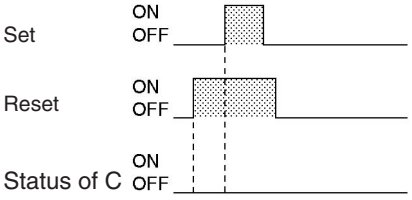




If S and R are ON simultaneously, the reset input takes precedence.

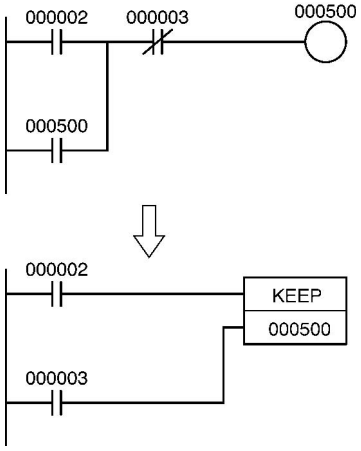


The set input (S) cannot be received while R is ON.

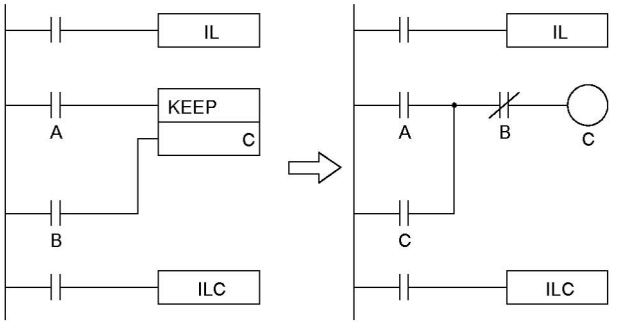


KEEP(011) has an immediate refreshing variation (!KEEP(011)). When an external output bit has been specified for B in a !KEEP(011) instruction, any changes to B will be refreshed when !KEEP(011) is executed and reflected immediately in the output bit. (The changes will not be reflected immediately if 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.)

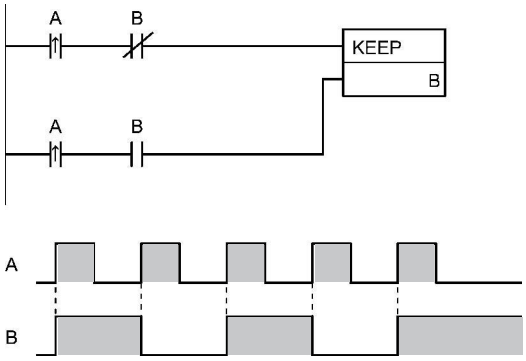
KEEP(011) operates like the self-maintaining bit, but a self-maintaining bit programmed with KEEP(011) requires one less instruction.



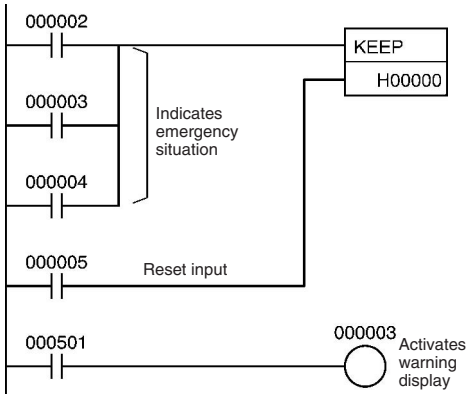
Self-maintaining bits programmed with KEEP(011) will maintain status even in an interlock program section, unlike the self-maintaining bit programmed without KEEP(011).



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



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



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

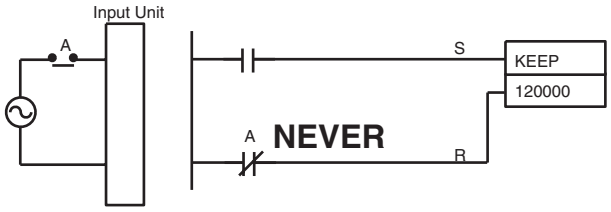
Flags

No flags are affected by KEEP(011).

Precautions

Never use an input bit in a normally closed condition on the reset (R) for KEEP(011) when the input device uses an AC power supply. The delay in shutting down the PLC's DC power supply (relative to the AC power supply to

the input device) can cause the operand bit of KEEP(011) to be reset. This situation is shown below.

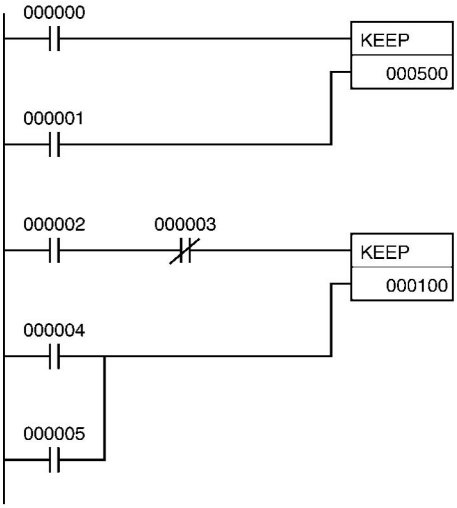


The operands for KEEP(011) are input in a different order in ladder diagrams and mnemonic code.
Ladder diagram order: Set input → KEEP(011) → Reset input
Mnemonic code order: Set input → Reset input → KEEP(011)

Example

When CIO 000000 goes ON in the following example, CIO 00500 is turned ON. CIO 00500 remains ON until CIO 000001 goes ON.

When CIO 000002 goes ON and CIO 000003 goes OFF in the following example, CIO 00100 is turned ON. CIO 00100 remains ON until CIO 000004 or CIO 000005 goes ON.



Coding

Address	Instruction	Operand
000100	LD	000000
000101	LD	000001
000102	KEEP (011)	000500
000103	LD	000002
000104	AND NOT	000003
000105	LD	000004
000106	OR	000005
000107	KEEP (011)	000100

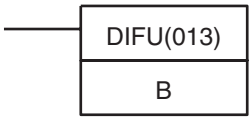
Note KEEP(011) is input in different orders on in ladder and mnemonic form. In ladder form, input the set input, KEEP(011), and then the reset input. In mnemonic form, input the set input, the reset input, and then KEEP(011).

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

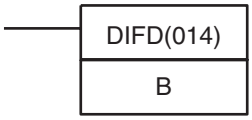
Purpose

DIFU(013) turns the designated bit ON for one cycle when the execution condition goes from OFF to ON (rising edge).
DIFD(014) turns the designated bit ON for one cycle when the execution condition goes from ON to OFF (falling edge).

Ladder Symbols



B: Bit



B: Bit

Variations

Variations	**Executed Each Cycle for ON Condition**	Not supported
	Executed Once for Upward Differentiation	DIFU(013)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification (See note.)		!DIFU(013)

Note Immediate refreshing is not supported by CS1D CPU Units.

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

Note Immediate refreshing is not supported by CS1D CPU Units.

Applicable Program Areas

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

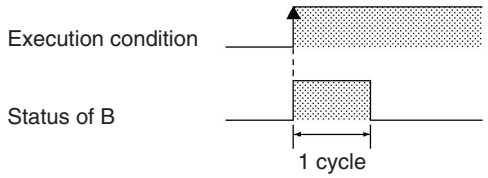
Operand Specifications

Area	B
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A44800 to A95915
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---

Area	B
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to ,15–(– –) IR

Description

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



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

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

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

Flags

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

Precautions

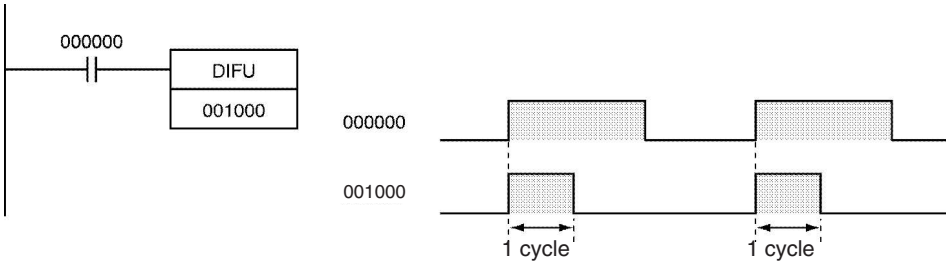
The operation of DIFU(013) or DIFD(014) depends on the execution condition for the instruction itself as well as the execution condition for the program section when it is programmed in an interlocked program section, a jumped program section, or a subroutine. Refer to *3-5-4 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003), 3-5-6 JUMP and JUMP END: JMP(004) and JME(005)*, and *3-20 Interrupt Control Instructions* for details.

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

Examples

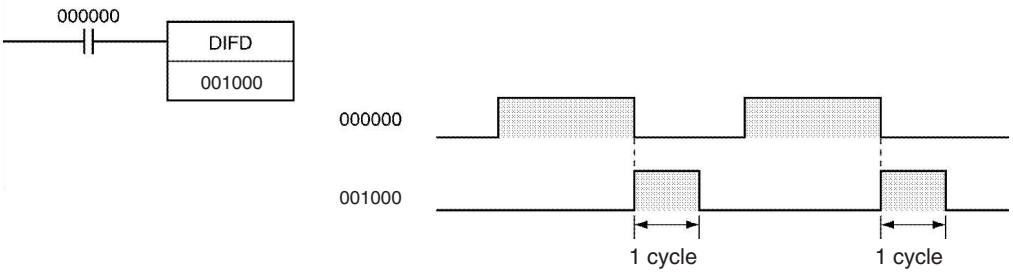
Operation of DIFU(013)

When CIO 000000 goes from OFF to ON in the following example, CIO 001000 is turned ON for one cycle.



Operation of DIFD(014)

When CIO 000000 goes from ON to OFF in the following example, CIO 001000 is turned ON for one cycle.



Note Observe the following precaution when using DIFU(013) in a function block definition.

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 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.*

Observe the following precaution when using DIFU(013) in a subroutine.

The operation of DIFU(013) will not be consistent if the same subroutine is executed more than once in the same cycle.

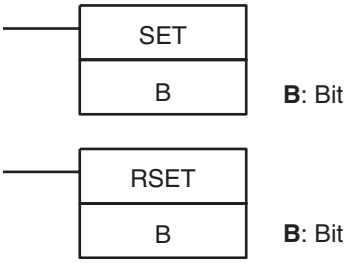
An subroutine will not be executed while the input condition for the subroutine is OFF. Caution is thus required when using DIFU(013) in a function block definition. For details, refer to information on SBS(091).

3-4-5 SET and RESET: SET and RSET

Purpose

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

Ladder Symbols



B: Bit

B: Bit

Variations

Variations	**Executed Each Cycle for ON Condition**	SET
	Executed Once for Upward Differentiation	@SET
	Executed Once for Downward Differentiation	%SET
Immediate Refreshing Specification (See note.)		!SET
Combined variations	**Executed Once and Bit Refreshed Immediately for Upward Differentiation (See note.)**	!@SET
	Executed Once and Bit Refreshed Immediately for Downward Differentiation (See note.)	!%SET

Note Immediate refreshing is not supported by CS1D CPU Units.

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

Note Immediate refreshing is not supported by CS1D CPU Units.

Applicable Program Areas

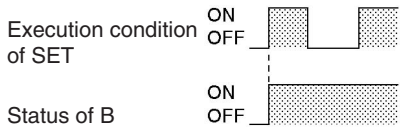
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

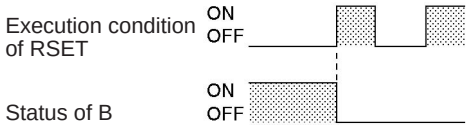
Area	B
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A44800 to A95915
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to ,–(– –) IR15

Description

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



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



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

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

Flags

No flags are affected by SET and RSET.

Precautions

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

When SET or RSET is programmed between IL(002) and ILC(003) or JMP(004) and JME(005), the status of the specified bit will not be changed if the program section is interlocked or jumped.

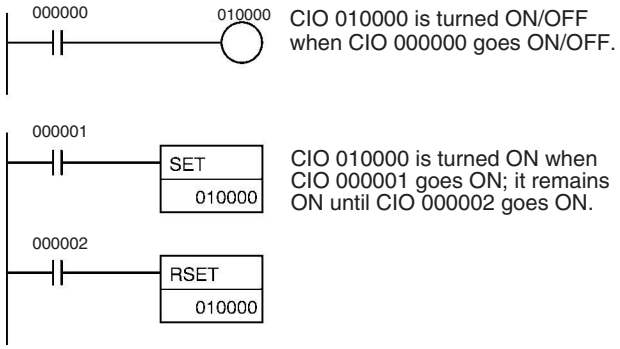
Note SET cannot be used for addresses in the DM and EM Areas. Use SETB(531) instead.

Note RSET cannot be used for addresses in the DM and EM Areas. Use RSTB(533) instead.

Example

Differences between OUT/OUT NOT and SET/RSET

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

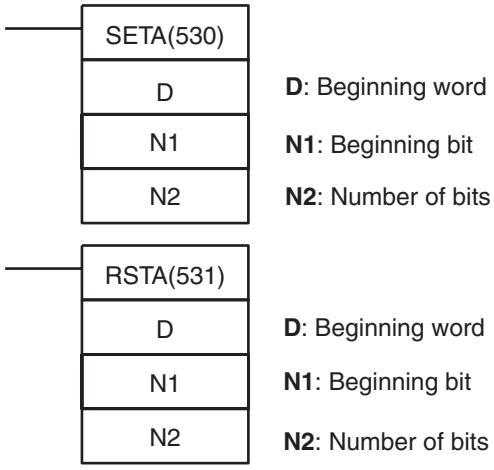


3-4-6 MULTIPLE BIT SET/RESET: SETA(530)/RSTA(531)

Purpose

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

Ladder Symbols



Variations

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

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

Applicable Program Areas

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

Operands

D: Beginning Word

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

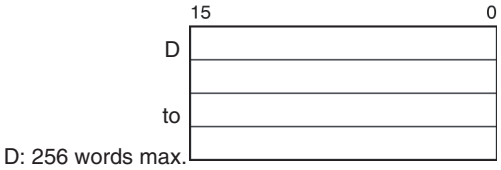
N1: Beginning Bit

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

N2: Number of Bits

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

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



Operand Specifications

Area	D	N1	N2
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A448 to A959	A000 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0000 to #000F (binary) or &0 to &15	#0000 to #FFFF (binary) or &0 to &65535
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

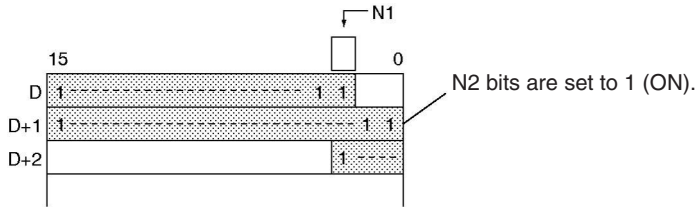
Description

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

Operation of SETA(530)

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

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

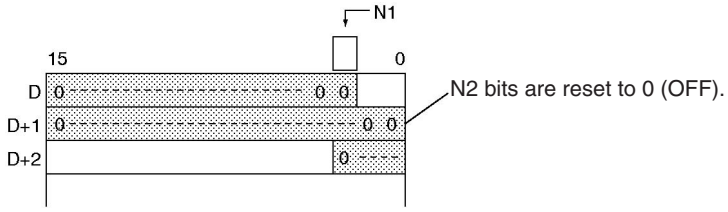


SETA(530) can be used to turn ON bits in data areas that are normally accessed by words only, such as the DM and EM areas.

Operation of RSTA(531)

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

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



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

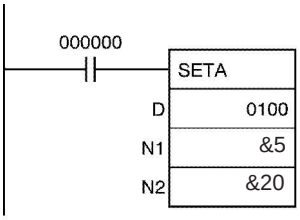
Flags

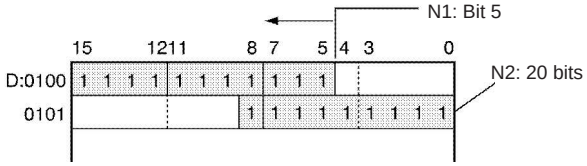
Name	Label	Operation
Error Flag	ER	ON if N1 is not within the specified range of 0000 to 000F. OFF in all other cases.

Examples

SETA(530) Example

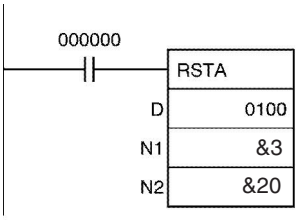
When CIO 000000 is turned ON in the following example, the 20 bits (0014 hexadecimal) beginning with bit 5 of CIO 0100 are turned ON.

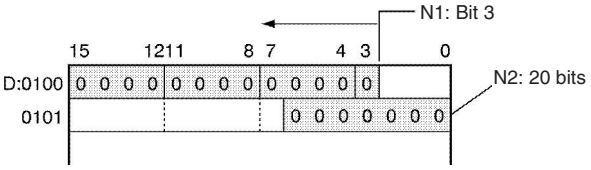




RSTA(531) Example

When CIO 000000 is turned ON in the following example, the 20 bits (0014 hexadecimal) beginning with bit 3 of CIO 0100 are turned OFF.





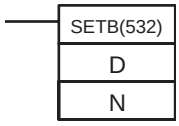
3-4-7 SINGLE BIT SET/RESET: SETB(532)/RSTB(533)

Purpose

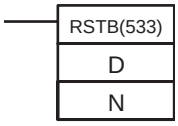
SETB(532) turns ON the specified bit.
RSTB(533) turns OFF the specified bit.

These instructions are supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbols



D: Word address
N: Bit number



D: Word address
N: Bit number

Variations

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

Note Immediate refreshing is not supported by CS1D CPU Units.

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

Note Immediate refreshing is not supported by CS1D CPU Units.

Applicable Program Areas

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

Operands

D: Word Address

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

N: Beginning Bit

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

Operand Specifications

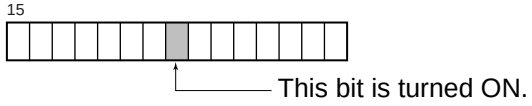
Area	D	N
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	A000 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	#0000 to #000F (binary) or &0 to &15
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15	

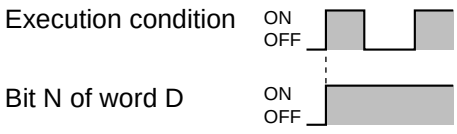
Description

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

Operation of SETB(532)

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



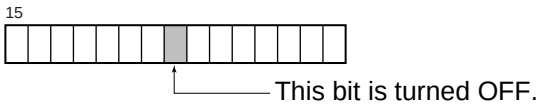


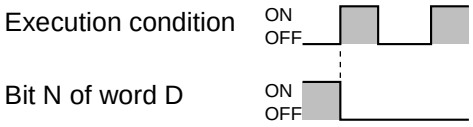
Bits turned ON by SETB(532) can be turned OFF by any other instruction, not just RSTB(533).

SETB(532) is supported by CS1-H, CJ1-H, and CJ1M CPU Units only.

Operation of RSTB(533)

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





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

RSTB(533) is supported by CS1-H, CJ1-H, and CJ1M CPU Units only.

Flags

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

Precautions

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

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

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

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

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

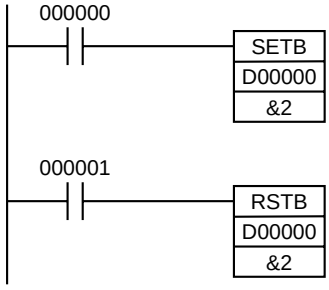
1. The instructions operate in the same way when the specified bit is in the CIO, W, H, or A Area.
2. The SETB(532) and RSTB(533) instructions can control bits in the DM and EM Areas, unlike SET and RSET.

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

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

1. The SETB(532) and RSTB(533) instructions change the status of the specified bit only when their execution condition is ON. These instructions have no effect on the status of the specified bit when their execution condition is OFF.

2. The OUTB(534) instruction turns ON the specified bit when its execution condition is ON and turns OFF the specified bit when its execution condition is OFF.
3. The set and reset inputs for a KEEP(011) instruction must be programmed with the instruction, but the SETB(532) and RSTB(533) instructions can be programmed completely independently. Furthermore, the same bit may be used as the operand in any number of SETB(532) and RSTB(533) instructions.



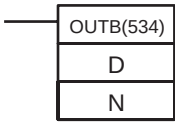
3-4-8 SINGLE BIT OUTPUT: OUTB(534)

Purpose

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

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbols



D: Word address
N: Bit number

Variations

Variations		
	Executed Each Cycle for ON Condition	OUTB(534)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification (See note.)		!OUTB(534)

Note Immediate refreshing is not supported by CS1D CPU Units.

Applicable Program Areas

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

Operands

D: Word Address

Specifies the word containing the bit to be controlled.

N: Beginning Bit

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

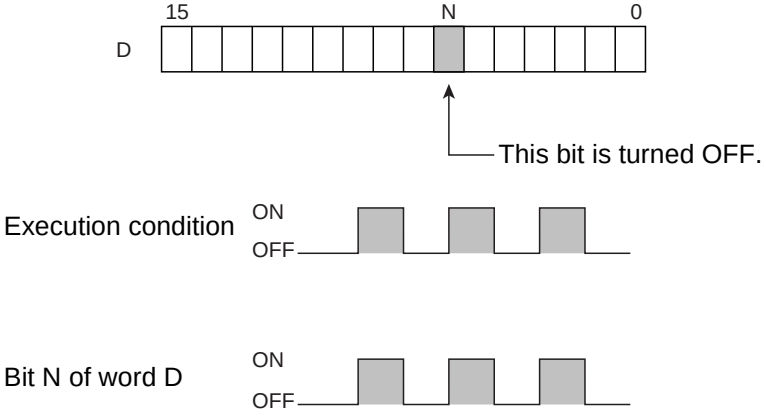
Operand Specifications

Area	D	N
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	A000 to A959

Area	D	N
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	#0000 to #000F (binary) or &0 to &15
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15	

Description

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.

OUTB(534) is supported by CS1-H, CJ1-H, and CJ1M CPU Units only.

Flags

There are no flags affected by this instruction.

Precautions

Immediate refreshing (!OUTB(534)) can be specified. An immediate refresh instruction updates the status of the output terminal just after the instruction is executed on an output bit allocated to a Basic Output Unit (but not for C200H Group 2 Multi-point Output Units or Basic Output Units on Slave Racks), at

the same time as it writes the status of the execution condition (power flow) to the specified output bit in I/O memory.

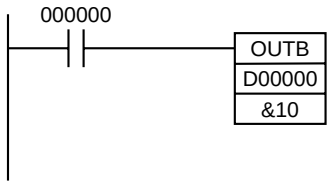
When OUTB(534) is programmed between IL(002) and ILC(003), the specified bit will be turned OFF if the program section is interlocked. (This is the same as an OUT instruction in an interlocked program section.)

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

Note 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.

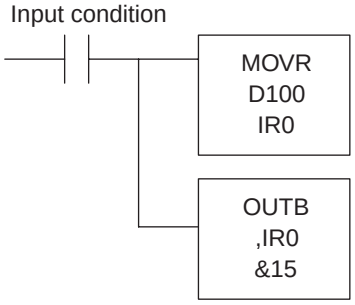
Example



Bit 10 of D00000 is turned OFF when CIO 000000 is OFF.

Note Precaution for Index Registers

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.

3-5 Sequence Control Instructions

3-5-1 END: END(001)

Purpose

Indicates the end of a program.

Ladder Symbol

Variations

Variations	Executed Each Cycle for ON Condition	END(001)
Immediate Refreshing Specification		Not supported

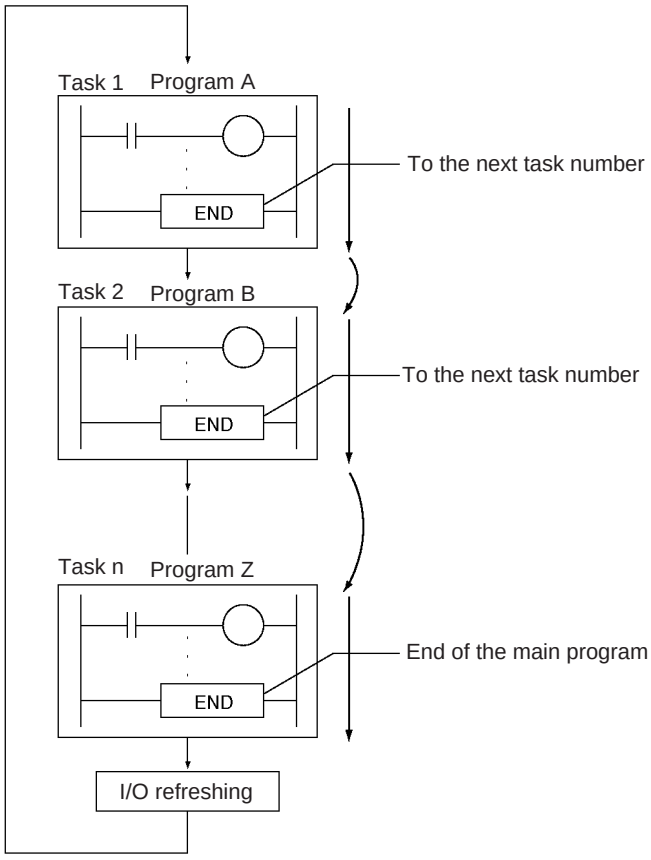
Applicable Program Areas

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

Description

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

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



Precautions

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

3-5-2 NO OPERATION: NOP(000)

Purpose

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

Ladder Symbol

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

Variations

Variations	Executed Each Cycle for ON Condition	NOP(000)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Description

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

Flags

No flags are affected by NOP(000).

Precautions

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

3-5-3 Overview of Interlock Instructions

Interlock Instructions

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

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

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

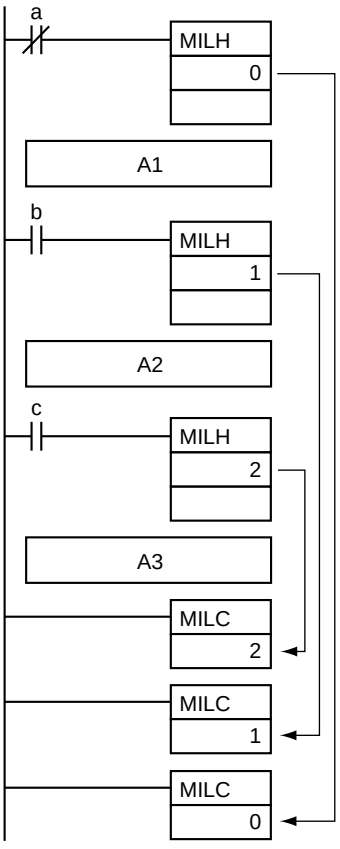
  Note MILR(518) does not hold the status of the Differentiation Flag, so differentiated instructions that were interlocked are not executed after the interlock is cleared.

  * These instructions are supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

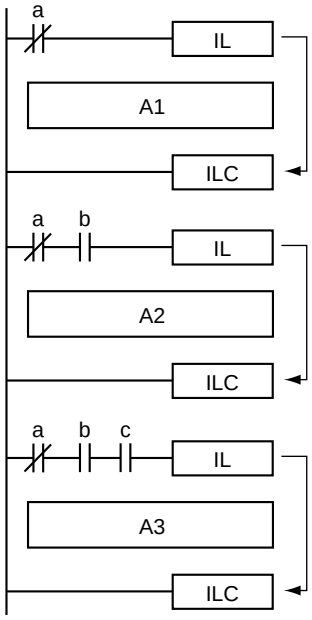
Differences between Interlocks and Multiple Interlocks

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).

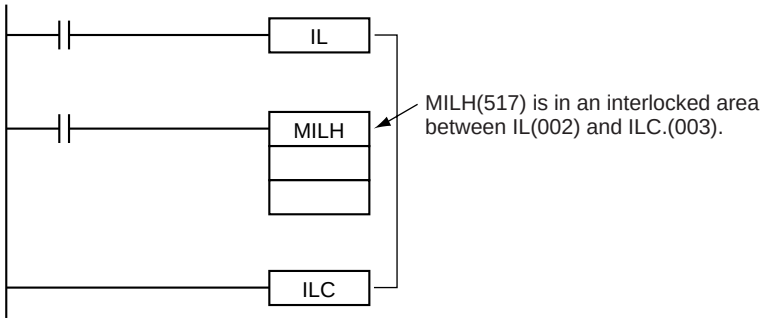
The operation of differentiated instructions in an interlock created with MILH(517) is identical to the operation in an interlock created with IL(002).

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

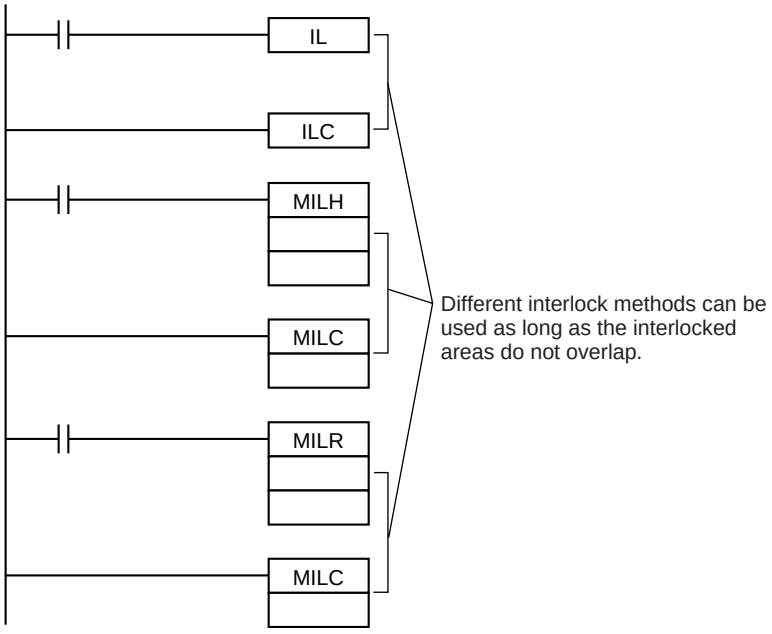
Do not combine interlocks created with different interlock instructions (IL-ILC, MILH-MILC, and MILR-MILC). The interlocks may not operate properly if different interlock methods are used together. For details on combining instructions, refer to *3-5-5 MULTI-INTERLOCK DIFFERENTIATION HOLD, MULTI-INTERLOCK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



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

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



Differences between Interlocks and Jumps

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

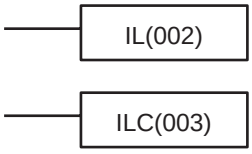
Item	Treatment in IL(002)/ILC(003), MILH(517)/ MILC(519), or MILR(518)/MILC(519))	Treatment in JMP(004)/JME(005)
Instruction execution	Instructions other than OUT, OUT NOT, OUTB(534), and timer instructions are not executed.	No instructions are executed.
Output status in instructions	Except for outputs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

3-5-4 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003)

Purpose

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

Ladder Symbols



Variations

Variations	Interlocks when OFF/Does Not interlock when ON	IL(002)
Immediate Refreshing Specification		Not supported

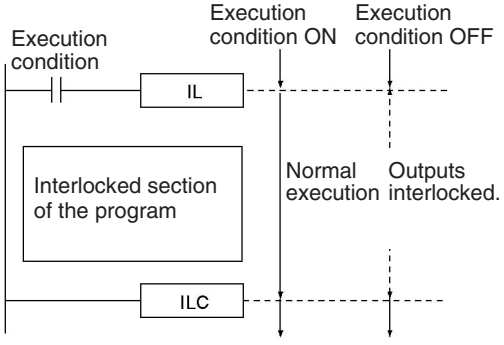
Variations	Executed Each Cycle for ON Condition	ILC(003)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Description

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



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

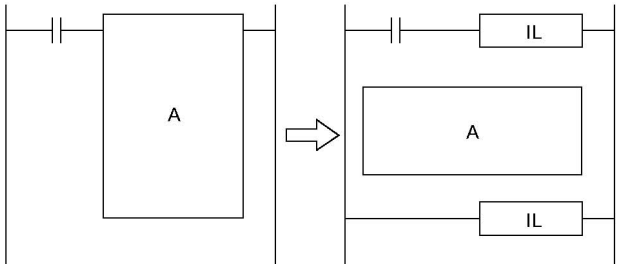
Instruction		Treatment
Bits specified in OUT, OUT NOT, or OUTB(534)		OFF
TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TIML(542), and TIMXL(553)	Completion Flag	OFF (reset)
	PV	Time set value (reset)
TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557) (See note 1.)	Cannot be referenced.	
Bits/words specified in all other instructions (See note 2.)		Retain previous status.

Note

1. These instructions are supported by the CJ1-H-R CPU Units only.
2. Bits and words in all other instructions including TTIM(087), TTIMX(555), MTIM(543), MTIMX(554), SET, RSET, CNT, CNTX(546), CNTR(012), CNTRX(548), SFT, and KEEP(011) retain their previous status.

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

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



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

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

Flags

Name	Label	Operation
Error Flag	ER	Unchanged (See note.)
Equals Flag	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

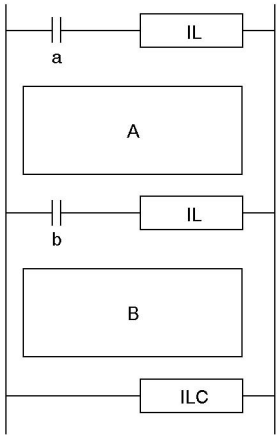
Note In CS1-H, CJ1-H, CJ1M, and CS1D (for Single-CPU System) CPU Units, the Equals and Negative Flags are left unchanged.
In CS1 and CJ1 CPU Units, the Equals and Negative Flags are turned OFF.

Precautions

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

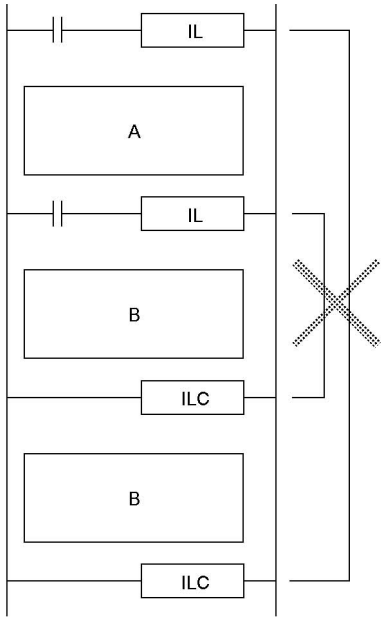
The operation of DIFU(013), DIFD(014), and differentiated instructions is not dependent solely on the status of the execution condition when they are programmed between IL(002) and ILC(003). Changes in the execution condition for DIFU(013), DIFD(014), or a differentiated instruction are not recorded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

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



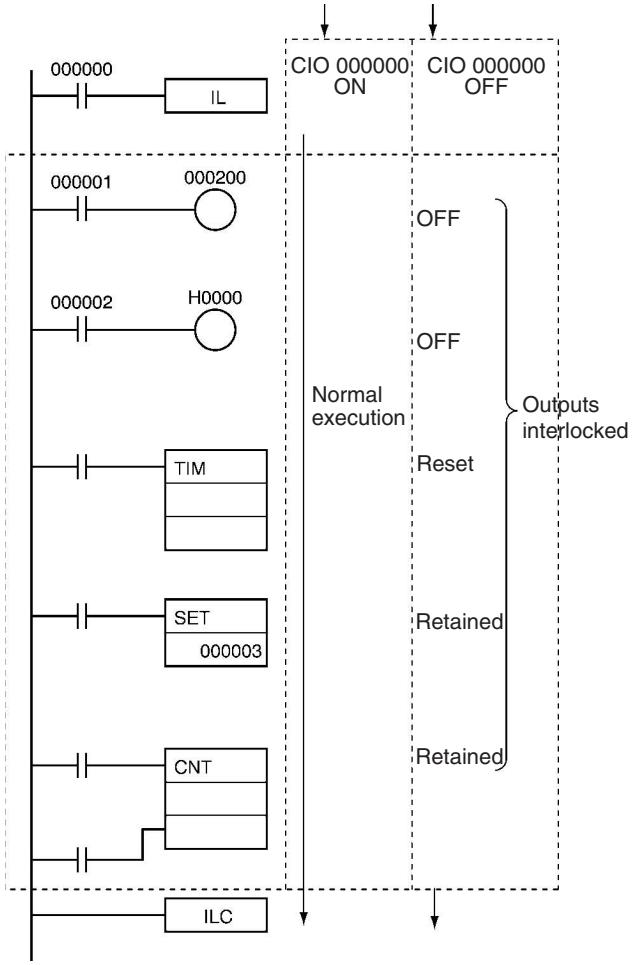
Execution condition		Program section	
a	**b**	**A**	**B**
OFF	ON	Interlocked	Interlocked
OFF	OFF	Interlocked	Interlocked
ON	OFF	Not interlocked	Interlocked
ON	ON	Not interlocked	Not interlocked

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



Examples

When CIO 000000 is OFF in the following example, all outputs between IL(002) and ILC(003) are interlocked. When CIO 000000 is ON in the following example, the instructions between IL(002) and ILC(003) are executed normally.



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

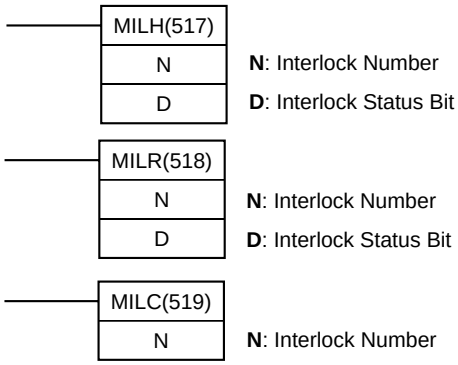
Purpose

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

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

These instructions are supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbols



Operands

N: Interlock Number

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

The interlock numbers can be used in any order.

D: Interlock Status Bit

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

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

Operand Specifications

Area	N	D
CIO Area	---	CIO 000000 to CIO 614315
Work Area	---	W00000 to W51115
Holding Bit Area	---	H00000 to H51115
Auxiliary Bit Area	---	A00000 to A95915
Timer Area	---	---
Counter Area	---	---
DM Area	---	---
EM Area without bank	---	---
EM Area with bank	---	---
Indirect DM/EM addresses in binary	---	---

Area	N	D
Indirect DM/EM addresses in BCD	---	---
Constants	0 to 15	---
Data Registers	---	---
Index Registers	---	---
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15

Variations

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

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

Applicable Program Areas

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

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

Description

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

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

Interlock Status

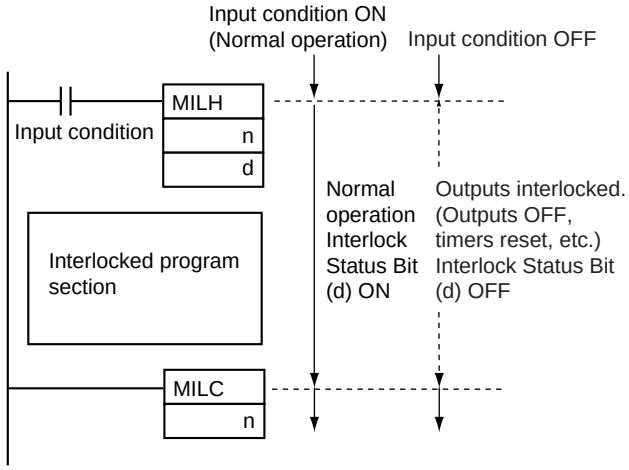
The following table shows the treatment of various outputs in an interlocked section between MILH(517)/MILR(518) instruction and the next MILC(519).

Instruction		Treatment
Bits specified in OUT, OUT NOT, or OUTB(534)		OFF
TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TIML(542), and TIMXL(553)	Completion Flag	OFF (reset)
	PV	Time set value (reset)
TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557) (See note 1.)	Cannot be referenced.	
Bits/words specified in all other instructions (See note 2.)		Retain previous status.

Note

1. These instructions are supported by the CJ1-H-R CPU Units only.
2. Bits and words in all other instructions including TTIM(087), TTIMX(555), MTIM(543), MTIMX(554), SET, RSET, CNT, CNTX(546), CNTR(012), CNTRX(548), SFT, and KEEP(011) retain their previous status.

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



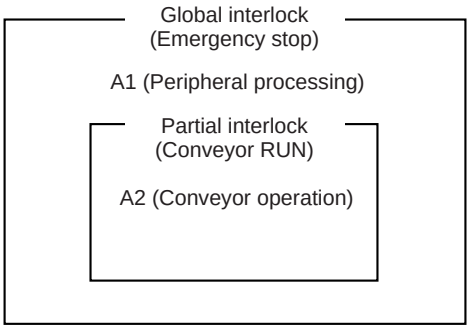
Nesting

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

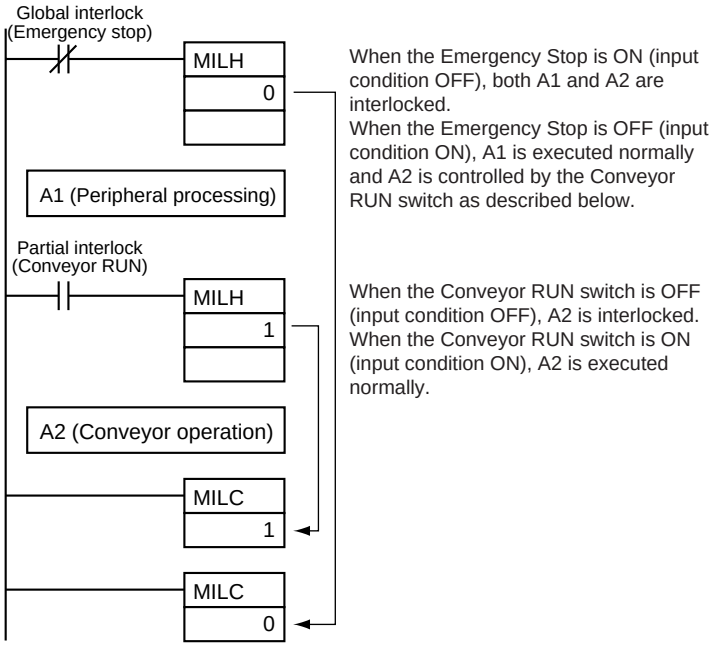
Nesting can be used for the following kinds of applications.

- Example 1

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

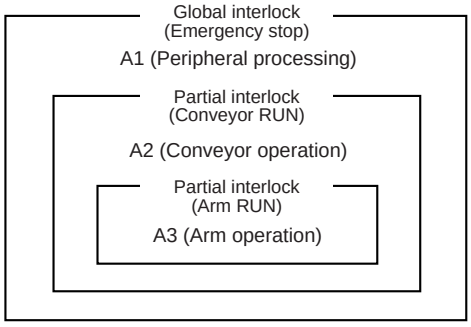


  - A1 and A2 are interlocked when the Emergency Stop Button is ON.
  - A2 is interlocked when Conveyor RUN is OFF.

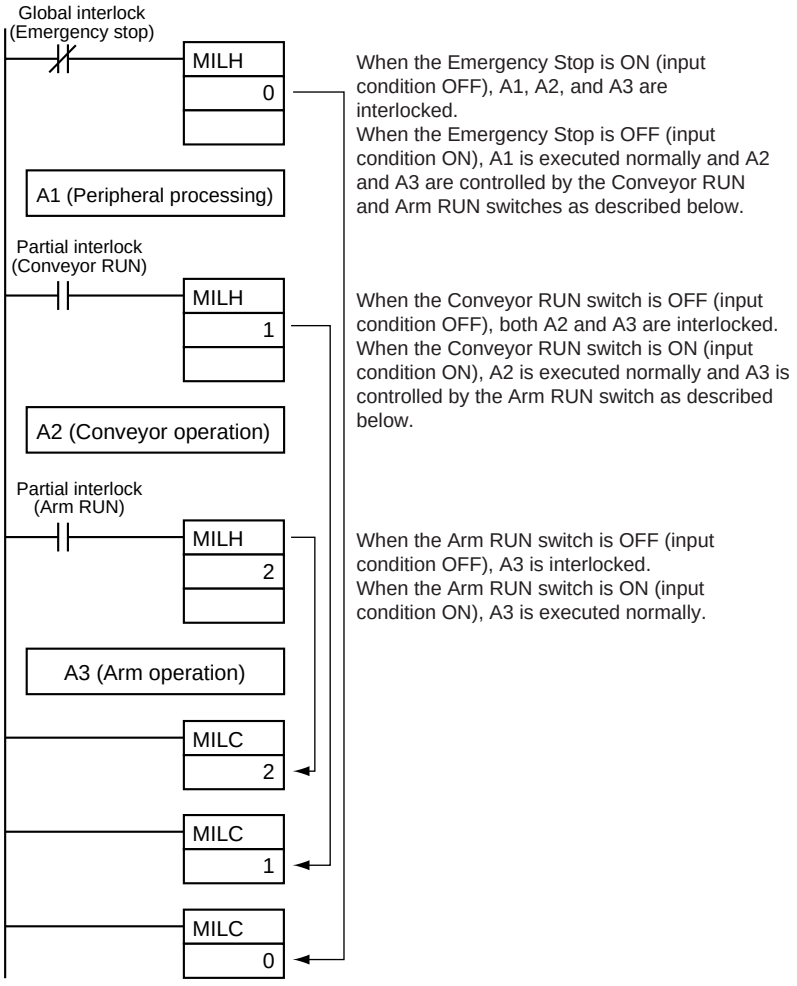


- Example 2

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



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



Differences between MILH(517) and MILR(518)

Differentiated instructions (DIFU, 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.

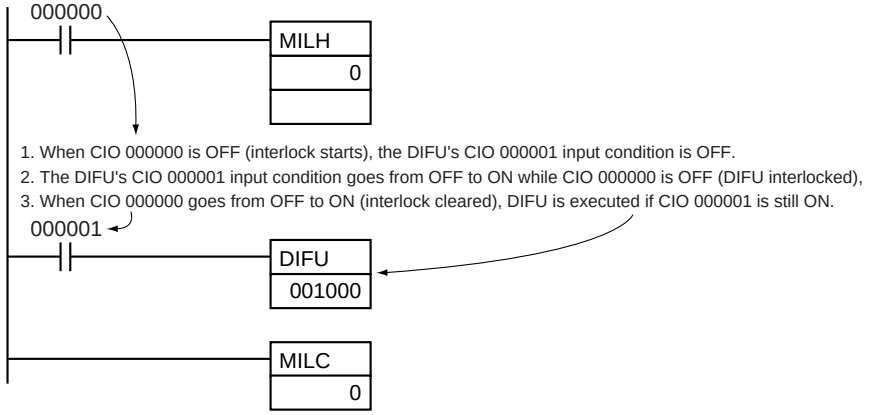
- Operation of Differentiated Instructions in an MILH(517) Interlock

  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. (The system compares the execution condition's status when the interlock started to its status when the interlock was cleared.)

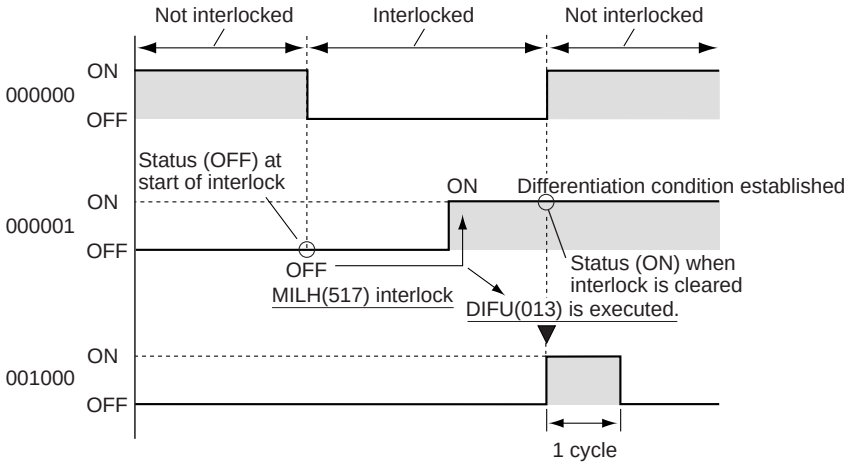
  In the same way, a differentiated instruction will be executed if its execution condition is established at the same time that the interlock is started or cleared.

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

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



Timing Chart



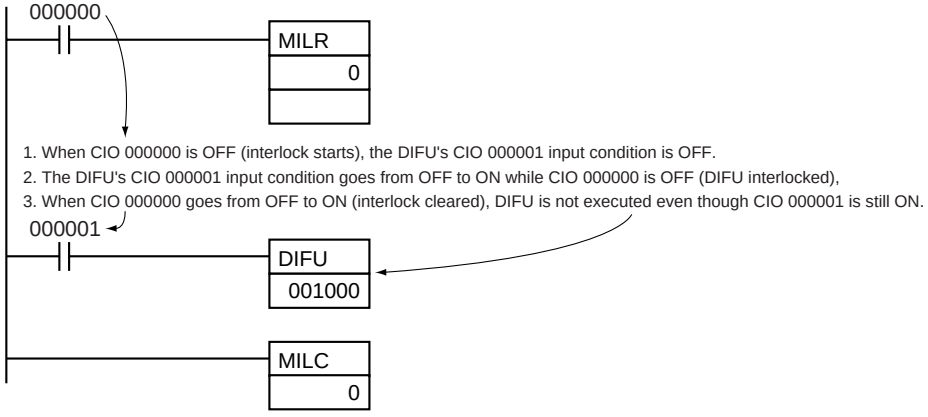
- Operation of Differentiated Instructions in an MILR(518) Interlock

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

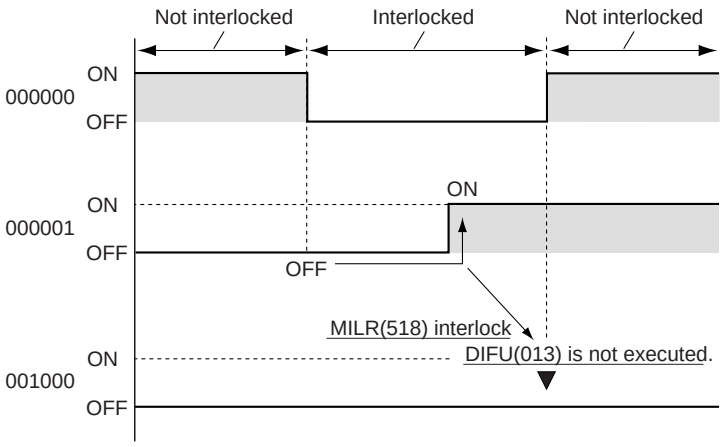
  In the same way, a differentiated instruction will not be executed if its execution condition is established at the same time that the interlock is started or cleared.

  - Example

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



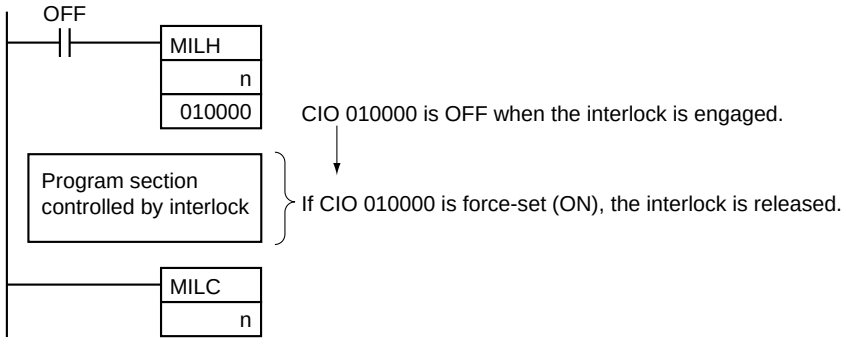
Timing Chart



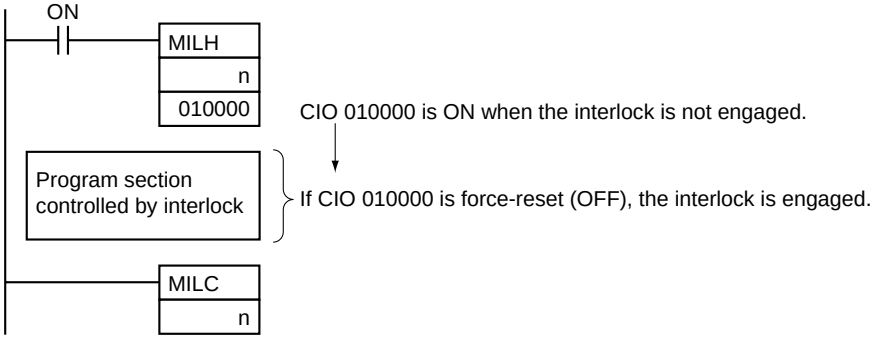
Controlling Interlock Status from a Programming Device

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.

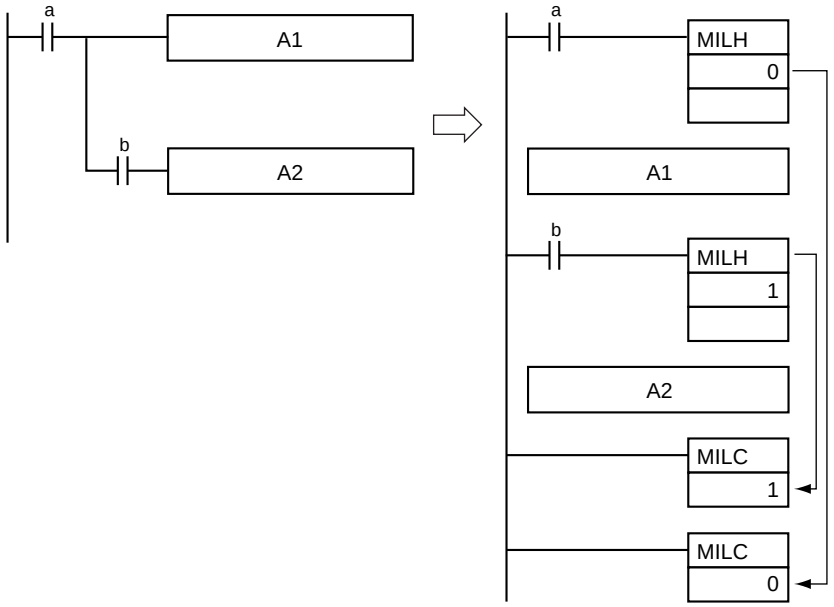


Force-reset: Engages the interlock.



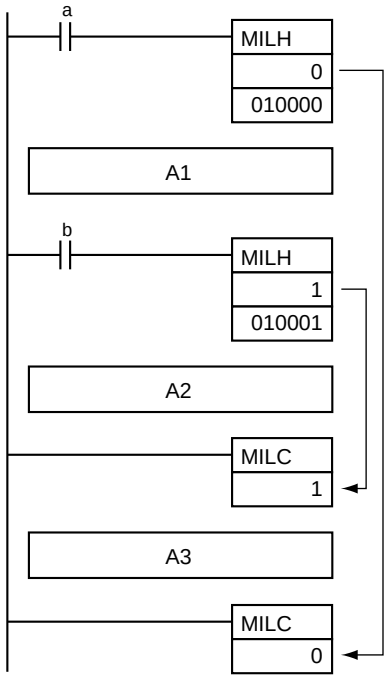
Note Program operation can be switched more efficiently by using interlocks with MILH(517) or MILR(518).

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



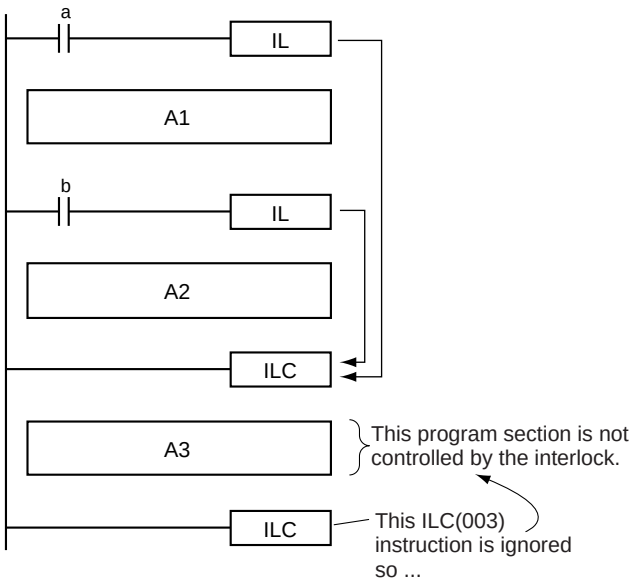
Unlike the IL(002) interlocks, MILH(517) and MILR(518) interlocks can be nested, so the operation of similar programs will be different if MILH(517) or MILR(518) is used instead of ILC(002).

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



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

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



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

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

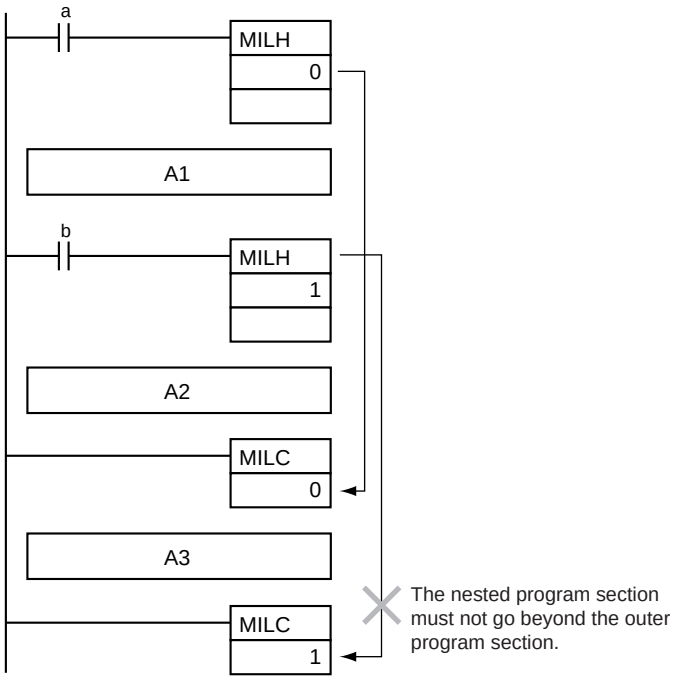
Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

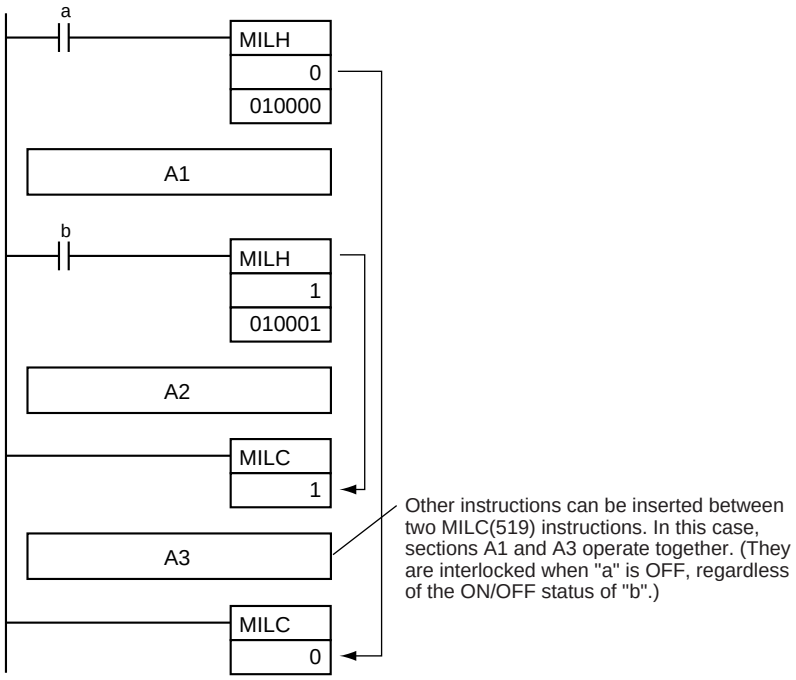
The cycle time is not shortened when a section of the program is interlocked by MILH(517) or MILR(518) because the interlocked instructions are executed internally.

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

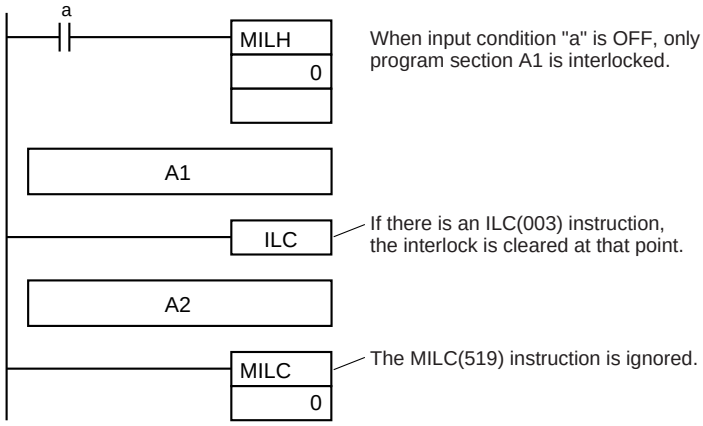


Execution condition		Program section		
a	**b**	**A1**	**A2**	**A3**
OFF	ON	Interlocked	Interlocked	Not interlocked
	OFF			
ON	OFF	Not interlocked	Interlocked	Interlocked
	ON	Not interlocked	Not interlocked	Not interlocked

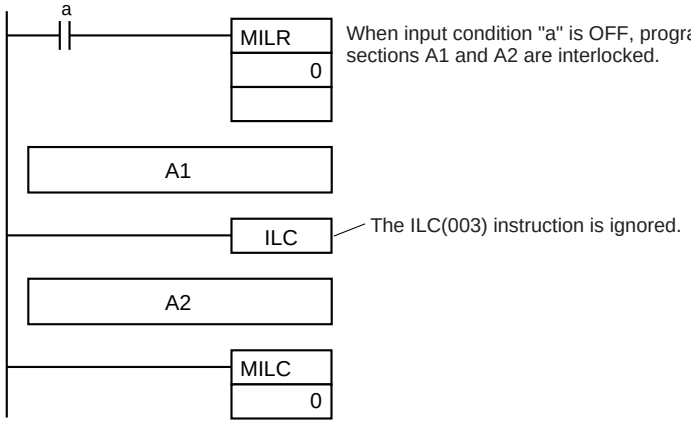
Other instructions can be input between the MILC(519) instructions, as shown in the following diagram.



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

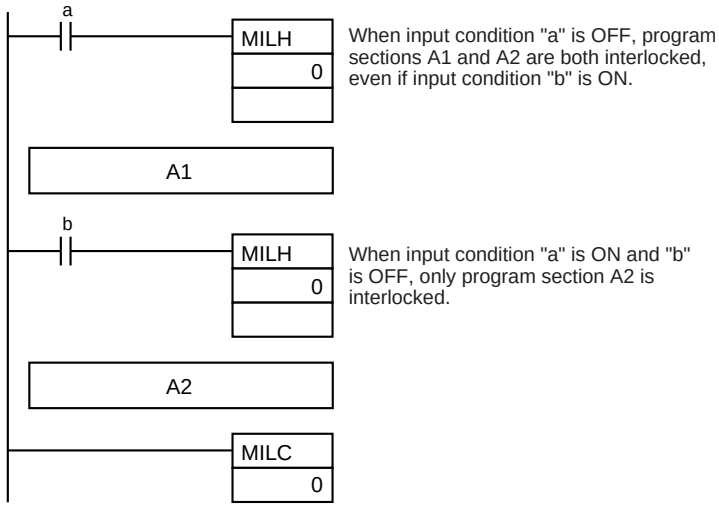


If there is an ILC(003) instruction between an MILR(518) and MILC(519) pair, the ILC(003) instruction will be ignored and the full program section between MILR(518) and MILC(519) will be interlocked.



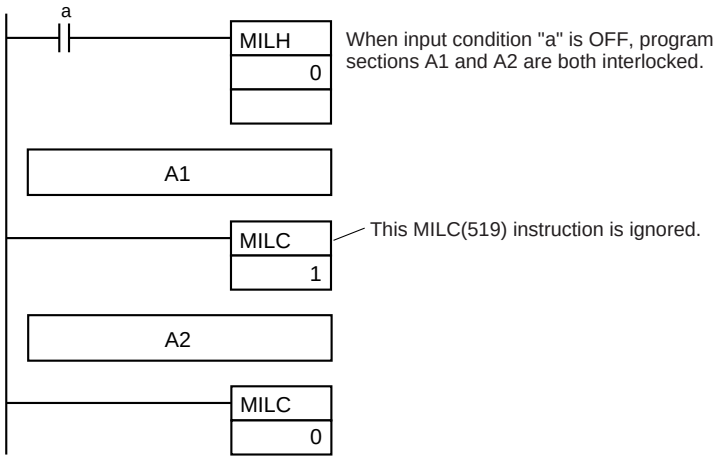
If there is another MILH(517) or MILR(518) instruction with the same interlock number between an MILH(517) and MILC(519) pair and the first MILH(517) instruction's interlock is engaged, the second MILH(517)/MILR(518) will not operate.

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



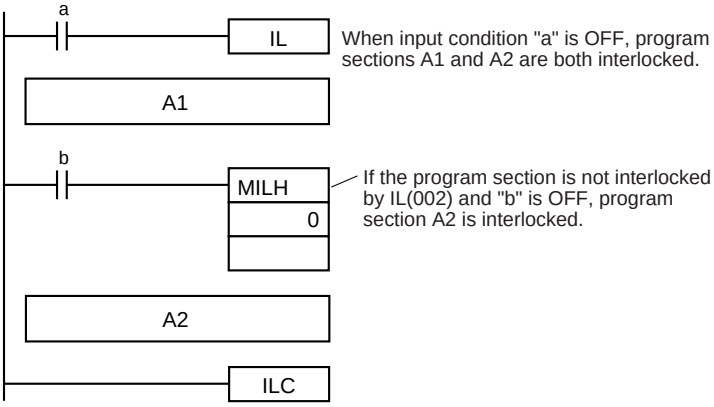
Note The MILR(518) interlocks operate in the same way if there is another MILH(517) or MILR(518) instruction with the same interlock number between an MILR(518) and MILC(519) pair.

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

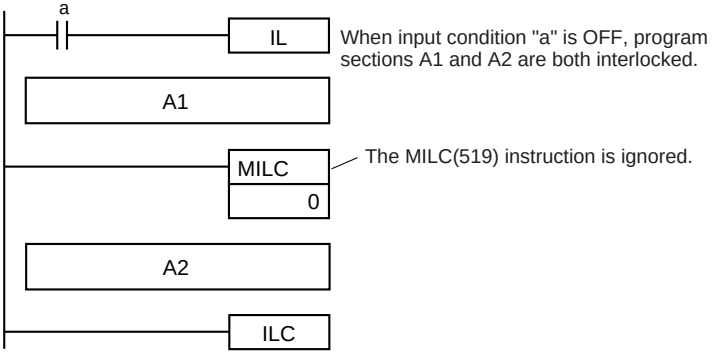


If there is an MILH(517) instruction between an IL(002) and ILC(003) pair and the IL(002) interlock is engaged, the MILH(517) instruction has no effect. In this case, the program section between IL(002) and ILC(003) will be interlocked.

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



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

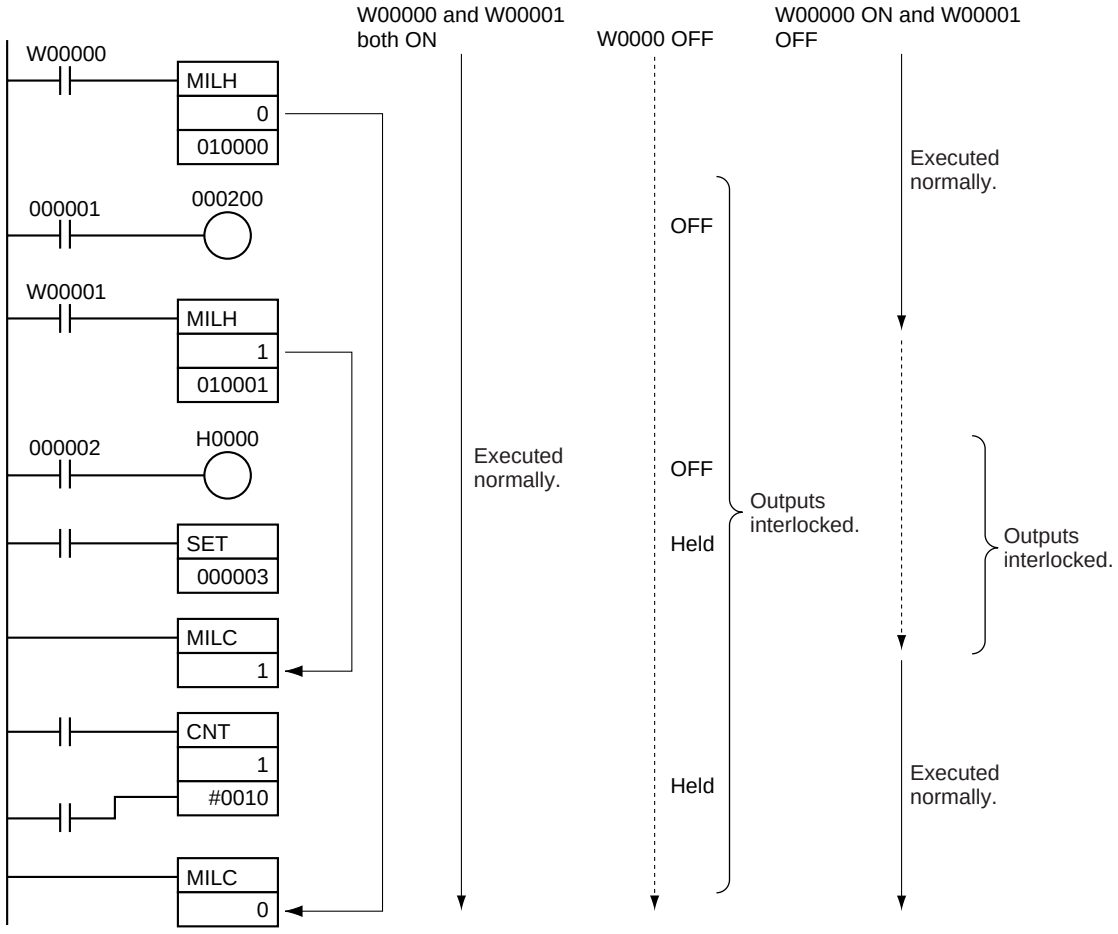


Examples

When W00000 and W00001 are both ON, the instructions between MILH(517) with interlock number 0 and MILC(519) with interlock number 0 are executed normally.

When W00000 is OFF, the instructions between MILH(517) with interlock number 0 and MILC(519) with interlock number 0 are interlocked.

When W00000 is ON and W00001 are OFF, the instructions between MILH(517) with interlock number 1 and MILC(519) with interlock number 1 are interlocked. The other instructions are executed normally.

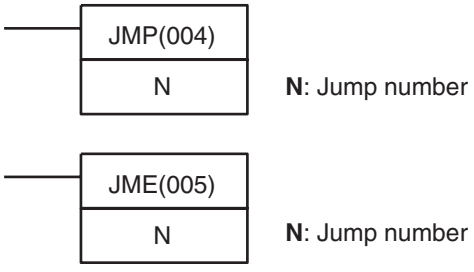


3-5-6 JUMP and JUMP END: JMP(004) and JME(005)

Purpose

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

Ladder Symbols



Variations

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

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

Applicable Program Areas

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

Operands

N: Jump Number

The jump number must be 0000 to 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	N	
	JMP(004)	JME(005)
CIO Area	CIO 0000 to CIO 6143	---
Work Area	W000 to W511	---
Holding Bit Area	H000 to H511	---
Auxiliary Bit Area	A000 to A959	---
Timer Area	T0000 to T4095	---
Counter Area	C0000 to C4095	---
DM Area	D00000 to D32767	---
EM Area without bank	E00000 to E32767	---
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	---
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	---
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	---
Constants	#0000 to #03FF (binary) or &0 to &1023 (See note.)	#0000 to #03FF (binary) or &0 to &1023 (See note.)
Data Registers	DR0 to DR15	---
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15	---

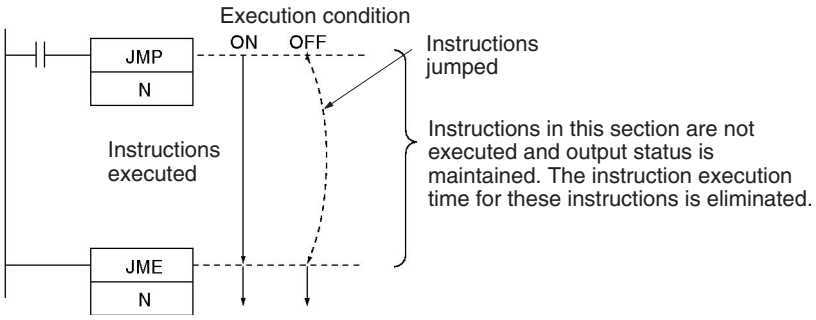
Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the range is #0000 to #00FF (binary) or &0 to &1023 (decimal).

Description

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

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

the instructions between JMP(004) and JME(005) are skipped regardless of the status of the execution condition.



Because all of instructions between JMP(004)/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.

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.

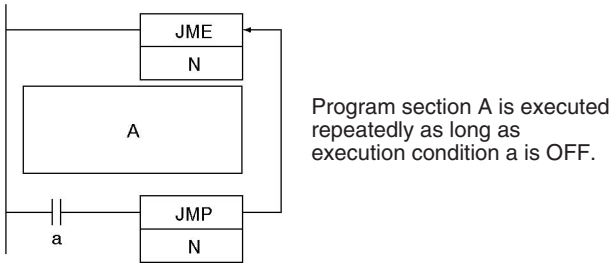
Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the range is 0 to 255 (0000 to 00FF hex).

Precautions

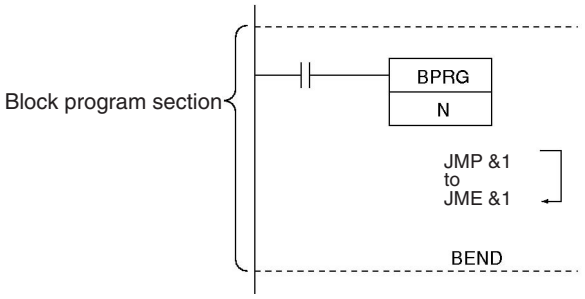
All of the outputs (bits and words) in jumped instructions retain their previous status. Operating timers (TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), TMHHX(552), TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557)) continue timing because the PVs are updated even when the timer instruction is not being executed.

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.

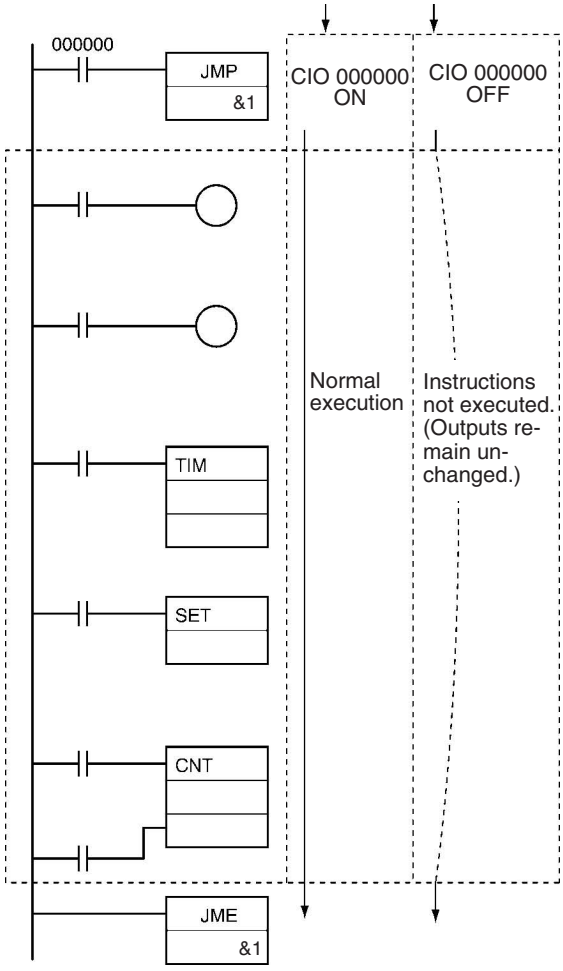
The operation of DIFU(013), DIFD(014), and differentiated instructions is not dependent solely on the status of the execution condition when they are programmed between JMP(004) and JME(005). When DIFU(013), DIFD(014), or a differentiated instruction is executed in an jumped section immediately after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

Examples

Basic Operation

When CIO 000000 is OFF in the following example, the instructions between JMP(004) and JME(005) are not executed and the outputs maintain their previous status.

When CIO 000000 is ON in the following example, the instructions between JMP(004) and JME(005) are executed normally.



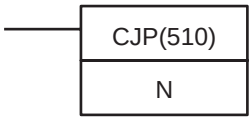
3-5-7 CONDITIONAL JUMP: CJP(510)/CJPN(511)

Purpose

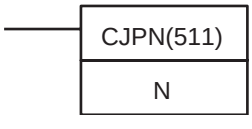
The operation of CJP(510) is the basically the opposite of JMP(004). When the execution condition for CJP(510) is ON, program execution jumps directly to the first JME(005) in the program with the same jump number. CJP(510) and JME(005) are used in pairs.

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

Ladder Symbols



N: Jump number



N: Jump number

Variations

Variations	Jumps when ON/Does Not Jump when OFF	CJP(510)
Immediate Refreshing Specification		Not supported

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

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

Applicable Program Areas

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

Operands

N: Jump Number

The jump number must be 0000 to 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	N		
	CJP(510)	CJPN(511)	JME(005)
CIO Area	CIO 0000 to CIO 6143		---
Work Area	W000 to W511		---
Holding Bit Area	H000 to H511		---
Auxiliary Bit Area	A000 to A959		---
Timer Area	T0000 to T4095		---
Counter Area	C0000 to C4095		---
DM Area	D00000 to D32767		---
EM Area without bank	E00000 to E32767		---
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		---
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		---
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		---
Constants	#0000 to #03FF (binary) or &0 to &1023 (See note.)		#0000 to #03FF (binary) or &0 to &1023 (See note.)
Data Registers	DR0 to DR15		---
Index Registers	---		---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15		---

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

Description

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

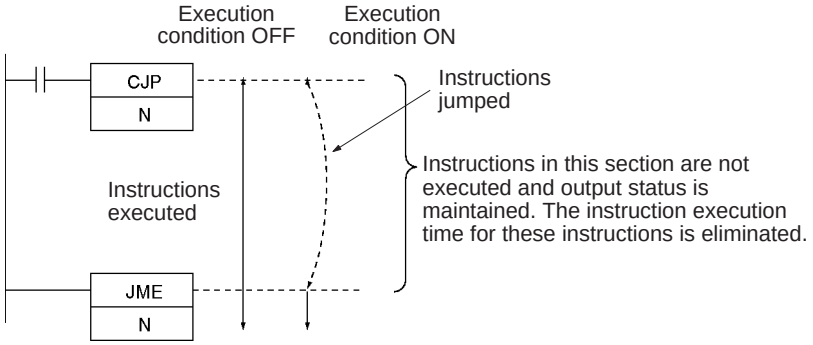
and CJPN(511) jumps to the first JME(005) when the execution condition is OFF.

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

Operation of CJP(510)

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

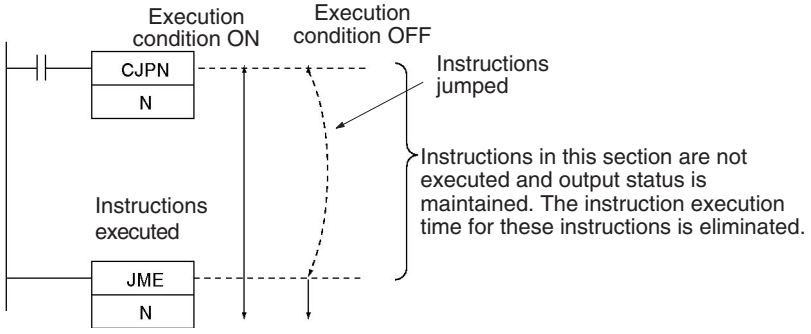
When the execution condition for CJP(510) is ON, program execution jumps directly to the first JME(005) in the program with the same jump number.



Operation of CJPN(511)

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

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



Flags

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

Name	Label	Operation
Error Flag	ER	ON if there is not a JME(005) with the same jump number as CJP(510) or CJPN(511). (See note.) ON if N is not within the specified range of 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).

Precautions

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

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

When JME(005) precedes the CJP(510) or CJPN(511) instruction in the program, the instructions in-between will be executed repeatedly as long as the execution condition remains OFF (CJP(510)) or ON (CJPN(511)). A Cycle Time Too Long error will occur if the jump is not completed by changing the execution condition executing END(001) within the maximum cycle time.

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

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

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

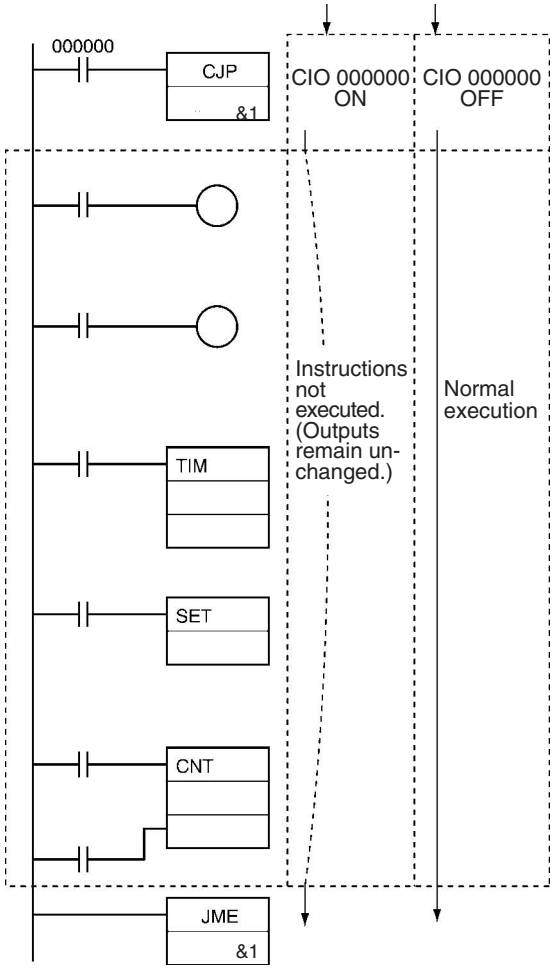
When a CJP(510) or CJPN(511) instruction is programmed in a task, there must be a JME(005) with the same jump number because jumps between tasks are not allowed. An error will occur if a corresponding JME(005) instruction is not programmed in the same task.

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

Example

When CIO 000000 is ON in the following example, the instructions between CJP(510) and JME(005) are not executed and the outputs maintain their previous status.

When CIO 000000 is OFF in the following example, the instructions between CJP(510) and JME(005) are executed normally.



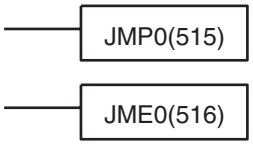
Note For CJPN(511), the ON/OFF status of CIO 000000 would be reversed.

3-5-8 MULTIPLE JUMP and JUMP END: JMP0(515) and JME0(516)

Purpose

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

Ladder Symbols



Variations

Variations	Jumps when OFF/Does Not Jump when ON	JMP0(515)
Immediate Refreshing Specification		Not supported

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

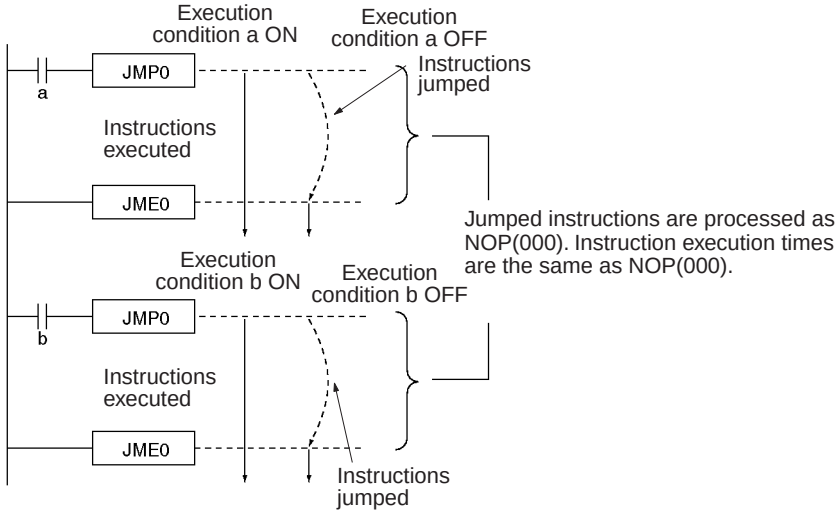
Applicable Program Areas

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

Description

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

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



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

Precautions

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

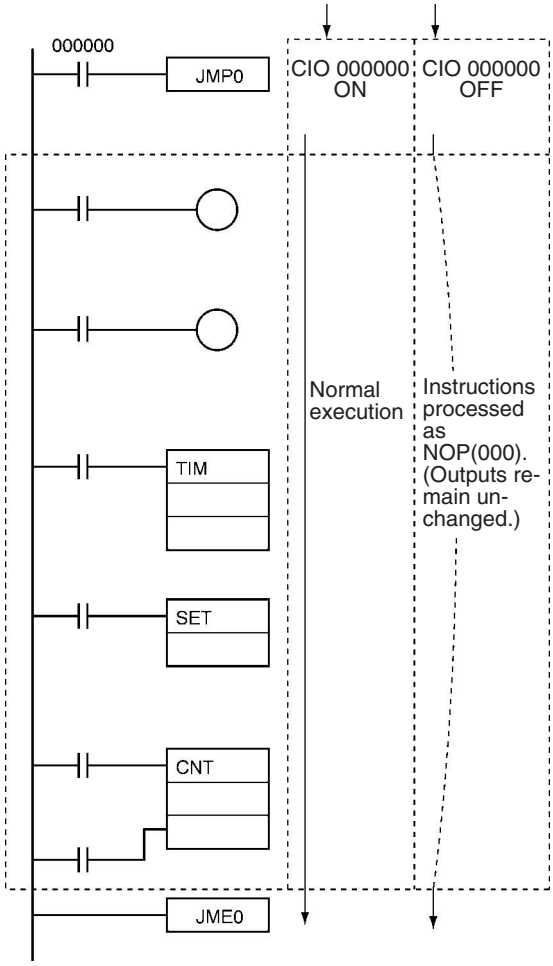
JMP0(515) and JME0(516) cannot be used in block programs.

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

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

Example

When CIO 000000 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 000000 is ON in the following example, the instructions between JMP0(515) and JME0(516) are executed normally.

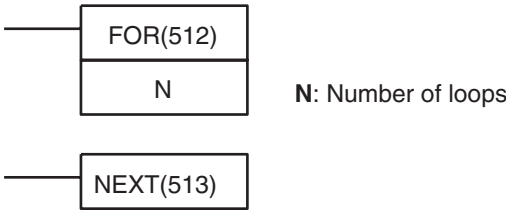


3-5-9 FOR-NEXT LOOPS: FOR(512)/NEXT(513)

Purpose

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

Ladder Symbols



N: Number of loops

Variations

Variations		
	Executed Each Cycle for ON Condition	FOR(512)
	Executed Each Cycle for ON Condition	NEXT(513)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Number of Loops

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

Operand Specifications

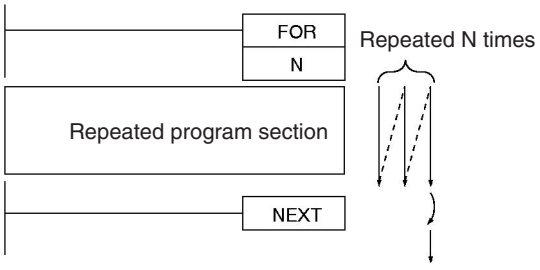
Area	N
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A000 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	#0000 to #FFFF (binary) or &0 to &65,535
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15

Description

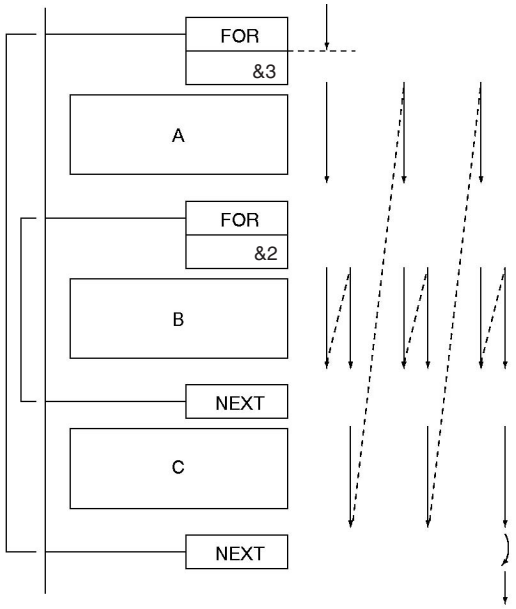
The instructions between FOR(512) and NEXT(513) are executed N times and then program execution continues with the instruction after NEXT(513). The BREAK(514) instruction can be used to cancel the loop.

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

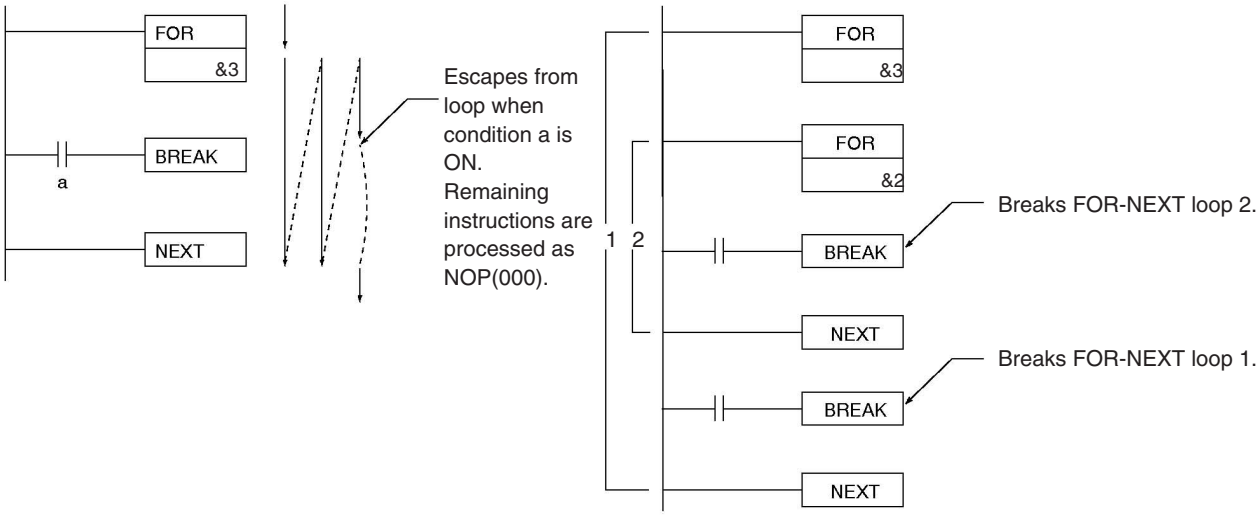
Loops can be used to process tables of data with a minimum amount of programming.



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



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



Alternative Looping Methods

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

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

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

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

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

Flags

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

Precautions

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

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

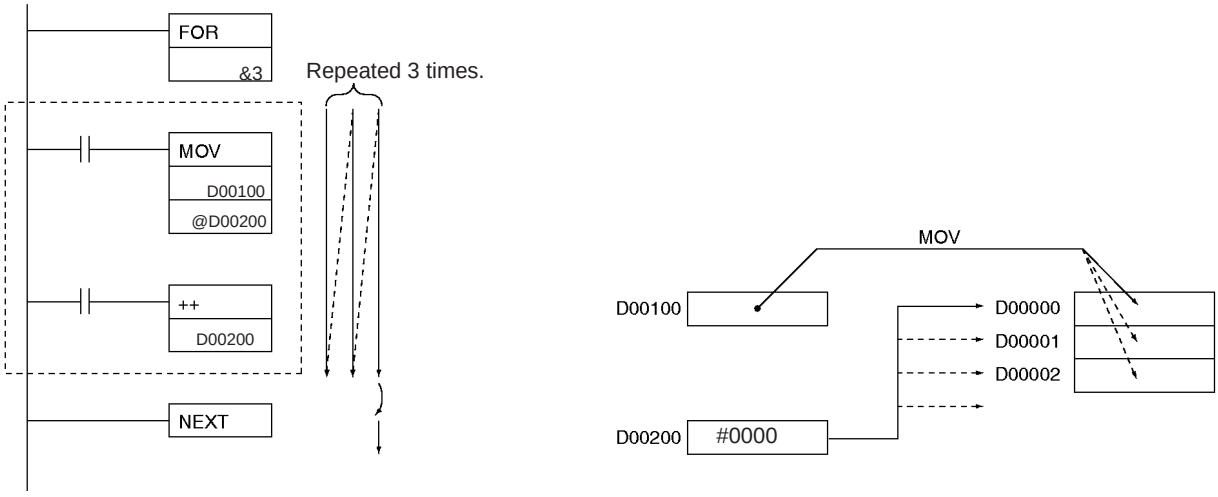
The following instructions cannot be used within FOR-NEXT loops:

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

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

Example

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



3-5-10 BREAK LOOP: BREAK(514)

Purpose

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

Ladder Symbol

Variations

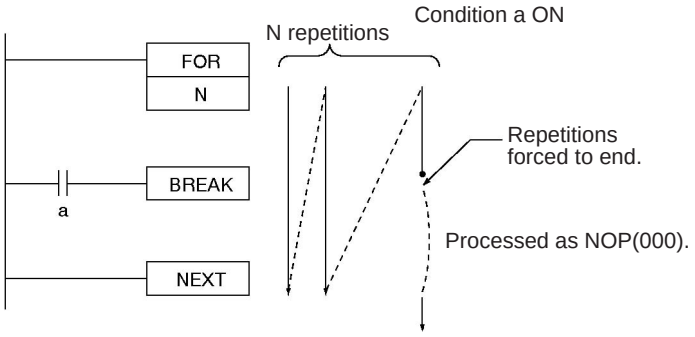
Variations	**Executed Each Cycle for ON Condition**	BREAK(514)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Description

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



Flags

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

Precautions

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

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

3-6 Timer and Counter Instructions

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

Instruction	Mnemonic	Function code	Page
HUNDRED-MS TIMER	TIM/TIMX	---/551	245
TEN-MS TIMER	TIMH/TIMHX	015/551	249
ONE-MS TIMER	TMHH/TIMHHX	540/552	253
TENTH-MS TIMER (See note.)	TIMU/TIMUX	541/556	256
HUNDREDTH-MS TIMER (See note.)	TMUH/TMUHX	544/557	259
ACCUMULATIVE TIMER	TTIM/TTIMX	087/555	262
LONG TIMER	TIML/TIMLX	542/553	266
MULTI-OUTPUT TIMER	MTIM/MTIMX	543/554	269
COUNTER	CNT/CNTX	---/546	275
REVERSIBLE COUNTER	CNTR/CNTRX	012/548	278
RESET TIMER/COUNTER	CNR/CNRX	545/547	282

Note TIMU(541), TIIMUX(556), TMUH(544), and TMUHX(557) are supported by CJ1-H-R CPU Units only.

Refresh Methods for Timer/Counter PV

■ Overview

In the CS1-H, CS1D, CJ1-H, and CJ1M CPU Units, the PV refresh method can be set to either BCD or binary for all of the timer/counter-related instructions. (See notes 1 and 2.)

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

Refer to *6-4 Changing the Timer/Counter PV Refresh Mode* in the *CS/CJ Series Programming Manual* (W394) for details on refresh methods.

Note

1. With CS1-H and CJ1-H CPU Units manufactured prior to 31 May 2002, the binary instructions will be displayed on the Programming Console with the mnemonic of the equivalent instruction for BCD operation. (For example, TIMX0 &16 will be displayed as TIM0 &16.) The instruction, however, will operate using binary mode.
2. The refresh method can be selected only with CX-Programmer version 3.0 or later. It cannot be selected with version 2.1 or early, or from a Programming Console.
3. User programs that use the binary update mode cannot be read with CX-Programmer version 2.1 or lower. They can be read only by changing to BCD mode.

■ 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)

Note TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557) are supported by CJ1-H-R CPU Units only.

Basic Timer Specifications

The following table shows the basic specifications of the timers.

Item		TIM/ TIMX(550)	TIMH(015)/ TIMHX(551)	TMHH(540)/ TMHHX(552)	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		Decrement-ing	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		At execution	At execution	At execution	At execution	At execution	At execution	At execution	At execution
Timer PV refreshing (See note 5.)		See note 1.	See note 2.	Every 1 ms At execution	At execution	At execution	At execution	At execution	At execution
Value after reset	Com-pletion Flags	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
	PVs	SV	SV	SV	--- (See note 4.)		0	SV	0

Note

1. TIM PVs are refreshed at execution, at the end of program execution each cycle, or every 80 ms by interrupt if the cycle time exceeds 80 ms.
2. TIMH(015)/TIMHX(551) PVs are refreshed at execution, at the end of program execution each cycle, and every 10 ms by interrupt.
3. TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557) are supported by CJ1-H-R CPU Units only.
4. It is not possible to read the timer PVs of TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557).
5. Timers are refreshed at different times depending on the timer number. Refer to the descriptions of individual timer instructions for details.

Timer Operation

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

Item		TIM/ TIMX(550)	TIMH(015)/ TIMHX(551)	TMHH(540)/ TMHHX(552)	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 applica-ble	Not applica-ble
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	Comple-tion Flag	ON						---	---
	PVs	Set to 0.			--- (See note 2.)		Set to 0.	---	---
Forced reset	Comple-tion Flags	OFF						---	---
	PVs	Reset to SV.			--- (See note 2.)		Set to 0.	---	---

Note

1. TIMU(541), TIMUX(556), TMUH(544), and TMUHX(557) are supported by CJ1-H-R CPU Units only.

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

3-6-1 HUNDRED-MS TIMER: TIM/TIMX(550)

Purpose

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

Note The timer accuracy for CS1D CPU Units is 10 ms + the cycle time. The timer accuracy for unit version 4.1 of the CJ1-H-R CPU Units is –0.1 to 0 s. The timer accuracy for other unit versions of the CJ1-H-R CPU Units is 0 to 0.01 s.

Ladder Symbol

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

Variations

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

Applicable Program Areas

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

Operands

N: Timer Number

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

S: Set Value

The set value must be between #0000 and 9999 (BCD).
(If the set value is set to #0000, the Completion Flag will be turned ON when TIM/TIMX(550) is executed.)

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	0000 to 4095 (decimal)	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767

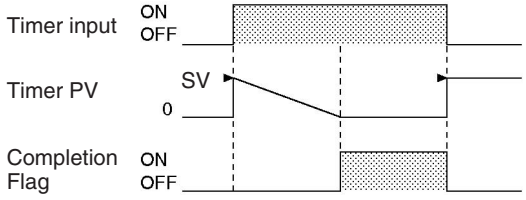
Area	N	S
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_032767 (n = 0 to C)
Constants	---	BCD: #0000 to 9999 (BCD) "&" cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

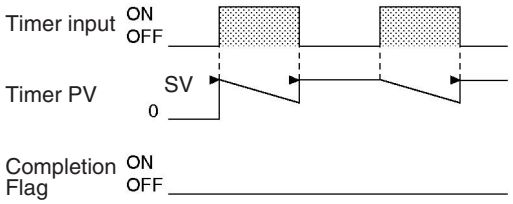
When the timer input is OFF, the timer specified by N is reset, i.e., the timer's PV is reset to the SV and its Completion Flag is turned OFF.

When the timer input goes from OFF to ON, TIM/TIMX(550) starts decrementing the PV. The PV will continue timing down as long as the timer input remains ON and the timer's Completion Flag will be turned ON when the PV reaches 0000.

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



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



Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the 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	=	OFF or unchanged (See note.)
Negative Flag	N	OFF or unchanged (See note.)

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

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 80 ms. Use timer numbers 0000 to 2047 when the cycle time is longer than 80 ms.

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

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

Condition	PV	Completion Flag
Operating mode changed from RUN or MONITOR mode to PROGRAM mode or vice versa.[1]	0000	OFF
Power supply interrupted and reset[2]	0000	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions[3]	BCD: 9999 Binary: FFFF	OFF
Operation in interlocked program section (IL(002)–ILC(003))	Reset to SV.	OFF
Operation in jumped program section (JMP(004)–JME(005))	PV continues decrementing.	Retains previous status.

Note

1. If the IOM Hold Bit (A50012) 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 (A50012) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted.
3. The PV will be set to the SV when TIM/TIMX(550) is executed.

When TIM/TIMX(550) is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will be reset to the SV and the Completion Flag will be turned OFF.

When an operating TIM/TIMX(550) timer created with a timer number between 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 operation of the = Flag and N Flag depends on the model of the CPU Unit. Refer to *Flags*, above, for details.

The timer's Completion Flag is refreshed only when TIM/TIMX(550) is executed, so a delay of up to one cycle may be required for the Completion Flag to be turned ON after the timer times out.

If online editing is used to overwrite a timer instruction, always reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

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

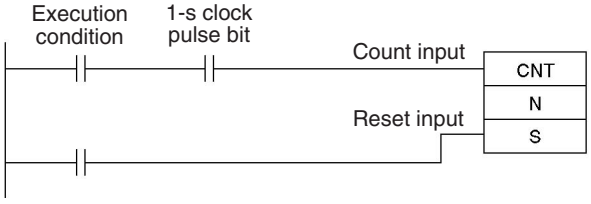
Timers Created with Timer Numbers 0000 to 2047

Execution of TIM/ TIMX(550)	The PV is updated every time that TIM/TIMX(550) is executed. The Completion Flag is turned ON if the PV is 0000. The Completion Flag is turned OFF if the PV is not 0000.
After executing all tasks	The PV is also updated every cycle at the end of program execution.
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

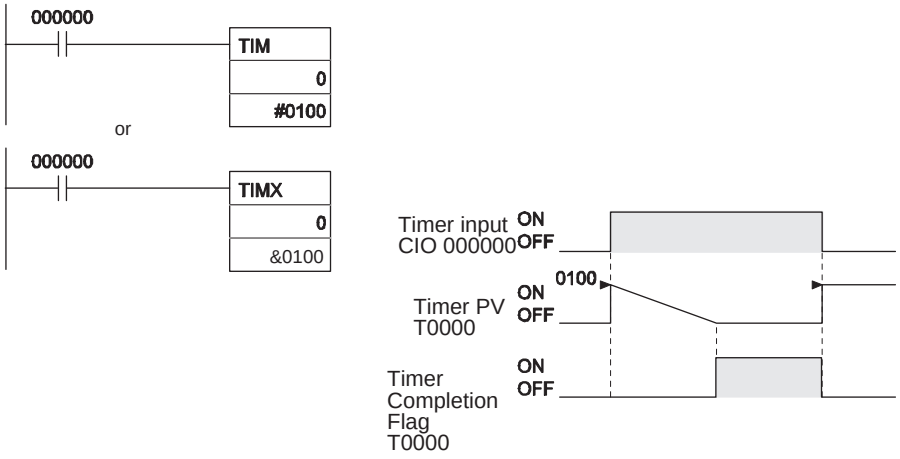
Execution of TIM	The PV is updated every time that TIM is executed. The Completion Flag is turned ON if the PV is 0000. The Completion Flag is turned OFF if the PV is not 0000.

Timers are reset (PV = SV, Completion Flag OFF) by power interruptions unless the IOM Hold Bit (A50012) is ON and the bit is protected in the PLC Setup. It is also possible use a clock pulse bit and a counter instruction to program a timer that will retain its PV in the event of a power interruption, as shown in the following diagram.



Example

When timer input CIO 000000 goes from OFF to ON in the following example, the timer PV will begin counting down from the SV. Timer Completion Flag T0000 will be turned ON when the PV reaches 0000.
When CIO 000000 goes OFF, the timer PV will be reset to the SV and the Completion Flag will be turned OFF.



3-6-2 TEN-MS TIMER: TIMH(015)/TIMHX(551)

Purpose

TIMH(015)/TIMHX(551) operates a decrementing timer with units of 10-ms. The setting range for the set value (SV) is 0 to 99.99 s for TIMH(015) and 0 to 655.35 s for TIMHX(551). The timer accuracy is 0 to 0.01 s.

Note The timer accuracy for CS1D CPU Units is 10 ms + the cycle time

Ladder Symbol

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

Variations

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

Applicable Program Areas

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

Operands

N: Timer Number

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

S: Set Value

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

Operand Specifications

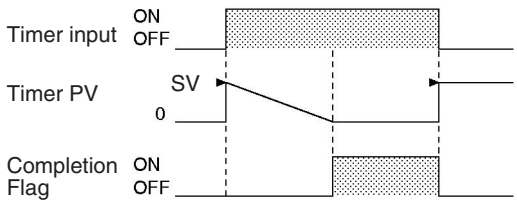
Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	0000 to 4095 (decimal)	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---	BCD: #0000 to 9999 (BCD) "&" cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

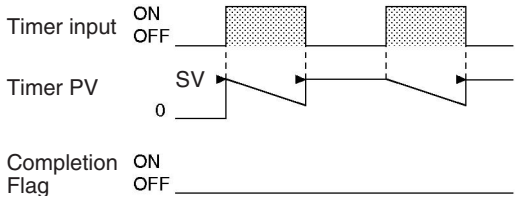
When the timer input is OFF, the timer specified by N is reset, i.e., the timer's PV is reset to the SV and its Completion Flag is turned OFF.

When the timer input goes from OFF to ON, TIMH(015)/TIMHX(551) starts decrementing the PV. The PV will continue timing down as long as the timer input remains ON and the timer's Completion Flag will be turned ON when the PV reaches 0000.

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



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



Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the 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	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

Note In 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 are turned OFF.

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 80 ms. Use timer numbers 0000 to 2047 when the cycle time is longer than 80 ms.

TIMH(015)/TIMHX(551) timers created with timer numbers 0000 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 operation of the = Flag and N Flag depends on the model of the CPU Unit. Refer to *Flags*, above, for details.

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]	0000	OFF
Power supply interrupted and reset[2]	0000	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions[3]	BCD: 9999 Binary: FFFF	OFF
Operation in interlocked program section (IL(002)–ILC(003))	Reset to SV.	OFF
Operation in jumped program section (JMP(004)–JME(005))	PV continues decrementing.	Retains previous status.

Note

1. If the IOM Hold Bit (A50012) 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 (A50012) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted.
3. The PV will be set to the SV when TIMH(015)/TIMHX(551) is executed.

When an operating TIMH(015)/TIMHX(551) timer created with a timer number between 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 operation of the = Flag and N Flag depends or the model of CPU Unit. Refer to *Flags* for details.

The timer's Completion Flag is refreshed only when TIMH(015)/TIMHX(551) is executed, so a delay of up to one cycle may be required for the Completion Flag to be turned ON after the timer times out.

If online editing is used to overwrite a timer instruction, always reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

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

Timers Created with Timer Numbers 0000 to 0255

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

Timers Created with Timer Numbers 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 0000. The Completion Flag is turned OFF if the PV is not 0000.
After executing all tasks	The PV is also updated every cycle at the end of program execution.
80-ms interval refreshing	If the cycle time exceeds 80 ms, the timer's PV is updated every 80 ms.

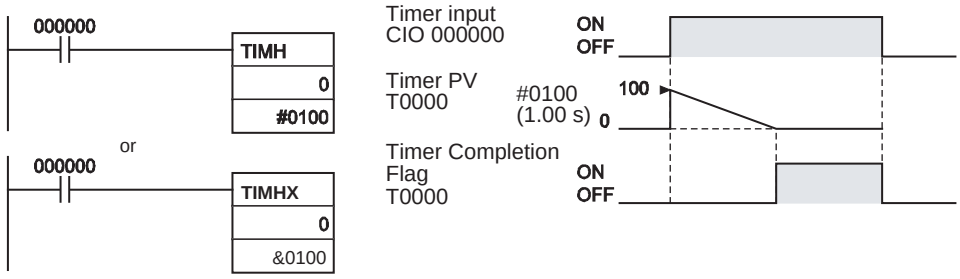
Timers Created with Timer Numbers 2048 to 4095

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 0000. The Completion Flag is turned OFF if the PV is not 0000.

Example

When timer input CIO 000000 goes from OFF to ON in the following example, the timer PV will begin counting down from the SV (#0064 = 100 = 1.00 s). The Timer Completion Flag, T0000, will be turned ON when the PV reaches 0000.

When CIO 000000 goes OFF, the timer PV will be reset to the SV and the Completion Flag will be turned OFF.



3-6-3 ONE-MS TIMER: TMHH(540)/TMHHX(552)

Purpose

TMHH(540)/TMHHX(552) operates a decrementing timer with units of 1-ms. The setting range for the set value (SV) is 0 to 9.999 s for TMHH(540) and 0 to 65.535 for TMHHX(552). The timer accuracy is –0.001 to 0 s.

Note The timer accuracy for CS1D CPU Units is 10 ms + the cycle time. The timer accuracy for unit version 4.1 of the CJ1-H-R CPU Units is –0.01 to 0 s. The timer accuracy for other unit versions of the CJ1-H-R CPU Units is –0.001 to 0 s.

Ladder Symbol

PV refresh method	Symbol	Operands
BCD	TMHH(540) N S **N**: Timer number **S**: Set value	N: 0 to 15 decimal, or 0 to 4,095 decimal (See note.) S: #0000 to #9999 (BCD)
Binary	TMHHX(552) N S **N**: Timer number **S**: Set value	N: 0 to 15 decimal, or 0 to 4,095 decimal (See note.) S: &0 to &65535 decimal #0000 to #FFFF hex

Note In CJ1-H-R CPU Units other than those with unit version 4.1, N can be set to between 0 and 4,095 decimal. In CJ1-H-R CPU Units with unit version 4.1, N can be set only to between 16 and 4095 decimal. For details, refer to *Refreshing of TMHH(540) and TMHHX(552) PVs and Completion Flags* on page 256.

Variations

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

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK in CJ1-H-R CPU Units only	OK	OK	Not allowed

Operands

N: Timer Number

The timer number must be between 0000 and 0015 (decimal).

S: Set Value

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

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	0000 to 0015 decimal, or 0000 to 4095 (See note.)	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Note In CJ1-H-R CPU Units other than those with unit version 4.1, N can be set to between 0 and 4,095 decimal. In CJ1-H-R CPU Units with unit version 4.1, N can be set only to between 16 and 4095 decimal. For details, refer to *Refreshing of TMHH(540) and TMHHX(552) PVs and Completion Flags* on page 256.

Description

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

When the timer input goes from OFF to ON, TMHH(540)/TMHHX(552) starts decrementing the PV. The PV will continue timing down as long as the timer

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

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

Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the 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	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

Note In 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 are turned OFF.

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. The present value of a high-speed timer with a timer number from 16 to 4095 will be held 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]	0000	OFF
Power supply interrupted and reset[2]	0000	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions[3]	BCD: 9999 Binary: FFFF	OFF
Operation in interlocked program section (IL(002)–ILC(003))	Reset to SV.	OFF
Operation in jumped program section (JMP(004)–JME(005))	PV continues decrementing.	Retains previous status.

Note

1. If the IOM Hold Bit (A50012) 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 (A50012) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted.
3. The PV will be set to the SV when TMHH(540)/TMHHX(552) is executed.

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

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

If online editing is used to overwrite a timer instruction, always reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

Refreshing of TMHH(540) and TMHHX(552) PVs and Completion Flags

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

Timer numbers 0 to 15 (Cannot be used with unit version 4.1 of the CJ1-H-R CPU Units, but can be used with other unit versions of the CJ1-H-R CPU Units.):

Refresh timing	Data refreshed
Execution of TMHH(540)/ TMHHX(552)	The Completion Flag is turned ON if the PV is 0000. The Completion Flag is turned OFF if the PV is not 0000.
1-ms interval refreshing	The timer's PV is refreshed every 1 ms.

Timer numbers 16 to 4,095 (CJ1-H-R CPU Units only):

Refresh timing	Data refreshed
Execution of TMHH(540)/ TMHHX(552)	The Completion Flag is turned ON if the PV is 0000. The Completion Flag is turned OFF if the PV is not 0000.

3-6-4 TENTH-MS TIMER: TIMU(541)/TIMUX(556)

Purpose

TIMU(541)/TIMUX(556) operates a decrementing timer with units of 0.1-ms. 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.

Note These instructions can be used in the CJ1-H-R CPU Units only.

Ladder Symbol

PV refresh method	Symbol	Operands
BCD	TIMU(541) N — **N**: Timer number S — **S**: Set value	N: 0000 to 4095 (decimal) S: #0000 to #9999 (BCD)
Binary	TIMUX(556) N — **N**: Timer number S — **S**: Set value	N: 0000 to 4095 (decimal) S: &0 to &65535 (decimal) #0000 to #FFFF (hex)

Variations

Variations		TIMU(541)/ TIMUX(556)
	Executed Each Cycle for ON Condition	
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	Not allowed	OK	OK	Not allowed

Operands

N: Timer Number

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

S: Set Value

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

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	0000 to 4095 (decimal)	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)

Area	N	S
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

When the timer input is OFF, the timer specified by N is reset, i.e., the timer’s 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.

Flags

Name	Label	Operation
Error Flag	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	=	Unchanged
Negative Flag	N	Unchanged

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

Condition	Completion Flag
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 (A50012) 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 (A50012) 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.

Note 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.

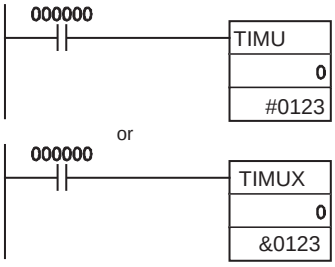
When a TIMU(541)/TIMUX(556) timer is forced set, its Completion Flag will be turned ON. When a TIMU(541)/TIMUX(556) timer is forced reset, its Completion Flag will be turned OFF.

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.

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.

Operation Example



When timer input CIO 000000 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 000000 goes OFF, the Timer Completion Flag, T0000, will be turned OFF.

3-6-5 HUNDREDTH-MS TIMER: TMUH(544)/TMUHX(557)

Purpose

TMUH(544)/TMUHX(557) operates a decrementing timer with units of 0.01-ms. 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.

Note These instructions can be used in the CJ1-H-R CPU Units only.

Ladder Symbol

PV refresh method	Symbol	Operands
BCD	TMUH(541) N — **N**: Timer number S — **S**: Set value	N: 0000 to 4095 (decimal) S: #0000 to #9999 (BCD)
Binary	TMUHX(557) N — **N**: Timer number S — **S**: Set value	N: 0000 to 4095 (decimal) S: &0 to &65535 (decimal) #0000 to #FFFF (hex)

Variations

Variations		
	Executed Each Cycle for ON Condition	TMUH(544)/ TMUHX(557)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	Not allowed	OK	OK	Not allowed

Operands

N: Timer Number

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

S: Set Value

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

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	0000 to 4095 (decimal)	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)

Area	N	S
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

When the timer input is OFF, the timer specified by N is reset, i.e., the timer’s 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.

Flags

Name	Label	Operation
Error Flag	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	=	Unchanged
Negative Flag	N	Unchanged

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

Condition	Completion Flag
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 (A50012) 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 (A50012) 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.

Note 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.

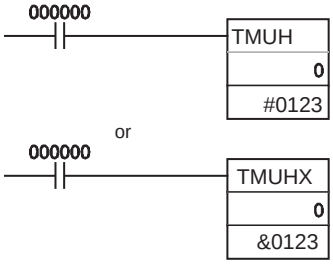
When a TIMU(541)/TIMUX(556) timer is forced set, its Completion Flag will be turned ON. When a TIMU(541)/TIMUX(556) timer is forced reset, its Completion Flag will be turned OFF.

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.

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

Execution of TMUH(544) /TMUHX(557)	The Completion Flag is turned ON if the SV is reached. The Completion Flag is turned OFF if the SV has not been reached.

Operation Example



When timer input CIO 000000 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 000000 goes OFF, the Timer Completion Flag, T0000, will be turned OFF.

3-6-6 ACCUMULATIVE TIMER: TTIM(087)/TTIMX(555)

Purpose

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

Note The timer accuracy for CS1D CPU Units is 10 ms + the cycle time

Ladder Symbol

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

Variations

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

Applicable Program Areas

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

Operands

N: Timer Number

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

S: Set Value

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

Operand Specifications

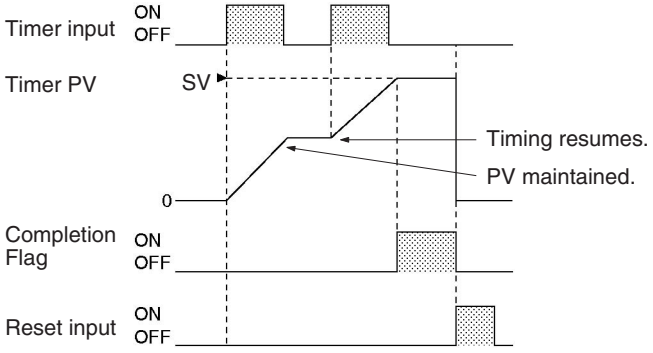
Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	0000 to 4095 (decimal)	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)

Area	N	S
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

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

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



Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the 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.

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 0000 and its Completion Flag is turned OFF.)

Condition	PV	Completion Flag
Operating mode changed from RUN or MONITOR mode to PROGRAM mode or vice versa.[1]	0000	OFF
Power supply interrupted and reset[2]	0000	OFF
Execution of CNR(545)/CNRX(547), the RESET TIMER/COUNTER instructions[3]	BCD: 9999 Binary: FFFF	OFF
Operation in interlocked program section (IL(002)–ILC(003))	Retains previous status.	Retains previous status.
Operation in jumped program section (JMP(004)–JME(005))	Retains previous status.	Retains previous status.

Note

1. If the IOM Hold Bit (A50012) 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 (A50012) has been turned ON and the status of the IOM Hold Bit itself is protected in the PLC Setup, the status of timer Completion Flags and PVs will be maintained even when the power is interrupted.
3. The PV will be set to the SV when TTIM(087)/TTIMX(555) is executed.

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

When an operating TTIM(087)/TTIMX(555) timer is in a program section between JMP(004) and JME(005) and the program section is jumped, the PV will retain its previous value. Be sure to take this fact into account when TTIM(087)/TTIMX(555) is programmed between JMP(004) and JME(005).

When a TTIM(087)/TTIMX(555) timer is forced set, its Completion Flag will be turned ON and its PV will be reset to 0000. When a TTIM(087)/TTIMX(555) timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to 0000. The forced set and forced reset operations take priority over the status of the timer and reset inputs.

The timer's PV is refreshed only when TTIM(087)/TTIMX(555) is executed, so the timer will not operate properly when the cycle time exceeds 100 ms because the timer increments in 100-ms units.

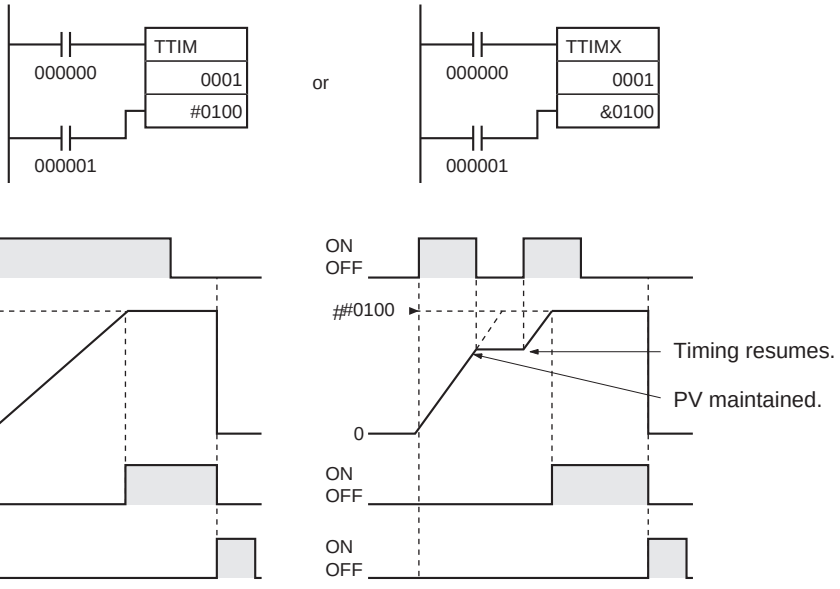
The timer's Completion Flag is refreshed only when TTIM(087)/TTIMX(555) is executed, so a delay of up to one cycle may be required for the Completion Flag to be turned ON after the timer times out.

Typical timers such as TIM/TIMX(550) are decrementing counters and the PV shows the time remaining until the timer times out. The PV of TTIM(087)/TTIMX(555) shows how much time has elapsed, so the PV can be used unchanged in many calculations and display outputs.

Example

When timer input CIO 000000 is ON in the following example, the timer PV will begin counting up from 0. Timer Completion Flag T0001 will be turned ON when the PV reaches the SV.
If the reset input is turned ON, the timer PV will be reset to 0000 and the Completion Flag (T0001) will be turned OFF. (Usually the reset input is turned ON to reset the timer and then the timer input is turned ON to start timing.)

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



3-6-7 LONG TIMER: TIML(542)/TIMLX(553)

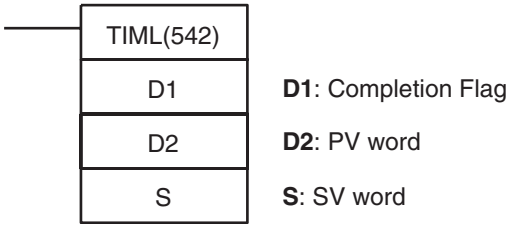
Purpose

TIML(542)/TIMLX(553) operates a decrementing timer with units of 0.1 s that can time up to 115 days for TIML(542) and 4,971 days for TIMLX(543). The timer accuracy is 0 to 0.01 s.

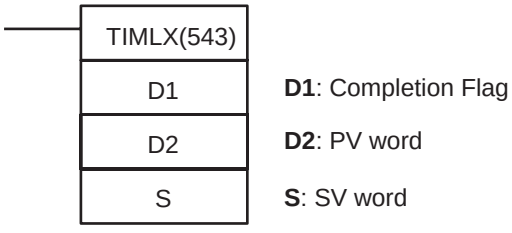
Note The timer accuracy for CS1D CPU Units is 10 ms + the cycle time

Ladder Symbol

BCD



Binary



Variations

Variations		TIML(542)/ TIMLX(553)
	Executed Each Cycle for ON Condition	
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

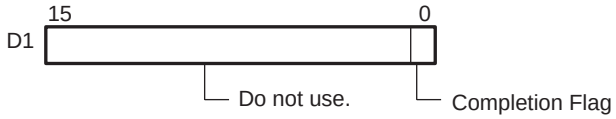
Applicable Program Areas

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

Operands

D1: Completion Flag

Bit 0 of D1 acts as the Completion Flag for TIML(542)/TIMLX(553).



D2: PV Word

D2+1 and D2 contain the 8-digit binary or BCD PV. (D2 and D2+1 must be in the same data area.) The PV can range from #00000000 to #99999999 for TIML(542) and &00000000 to &4294967294 (decimal) or #00000000 to #FFFFFFFF (hexadecimal) for TIMLX(553).

S: SV Word

S+1 and S contain the 8-digit binary or BCD SV. (S and S+1 must be in the same data area.) The SV must be between #00000000 to #99999999 for TIML(542) and &00000000 to &4294967294 (decimal) or #00000000 to #FFFFFFFF (hexadecimal) for TIMLX(553).

Operand Specifications

Area	D1	D2	S
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	
Work Area	W000 to W511	W000 to W510	
Holding Bit Area	H000 to H511	H000 to H510	
Auxiliary Bit Area	A448 to A959	A448 to A958	A000 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	D00000 to D32767	D00000 to D32766	
EM Area without bank	E00000 to E32767	E00000 to E32766	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		

Area	D1	D2	S
Constants	---		BCD: #00000000 to 99999999 (BCD) "&" cannot be used. Binary: &00000000 to &4294967294 (decimal) or #00000000 to #FFFFFFFF (hex)
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15		

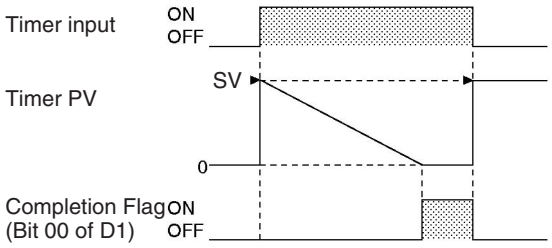
Description

TIML(542)/TIMLX(553) is a decrementing ON-delay timer with units of 0.1-s that uses an 8-digit SV and an 8-digit PV.

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

When the timer input goes from OFF to ON, TIML(542)/TIMLX(553) starts decrementing the PV in D2+1 and D2. The PV will continue timing down as long as the timer input remains ON and the timer's Completion Flag will be turned ON when the PV reaches 0000 0000.

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



Flags

Name	Label	Operation
Error Flag	ER	ON if the PV contained in D2+1 and D2 is not BCD. ON if the SV contained in S+1 and S is not BCD. OFF in all other cases.

Precautions

Unlike most timers, TIML(542)/TIMLX(553) does not use a timer number. (Timer area PV refreshing is not performed for TIML(542)/TIMLX(553).)

Since the Completion Flag for TIML(542)/TIMLX(553) is in a data area it can be forced set or forced reset like other bits, but the PV will not change.

The timer's PV is refreshed only when TIML(542)/TIMLX(553) is executed, so the timer will not operate properly when the cycle time exceeds 100 ms because the timer increments in 100-ms units.

The timer's Completion Flag is refreshed only when TIML(542)/TIMLX(553) is executed, so a delay of up to one cycle may be required for the Completion Flag to be turned ON after the timer times out.

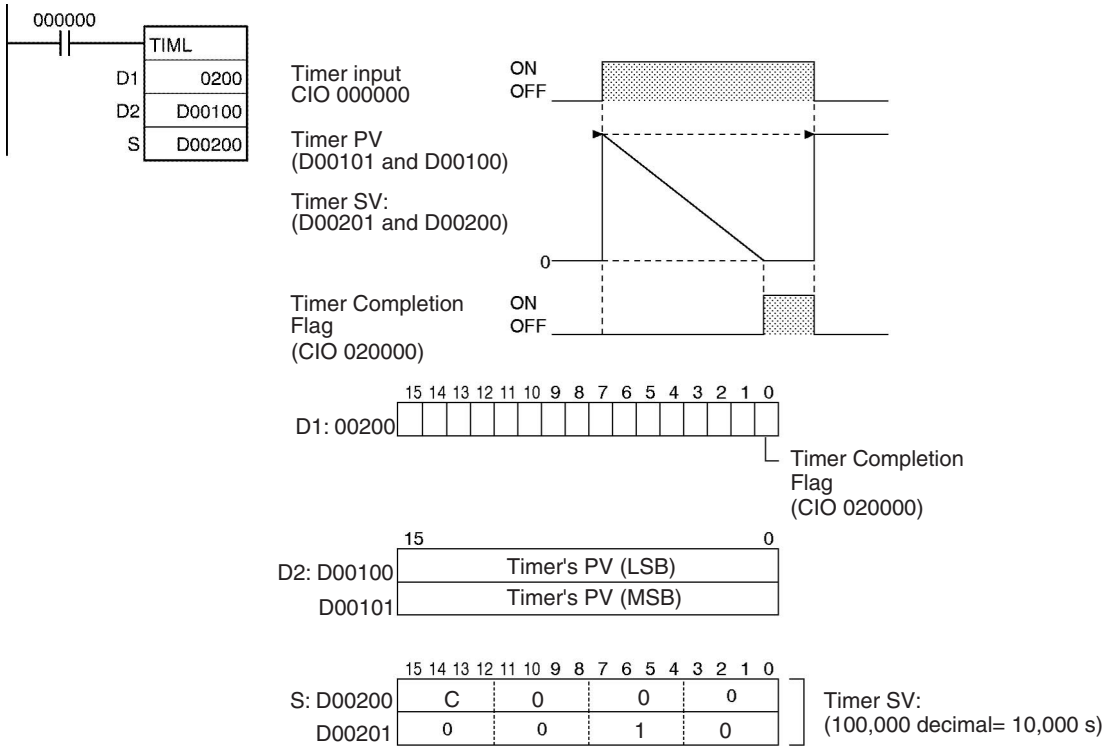
When TIML(542)/TIMLX(553) is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will be reset to the SV and the Completion Flag will be turned OFF.

When an operating TIML(542)/TIMLX(553) timer is in a program section between JMP(004) and JME(005) and the program section is jumped, the PV will retain its previous value. Be sure to take this fact into account when TIML(542)/TIMLX(553) is programmed between JMP(004) and JME(005).

Be sure that the words specified for the Completion Flag and PV (D1, D2, and D2+1) are not used in other instructions. If these words are affected by other instructions, the timer might not time out properly.

Example

When timer input CIO 000000 is ON in the following example, the timer PV (in D00101 and D00100) will be set to the SV (in D00101 and D00100) and the PV will begin counting down. The timer Completion Flag (CIO 020000) will be turned ON when the PV reaches 0000 0000.
When CIO 000000 goes OFF, the timer PV will be reset to the SV and the Completion Flag will be turned OFF.



3-6-8 MULTI-OUTPUT TIMER: MTIM(543)/MTIMX(554)

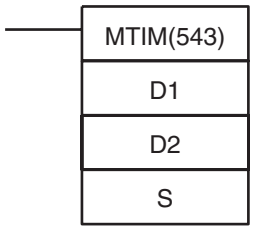
Purpose

MTIM(543)/MTIMX(554) operates a 0.1-s incrementing timer with eight independent SVs and Completion Flags. The set value is 0 to 999.9 s for MTIM(543) and 0 to 6,553.5 s for MTIMX(554), and the timer accuracy is 0 to 0.01 s.

Note The timer accuracy for CS1D CPU Units is 10 ms + the cycle time

Ladder Symbol

BCD

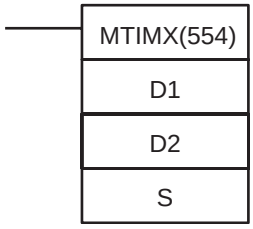


D1: Completion Flags

D2: PV word

S: First SV word

Binary



D1: Completion Flags

D2: PV word

S: First SV word

Variations

Variations		
	Executed Each Cycle for ON Condition	MTIM(543)/ MTIMX(554)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

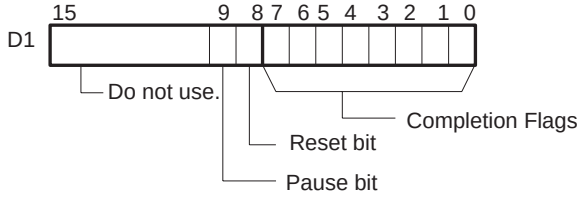
Applicable Program Areas

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

Operands

D1: Completion Flags

D1 contains the eight Completion Flags as well as the pause and reset bits.



D2: PV Word

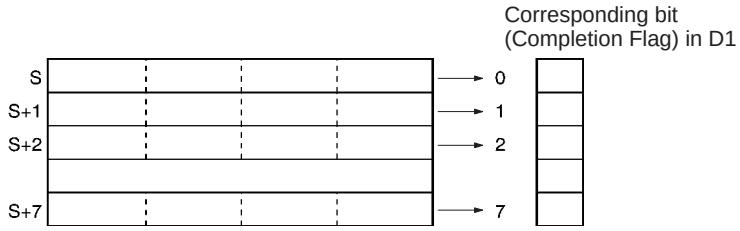
D2 contains the 4-digit binary or BCD PV.

Data	Range
BCD	#0000 to #9999
Binary	&0 to &65535 (decimal) #0000 to #FFFF (hex)

S: First SV Word

S through S+7 contain the eight independent SVs.
Each SV must be as follows:

Data	Range
BCD	#0000 to #9999
Binary	&0 to &65535 (decimal) #0000 to #FFFF (hex)



Data	Range
BCD	One word for each of 8 timer SV: #0000 to #9999
Binary	One word for each of 8 timer SV: &0 to &65535 (decimal) #0000 to #FFFF (hex)

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

Operand Specifications

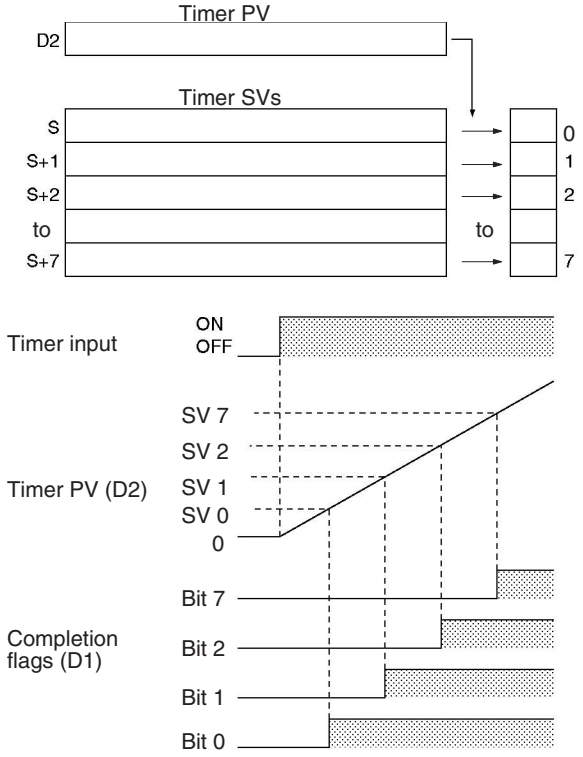
Area	D1	D2	S
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6136
Work Area	W000 to W511		W000 to W504
Holding Bit Area	H000 to H511		H000 to H504
Auxiliary Bit Area	A448 to A959		A000 to A952
Timer Area	T0000 to T4095		T0000 to T4088
Counter Area	C0000 to C4095		C0000 to C4088
DM Area	D00000 to D32767		D00000 to D32760
EM Area without bank	E00000 to E32767		E00000 to E32760
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32760 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---	DR0 to DR15	---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

When the execution condition for MTIM(543)/MTIMX(554) is ON and the reset and timer bits are both OFF, MTIM(543)/MTIMX(554) increments the PV in D2. If the pause bit is turned ON, the timer will stop incrementing the PV, but the PV will retain its value. MTIM(543)/MTIMX(554) will resume timing when the pause bit goes OFF again.

The PV (content of D2) is compared to the eight SVs in S through S+7 each time that MTIM(543)/MTIMX(554) is executed, and if any of the SVs is less than or equal to the PV, the corresponding Completion Flag (D1 bits 00 through 07) is turned ON.

When the PV reaches 9999, the PV will be reset to 0000 and all of the Completion Flags will be turned OFF. If the reset bit is turned ON while the timer is operating or paused, the PV will be reset to 0000 and all of the Completion Flags will be turned OFF.



The following table shows the operation of MTIM(543)/MTIMX(554) for the four possible combinations of the reset and pause bits.

Reset bit (Bit 08)	Pause bit (Bit 09)	Operation
OFF	OFF	The PV will be updated and the corresponding Completion Flag will be turned ON when SV ≤ PV.
	ON	The PV will not be updated and MTIM(543)/MTIMX(554) will be treated as NOP(000).
ON	OFF	The PV will be reset to 0000 and the Completion Flags will be turned OFF. The PV will not be updated.
	ON	

The reset and pause bits are effective only when the execution condition for MTIM(543)/MTIMX(554) is ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if the PV contained in D2 is not BCD. OFF in all other cases.

Precautions

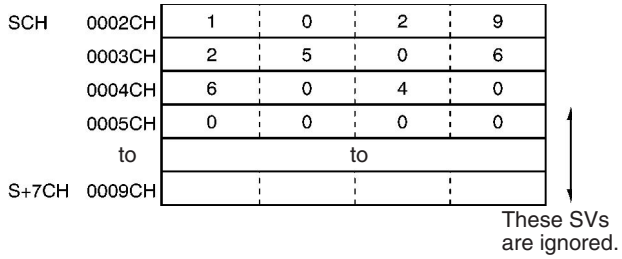
Unlike most timers, MTIM(543)/MTIMX(554) does not use a timer number. (Timer area PV refreshing is not performed for MTIM(543)/MTIMX(554).)

When the PV reaches 9999, the PV will be reset to 0000 and all of the Completion Flags will be turned OFF.

If in BCD mode and an SV in S through S+7 does not contain BCD data, that SV will be ignored. An error will not occur and the Error Flag will not be turned ON.

Since the Completion Flag for MTIM(543)/MTIMX(554) is in a data area it can be forced set or forced reset like other bits, but the PV will not change.

When eight or fewer SVs are required, set the word after the last SV to 0000. MTIM(543)/MTIMX(554) will ignore the SV that is set to 0000 and all of the remaining SVs.



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

The timer's Completion Flag is refreshed only when MTIM(543)/MTIMX(554) is executed, so a delay of up to one cycle may be required for the Completion Flag to be turned ON after the timer times out.

When MTIM(543)/MTIMX(554) is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will retain its previous value (it will not be reset). Be sure to take this fact into account when MTIM(543)/MTIMX(554) is programmed between IL(002) and ILC(003).

When an operating MTIM(543)/MTIMX(554) timer is in a program section between JMP(004) and JME(005) and the program section is jumped, the PV will retain its previous value. Be sure to take this fact into account when MTIM(543)/MTIMX(554) is programmed between JMP(004) and JME(005).

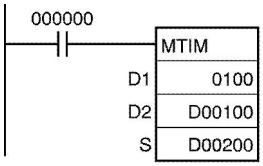
Be sure that the words specified for the Completion Flags and PV (D1 and D2) are not used in other instructions. If these words are affected by other instructions, the timer might not time out properly.

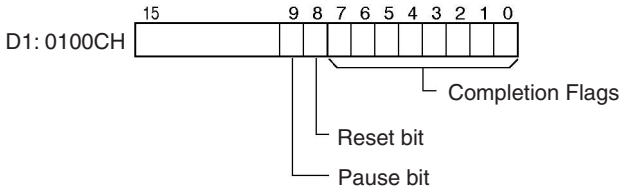
If a word in the CIO area is specified for D1, the SET and RSET instructions can be used to control the pause and reset bits.

Example

When CIO 000000 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 0000.

The eight SVs in D00200 through D00207 are compared to the PV and the corresponding Completion Flags (CIO 010000 through CIO 010007) are turned on when the SV ≤ PV.





Timer PV (D2: D00100)

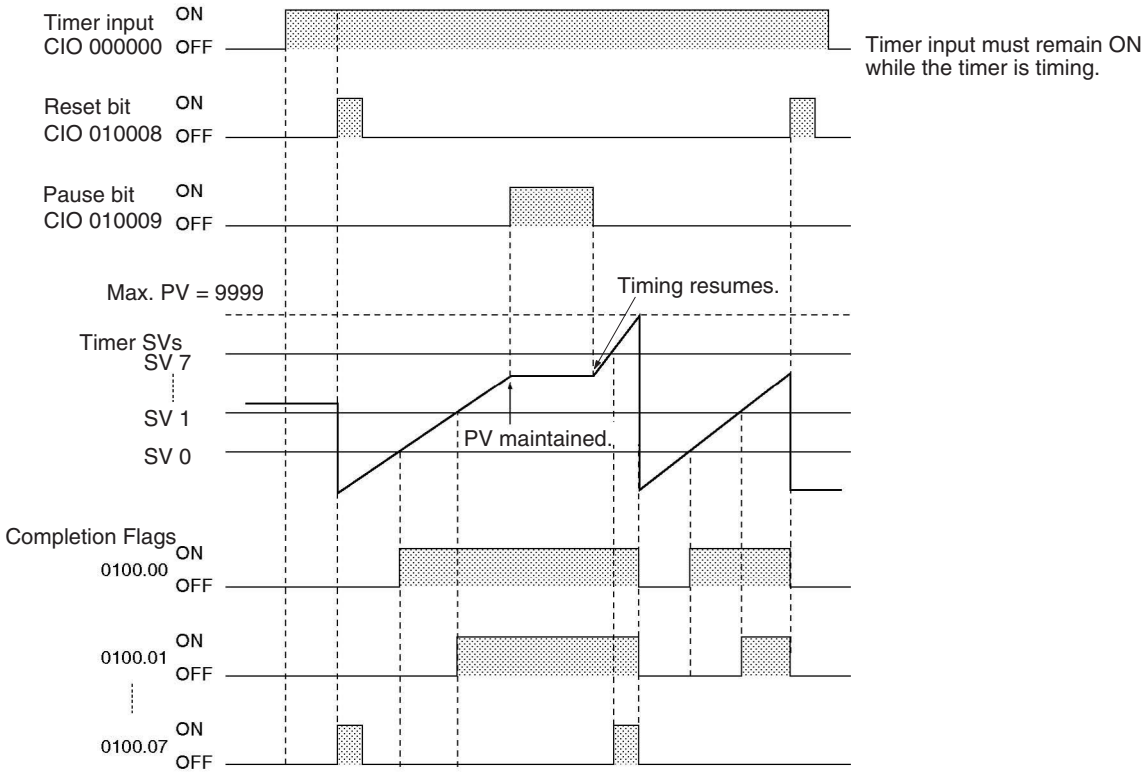
15			0
0	1	0	0

(Incrementing)

Corresponding completion flag ON when SV ≤ PV.

Timer SVs

	15			0		
S: D00200	0	0	8	0	1	0
S+1: D00201	0	0	9	0	1	1
S+2: D00202	0	1	0	0	1	2
S+3: D00203	0	1	1	0	0	3
S+4: D00204	0	1	2	0	0	4
S+5: D00205	0	1	3	0	0	5
S+6: D00206	0	1	5	0	0	6
S+7: D00207	1	0	0	0	0	7



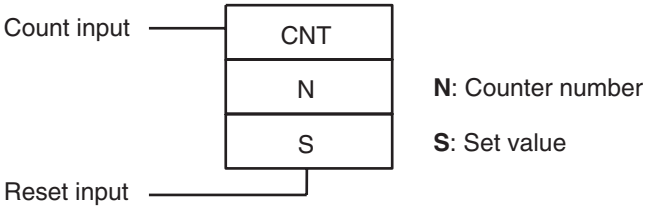
3-6-9 COUNTER: CNT/CNTX(546)

Purpose

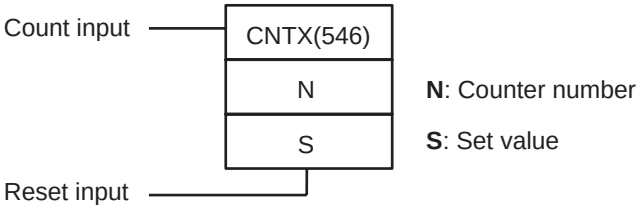
CNT/CNTX(546) operates a decrementing counter. The setting range 0 to 9,999 for CNT and 0 to 65,535 for CNTX(546).

Ladder Symbol

BCD



Binary



Variations

Variations		
	Executed Each Cycle for ON Condition	CNT/ CNTX(546)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Counter Number

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

S: Set Value

Data	Range
BCD	#0000 to #9999
Binary	&0 to &65535 (decimal) #0000 to #FFFF (hex)

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T4095
Counter Area	0000 to 4095 (decimal)	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area with-out bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)

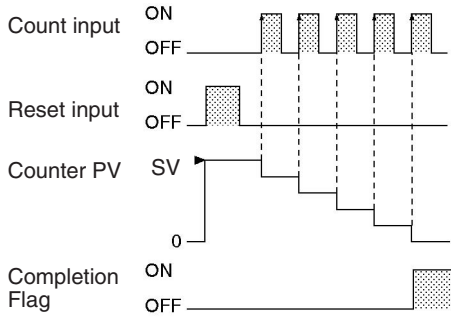
Area	N	S
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

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

Once the Completion Flag is turned ON, reset the counter by turning the reset input ON or by using the CNR(545)/CNRX(547) instruction. Otherwise, the counter cannot be restarted.

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



Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the 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	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

Note In 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 are turned OFF.

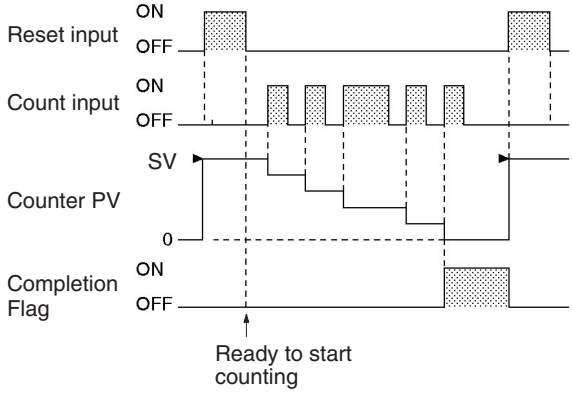
Precautions

Counter numbers are shared by the CNT, CNTX(546), CNTR(012), CNTRX(548), CNTW(814), and CNTWX(818) instructions. If two counters share the same counter number but are not used simultaneously, a duplication error will be generated when the program is checked but the counters will operate normally. Counters which share the same counter number will not operate properly if they are used simultaneously.

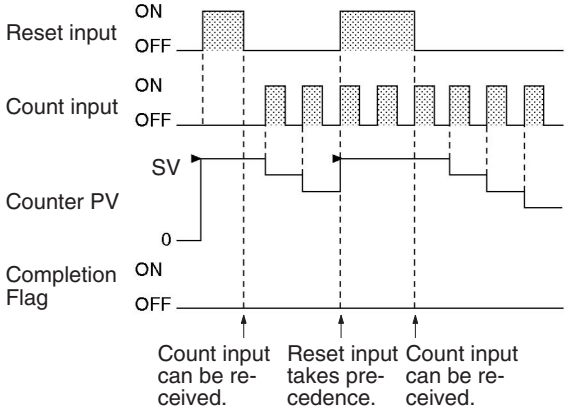
A counter's PV is refreshed when the count input goes from OFF to ON and the Completion Flag is refreshed each time that CNT/CNTX(546) is executed. The Completion Flag is turned ON if the PV is 0 and it is turned OFF if the PV is not 0.

When a CNT/CNTX(546) counter is forced set, its Completion Flag will be turned ON and its PV will be reset to 0000. When a CNT/CNTX(546) counter is forced reset, its Completion Flag will be turned OFF and its PV will be set to the SV.

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



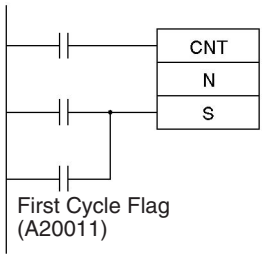
The reset input will take precedence and the counter will be reset if the reset input and count input are both ON at the same time. (The PV will be reset to the SV and the Completion Flag will be turned OFF.)



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

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

Counter PVs are retained even through a power interruption. If you want to restart counting from the SV instead of resuming the count from the retained PV, add the First Cycle Flag (A20011) as a reset input to the counter.



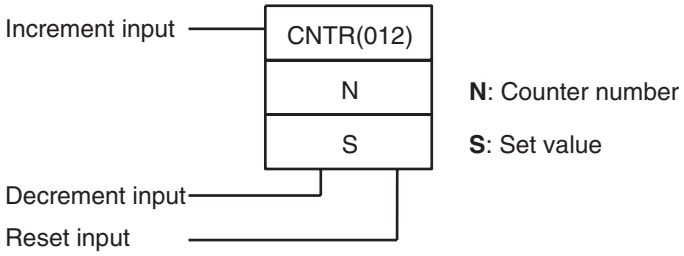
3-6-10 REVERSIBLE COUNTER: CNTR(012)/CNTRX(548)

Purpose

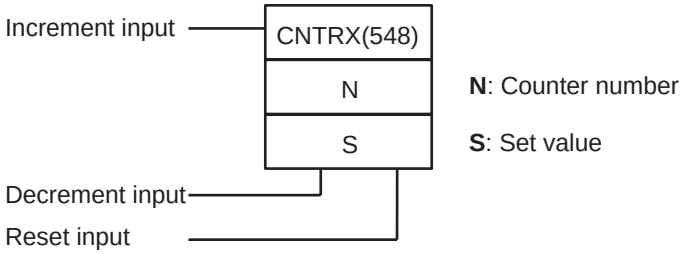
CNTR(012)/CNTRX(548) operates a reversible counter.

Ladder Symbol

BCD



Binary



Variations

Variations		
	Executed Each Cycle for ON Condition	CNTR(012)/ CNTRX(548)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Counter Number

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

S: Set Value

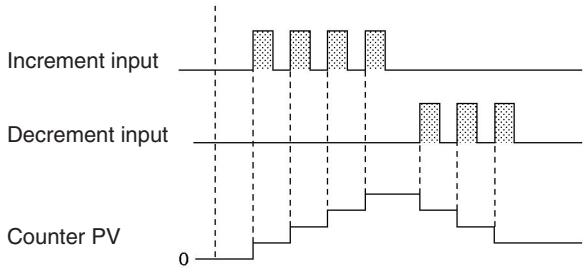
Data	Range
BCD	#0000 to #9999
Binary	&0 to &65535 (decimal) #0000 to #FFFF (hex)

Operand Specifications

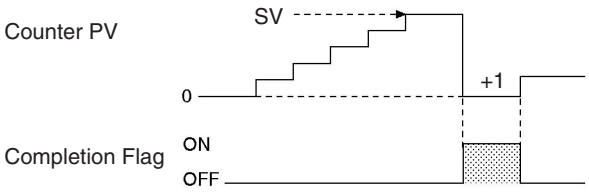
Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T4095
Counter Area	0000 to 4095 (decimal)	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area with-out bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect address-ing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

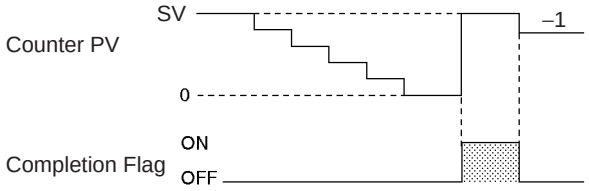
The counter PV is incremented by 1 every time that the increment input goes from OFF to ON and it is decremented by 1 every time that the decrement input goes from OFF to ON. The PV can fluctuate between 0 and the SV.



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



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



Flags

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

Precautions

Counter numbers are shared by the CNT, CNTX(546), CNTR(012), CNTRX(548), CNTW(814), and CNTWX(818) instructions. If two counters share the same counter number but are not used simultaneously, a duplication error will be generated when the program is checked but the counters will operate normally. Counters which share the same counter number will not operate properly if they are used simultaneously.

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

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

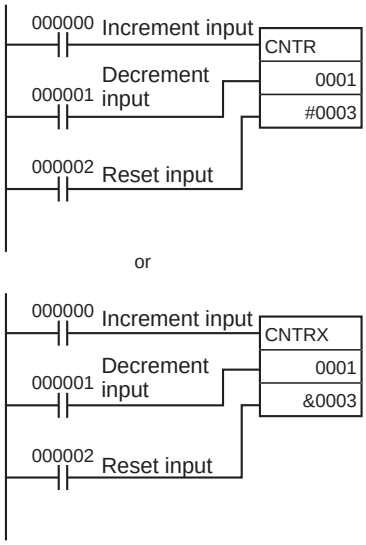
When inputting the CNTR(012)/CNTRX(548) instruction with mnemonics, first enter the increment input (II), then the decrement input (DI), the reset input (R), and finally the CNTR(012)/CNTRX(548) instruction. When entering with the ladder diagrams, first input the increment input (II), then the CNTR(012)/CNTRX(548) instruction, the decrement input (DI), and finally the reset input (R).

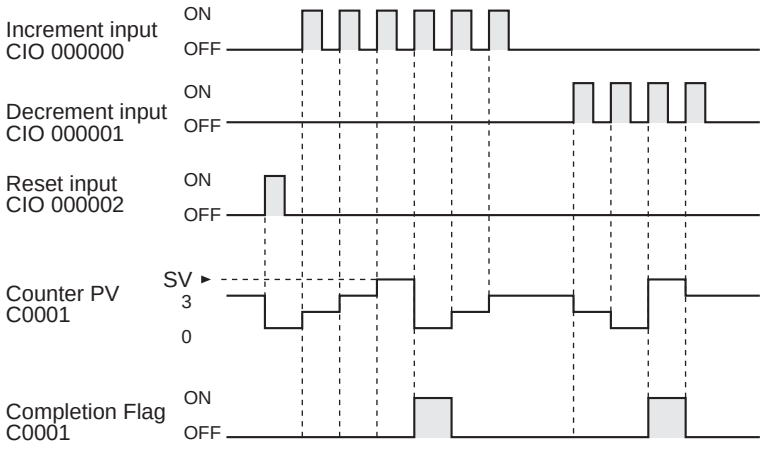
Examples

Basic Operation of CNTR(012)/CNTRX(548)

The counter PV is reset to 0 by turning the reset input (CIO 000002) ON and OFF. The PV is incremented by 1 each time that the increment input (CIO 000000) 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.

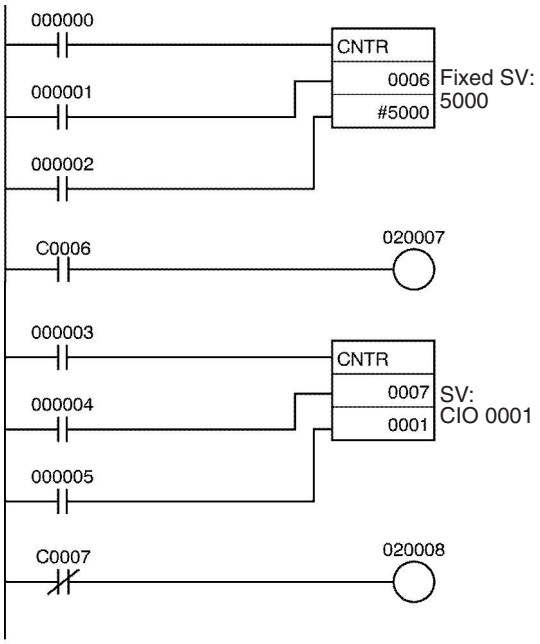
Likewise, the PV is decremented by 1 each time that the decrement input (CIO 000001) goes from OFF to ON. When the PV is decremented from 0, it is automatically set to the SV (3) and the Completion Flag is turned ON.

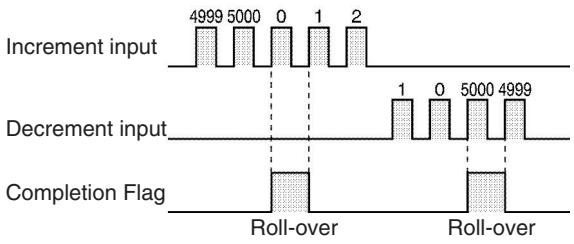




Specifying the SV in a Word

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





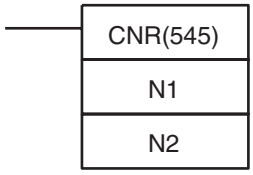
3-6-11 RESET TIMER/COUNTER: CNR(545)/CNRX(547)

Purpose

Resets the timers or counters within the specified range of timer or counter numbers.

Ladder Symbol

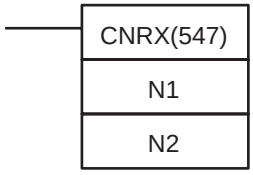
BCD



N_1: First number in range

N_2: Last number in range

Binary



N_1: First number in range

N_2: Last number in range

Variations

Variations		
	Executed Each Cycle for ON Condition	CNR(545)/ CNRX(547)
	Executed Once for Upward Differentiation	@CNR(545)/ CNRX(547)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N_1: First Number in Range

N_1 must be a timer number between T0000 and T4095 or a counter number between C0000 and C4095.

N_2: Last Number in Range

N_2 must be a timer number between T0000 and T4095 or a counter number between C0000 and C4095.

Note N_1 and N_2 must be in the same data area, i.e., N_1 and N_2 must be timer numbers or counter numbers.

Operand Specifications

Area	N_1	N_2
CIO Area	---	---
Work Area	---	---
Holding Bit Area	---	---
Auxiliary Bit Area	---	---
Timer Area	C0000 to C4095	C0000 to C4095
Counter Area	T0000 to T4095	T0000 to T4095
DM Area	---	---
EM Area without bank	---	---
EM Area with bank	---	---
Indirect DM/EM addresses in binary	---	---
Indirect DM/EM addresses in BCD	---	---

Area	N_1	N_2
Constants	---	---
Data Registers	---	---
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

CNR(545)/CNRX(547) resets the Completion Flags of all timers or counters from N_1 to N_2. At the same time, the PVs will all be set to the maximum value (9999 for BCD and FFFF for binary). (The PV will be set to the SV the next time that the timer or counter instruction is executed.)

Operation of CNR(545)

The following table shows the timer and counter instructions (with BCD PVs), which are reset by CNR(545).

Instructions reset	Operation of CNR(545)
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 (TIMU(541) and TMUH(544) are supported by CJ1-H-R CPU Units only.)	The Completion Flag is turned OFF. (The PV cannot be read.)

Operation of CNRX(547)

The following table shows the timer and counter instructions (with binary PVs), which are reset by CNRX(547).

Instructions reset	Operation of CNR(545)
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 (TIMUX(556) and TMUHX(557) are supported by CJ1-H-R CPU Units only.)	The Completion Flag is turned OFF. (The PV cannot be read.)

Flags

Name	Label	Operation
Error Flag	ER	ON if N_1 is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a timer or counter. ON if N_2 is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a timer or counter. ON if N_1 and N_2 are not in the same data area. OFF in all other cases.

Precautions

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.

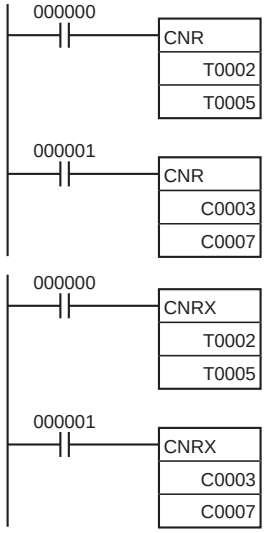
The CNR(545)/CNRX(547) instructions do not reset the timer/counter instructions themselves, they reset the PVs and Completion Flags allocated to those instructions. In most cases, the effect of CNR(545)/CNRX(547) is different from directly resetting the instructions. For example, when a TIM/TIMX(550) 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

When CIO 000000 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 and FFFF for binary).

When CIO 000001 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 and FFFF for binary).



3-6-12 Example Timer and Counter Applications

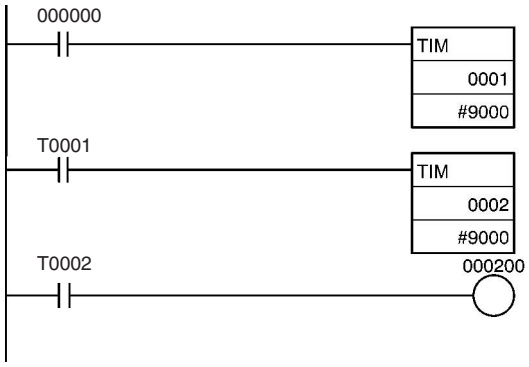
The following examples show various applications of timer and counter instructions including long-term timers, a two-stage counter, ON/OFF delay, one-shot bit, and flicker bit.

Example 1: Long-term Timers

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

Two TIM Instructions

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

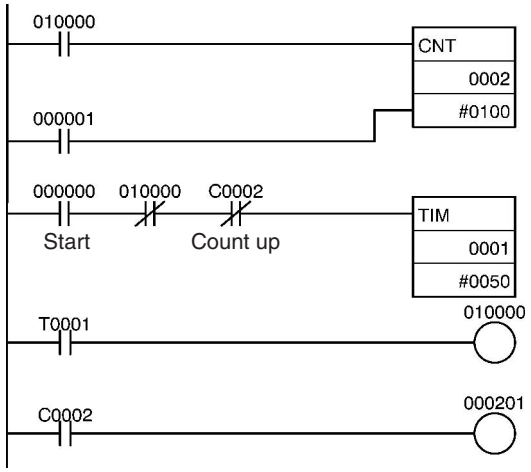


Address	Instruction	Operands
000000	LD	000000
000001	TIM	0001
		#9000
000002	LD	T0001
000003	TIM	0002
		#9000
000004	LD	T0002
000005	OUT	000200

TIM and CNT Instructions

In this example, a TIM instruction and a CNT instruction are combined to make a 500-second timer.

TIM 0001 generates a pulse every 5 s and CNT 0002 counts these pulses. The set value for this combination is the timer interval × counter SV. In this case, the timer SV would be 5 s × 100 = 500 s. With this combination, the long-term timer's PV is actually the PV of a counter, which is maintained through power interruptions.

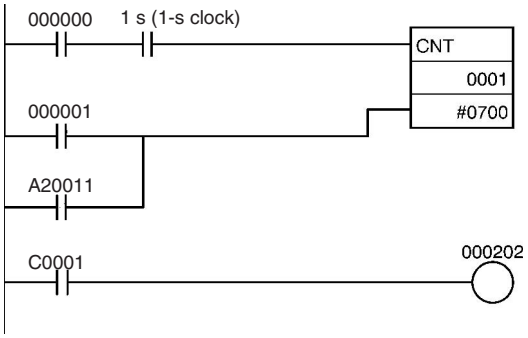


Address	Instruction	Operands
000000	LD	010000
000001	LD	000001
000002	CNT	0002
		#0100
000003	LD	000000
000004	AND NOT	010000
000005	AND NOT	C0002
000006	TIM	0001
		#0050
000007	LD	T0001
000008	OUT	010000
000009	LD	C0002
000010	OUT	000201

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.

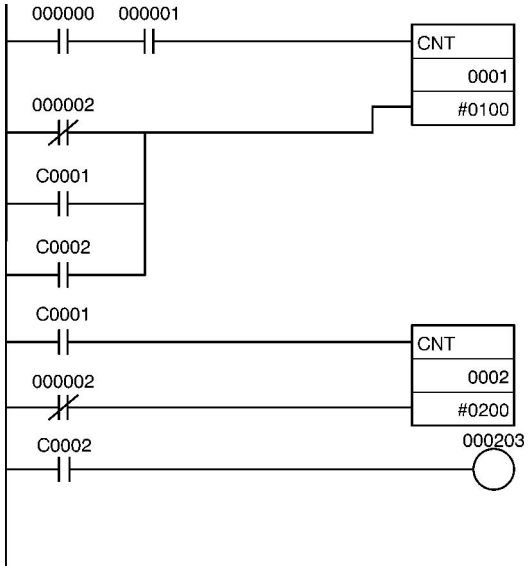
If the First Cycle Flag (A20011) is ORed with the counter's reset input (CIO 000001), the counter's PV will be reset to the SV (0700) when program execution begins rather than resuming the count from the previous PV.



Address	Instruction	Operands
000000	LD	000000
000001	AND	1 s
000002	LD	000001
000003	OR	A20011
000004	CNT	0001
		#0700
000005	LD	C0001
000006	OUT	000202

Example 2: Two-stage Counter

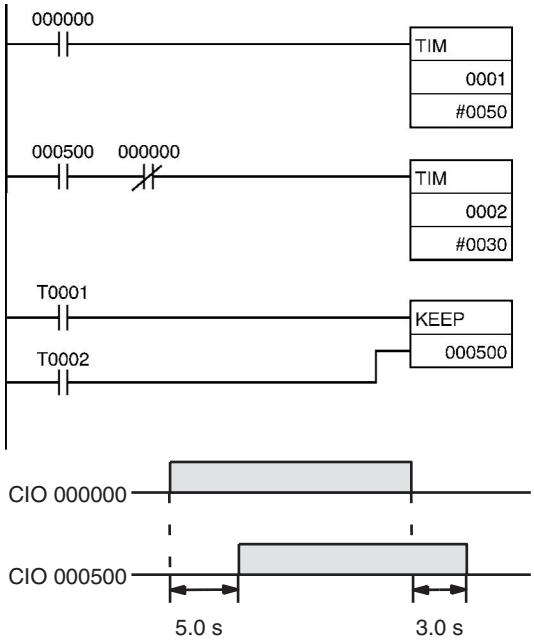
When an SV higher than 9999 is required, two counters can be combined as shown in the following example. In this case, two CNT instructions are combined to make a BCD counter with an SV of 20,000.



Address	Instruction	Operands
000000	LD	000000
000001	AND	000001
000002	LD NOT	000002
000003	OR	C0001
000004	OR	C0002
000005	CNT	0001
		#0100
000006	LD	C0001
000007	LD NO	000002
000008	CNT	0002
		#0200
000009	LD	C0002
000010	OUT	000203

Example 3: ON/OFF Delay

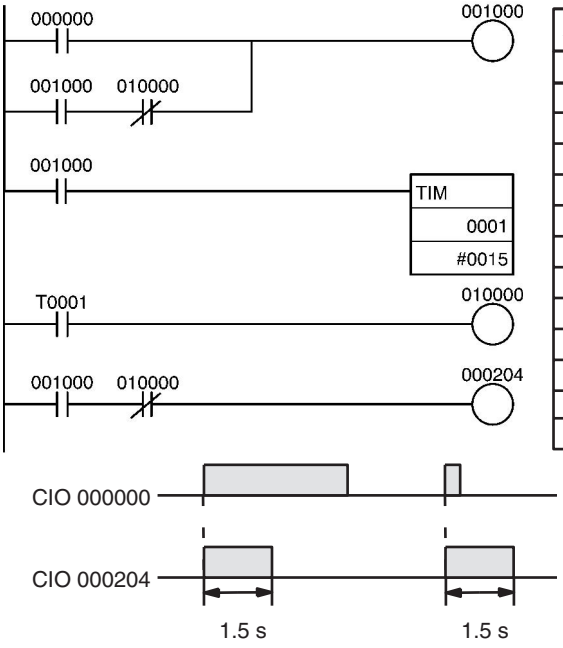
In this example two TIM timers are combined with KEEP(011) to make an ON delay and an OFF delay. CIO 000500 will be turned ON 5.0 seconds after CIO 000000 goes ON and it will be turned OFF 3.0 seconds after CIO 000000 goes OFF.



Address	Instruction	Operands
000000	LD	000000
000001	TIM	0001
		#0050
000002	LD	000500
000003	AND NOT	000000
000004	TIM	0002
		#0030
000005	LD	T0001
000006	LD	T0002
000007	KEEP(011)	000500

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 000204 will be ON for 1.5 seconds (the SV of T0001) after CIO 000000 goes ON.



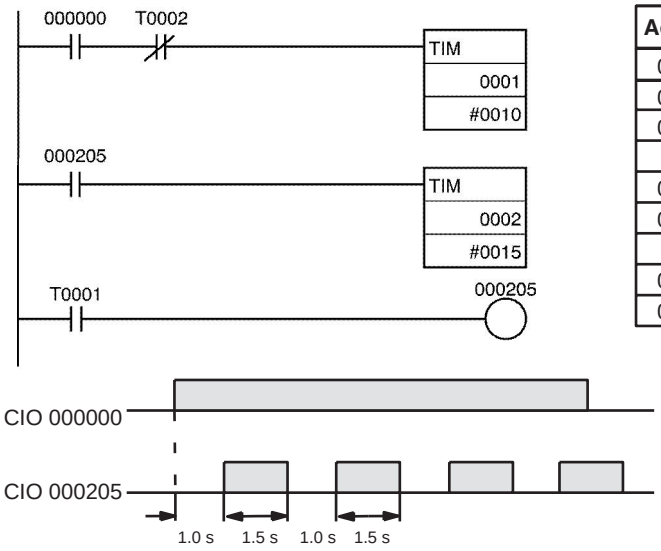
Address	Instruction	Operands
000000	LD	000000
000001	LD	001000
000002	AND NOT	010000
000003	OR	000000
000004	OUT	001000
000005	LD	001000
000006	TIM	0001
		#0015
000007	LD	T0001
000008	OUT	010000
000009	LD	001000
000010	AND NOT	010000
000011	OUT	000204

Example 4: Flicker Bit

The following program examples show two ways to create flicker bits. The second example just mimics a clock pulse.

Two TIM Instructions

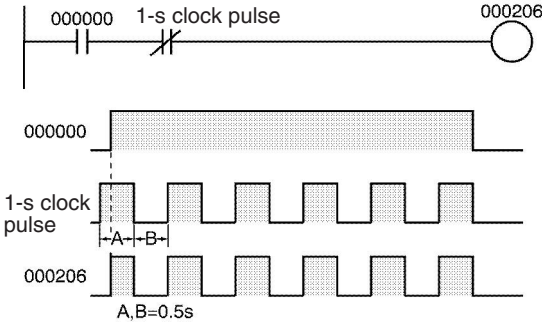
Two TIM timers can be combined to make a bit turn ON and OFF at regular intervals while the execution condition is ON. In this example, CIO 000205 will be OFF for 1.0 second and then ON for 1.5 seconds as long as CIO 000000 is ON.



Address	Instruction	Operands
000000	LD	000000
000001	AND	T0002
000002	TIM	0001
		#0010
000003	LD	000205
000004	TIM	0002
		#0015
000005	LD	T0001
000006	OUT	000205

Clock Pulse

The desired execution condition can be combined with a clock pulse to mimic the clock pulse (0.1 s, 0.2 s, or 1.0 s).



Address	Instruction	Operands
000000	LD	000000
000001	AND	1s
000002	OUT	000206

3-6-13 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.

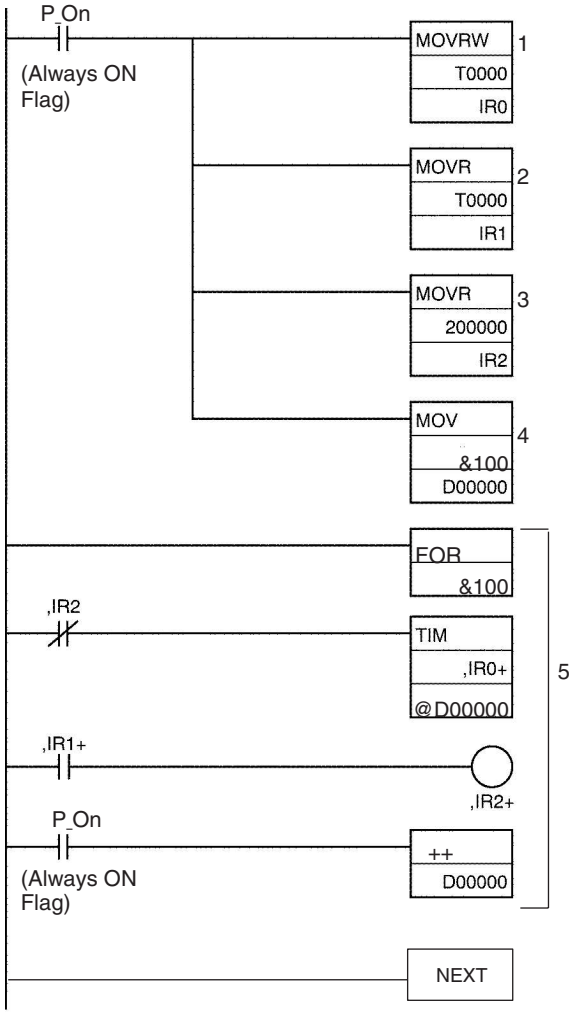
Using Index Registers to indirectly address timers and counters can reduce the size of the program and increase flexibility. For example, common subroutines can be created.

Example

The following example shows a program section that uses indirect addressing to define and start 100 timers with SVs contained in D00100 through D00199.

IR0 contains the PLC memory address of the timer PV and IR1 contains the PLC memory address of the timer Completion Flag.

DM address	Content	Function
D00100	0010	SV for T0000
D00101	0100	SV for T0001
D00102	0050	SV for T0002
. . .	. . .	. . .
D00199	0999	SV for T0099



1,2,3... 1. MOVRW(561) moves the PLC memory address of the PV for timer T0000 to IR0. Afterwards IR0 can be used in place of the timer number.

2. MOVR(560) moves the PLC memory address of the Completion Flag for timer T0000 to IR1.

3. MOVR(560) moves the PLC memory address of CIO 200000 into IR2.

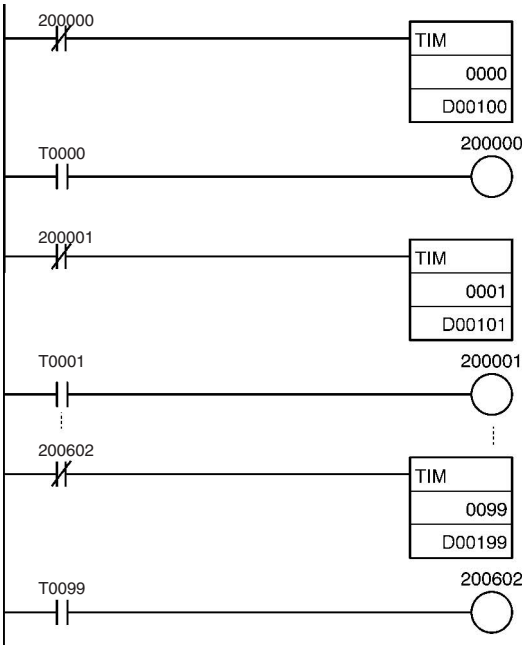
4. MOV(021) moves &100 into D00000 for indirect addressing of the timer SVs.

5. The content of IR0, IR1, IR2, and D00000 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
D00000 The DM address of the word containing the timer's SV

The subroutine above is equivalent to the 400 instructions below.



Address	Instruction	Operands
000000	LD NOT	200000
000001	TIM	0000
		D00100
000002	LD	T0000
000003	OUT	200000
000004	LD NOT	200001
000005	TIM	0001
		D00101
000006	LD	T0001
000007	OUT	200001
000008	LD NOT	200002
000009	TIM	0002
		D00102
000010	LD	T0002
000011	OUT	200002
≈	≈	≈
000396	LD NOT	200602
000397	TIM	0099
		D00199
000398	LD	T0000
000399	OUT	200602

3-7 Comparison Instructions

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

Instruction	Mnemonic	Function code	Page
Input Comparison Instructions	=, <>, <, <=, >, >= (S, L) (LD, AND, OR)	300 to 328	291
Time Comparison Instructions	=DT, <>DT, <DT, <=DT, >DT, >=DT (LD, AND, OR)	341 to 346	297
COMPARE	CMP	020	303
DOUBLE COMPARE	CMPL	060	306
SIGNED BINARY COMPARE	CPS	114	309
DOUBLE SIGNED BINARY COMPARE	CPSL	115	312
MULTIPLE COMPARE	MCMP	019	315
TABLE COMPARE	TCMP	085	317
BLOCK COMPARE	BCMP	068	320
EXPANDED BLOCK COMPARE	BCMP2	502	322
AREA RANGE COMPARE	ZCP	088	326
DOUBLE AREA RANGE COMPARE	ZCPL	116	329

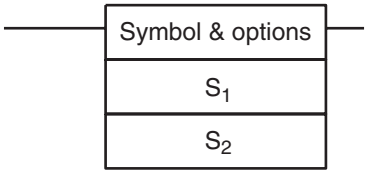
3-7-1 Input Comparison Instructions (300 to 328)

Purpose

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

Note Refer to *3-15-24 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 Instructions* for details on double-precision floating-point input comparison instructions.

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations	Creates ON Each Cycle Comparison is True	Input comparison instruction
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications for Instructions for One-word Data

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	

Area	S_1	S_2
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_ 32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF (binary)	
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Operand Specifications for Instructions for Double-length Data

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	
Data Registers	---	

Area	S_1	S_2
Index Registers	IR0 to IR15 (for unsigned data only)	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

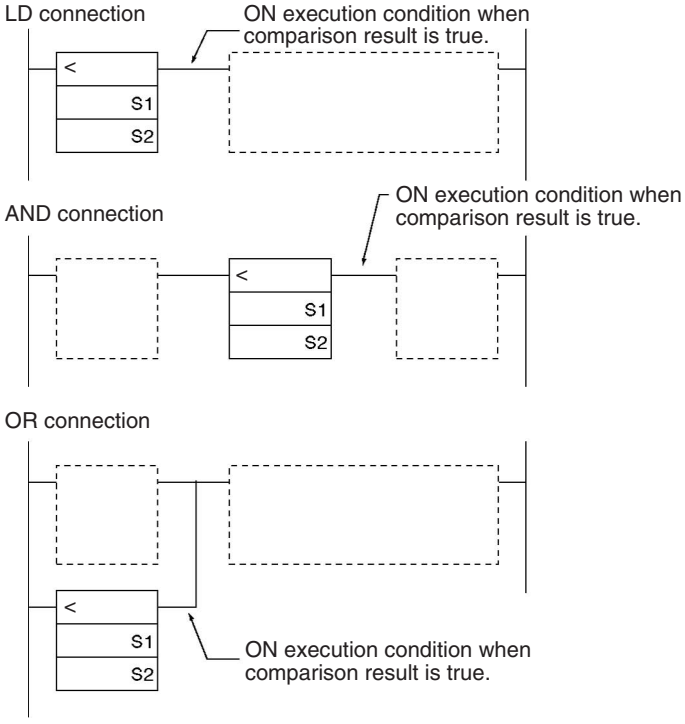
Description

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

Inputting the Instructions

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

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



Options

The input comparison instructions can compare signed or unsigned data and they can compare one-word or double values. If no options are specified, the

comparison will be for one-word unsigned data. With the three input types and two options, there are 72 different input comparison instructions.

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

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

Summary of Input Comparison Instructions

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

Code	Mnemonic	Name	Function
300	LD=	LOAD EQUAL	True if C1 = C2
	AND=	AND EQUAL	
	OR=	OR EQUAL	
301	LD=L	LOAD DOUBLE EQUAL	
	AND=L	AND DOUBLE EQUAL	
	OR=L	OR DOUBLE EQUAL	
302	LD=S	LOAD SIGNED EQUAL	
	AND=S	AND SIGNED EQUAL	
	OR=S	OR SIGNED EQUAL	
303	LD=SL	LOAD DOUBLE SIGNED EQUAL	
	AND=SL	AND DOUBLE SIGNED EQUAL	
	OR=SL	OR DOUBLE SIGNED EQUAL	
305	LD<>	LOAD NOT EQUAL	True if C1 ≠ C2
	AND<>	AND NOT EQUAL	
	OR<>	OR NOT EQUAL	
306	LD<>L	LOAD DOUBLE NOT EQUAL	
	AND<>L	AND DOUBLE NOT EQUAL	
	OR<>L	OR DOUBLE NOT EQUAL	
307	LD<>S	LOAD SIGNED NOT EQUAL	
	AND<>S	AND SIGNED NOT EQUAL	
	OR<>S	OR SIGNED NOT EQUAL	
308	LD<>SL	LOAD DOUBLE SIGNED NOT EQUAL	
	AND<>SL	AND DOUBLE SIGNED NOT EQUAL	
	OR<>SL	OR DOUBLE SIGNED NOT EQUAL	

Code	Mnemonic	Name	Function
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	
325	LD>=	LOAD GREATER THAN OR EQUAL	True if C1 ≥ C2
	AND>=	AND GREATER THAN OR EQUAL	
	OR>=	OR GREATER THAN OR EQUAL	
326	LD>=L	LOAD DOUBLE GREATER THAN OR EQUAL	
	AND>=L	AND DOUBLE GREATER THAN OR EQUAL	
	OR>=L	OR DOUBLE GREATER THAN OR EQUAL	
327	LD>=S	LOAD SIGNED GREATER THAN OR EQUAL	
	AND>=S	AND SIGNED GREATER THAN OR EQUAL	
	OR>=S	OR SIGNED GREATER THAN OR EQUAL	
328	LD>=SL	LOAD DBL SIGNED GREATER THAN OR EQUAL	
	AND>=SL	AND DBL SIGNED GREATER THAN OR EQUAL	
	OR>=SL	OR DBL SIGNED GREATER THAN OR EQUAL	

Flags

Name	Label	Operation
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.
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 CS1 and CJ1 CPU Units, these Flags are turned OFF.
In CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, these Flags are left unchanged.

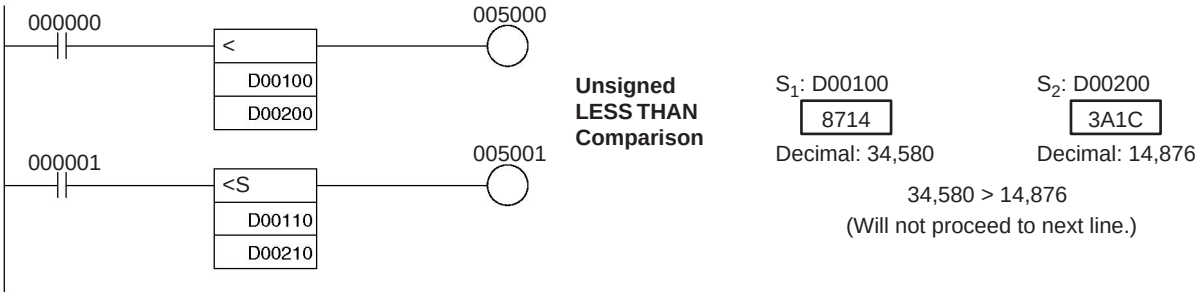
Precautions

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

Examples

AND LESS THAN: AND<(310)

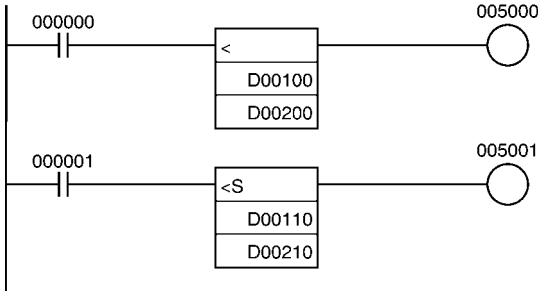
When CIO 000000 is ON in the following example, the contents of D00100 and D00200 are compared in as unsigned binary data. If the content of D00100 is less than that of D00200, CIO 005000 is turned ON and execution proceeds to the next line. If the content of D00100 is not less than that of D00200, the remainder of the instruction line is skipped and execution moves to the next instruction line.



AND SIGNED LESS THAN: AND<S(312)

When CIO 000001 is ON in the following example, the contents of D00110 and D00210 are compared as signed binary data. If the content of D00110 is less than that of D00210, CIO 005001 is turned ON and execution proceeds to the next line. If the content of D00110 is not less than that of D00210, the

remainder of the instruction line is skipped and execution moves to the next instruction line.



Signed LESS THAN Comparison

S_1: D00110 8714 Decimal: –30,956

S_2: D00210 3A1C Decimal: 14,876

–30,956 < 14,876
(Will proceed to next line.)

3-7-2 Time Comparison Instructions (341 to 346)

Purpose

Time comparison instructions compare two BCD time values and create an ON execution condition when the comparison condition is true.

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

These instructions are supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol

LD

Symbol	
C	C: Control word
S_1	S_1: First word of present time
S_2	S_2: First word of comparison time

AND

Symbol	
C	C: Control word
S_1	S_1: First word of present time
S_2	S_2: First word of comparison time

OR

Symbol	
C	C: Control word
S_1	S_1: First word of present time
S_2	S_2: First word of comparison time

Variations

Variations	**Creates ON Each Cycle Comparison is True**	Time comparison instruction
Immediate Refreshing Specification		Not supported

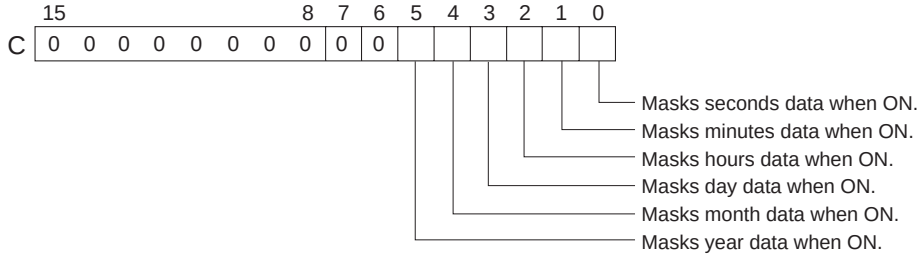
Applicable Program Areas

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

Operands

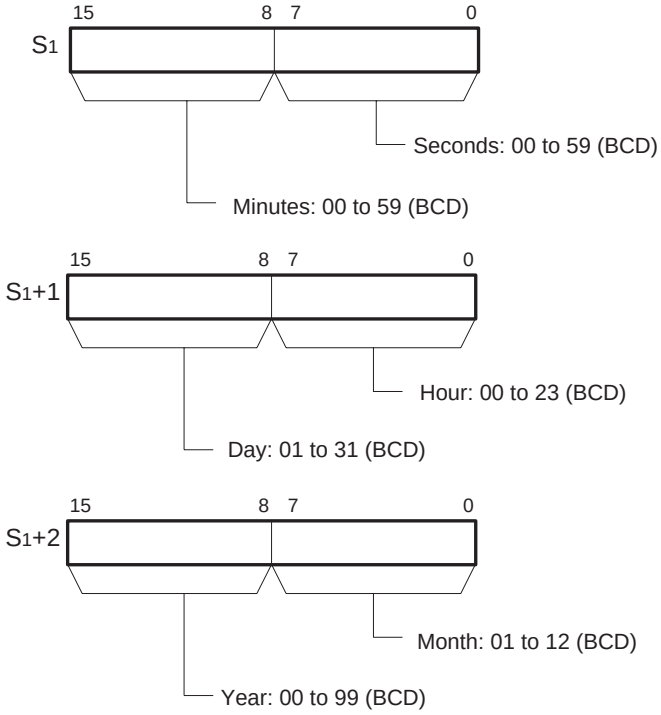
C: Control Word

Bits 00 to 05 of C specify whether or not the time data will be masked for the comparison. Bits 00 to 05 mask the seconds, minutes, hours, day, month, and year, respectively. If all 6 values are masked, the instruction will not be executed, the execution condition will be OFF, and the Error Flag will be turned ON.



S_1 through S_1+2: Present Time Data

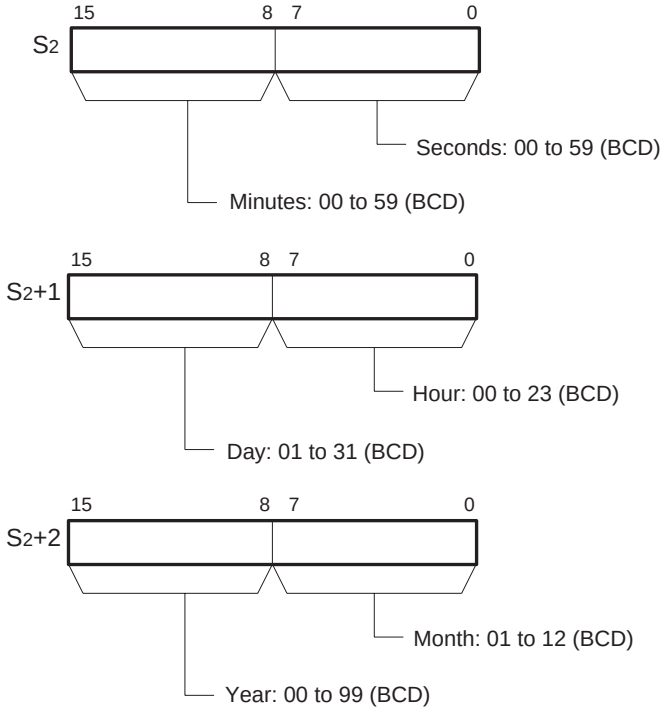
S_1 through S_1+2 contain the present time data. S_1 through S_1+2 must be in the same data area.



Note When using the CPU Unit's internal clock data for the comparison, set S_1 to A351 to specify the CPU Unit's internal clock data (A351 to A353).

S_2 through S_2+2: Comparison Time Data

S_2 through S_2+2 contain the comparison time data. S_2 through S_2+2 must be in the same data area.



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	C	S_1	S_2
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6141	
Work Area	W000 to W511	W000 to W509	
Holding Bit Area	H000 to H511	H000 to H509	
Auxiliary Bit Area	A448 to A959	A000 to A957	
Timer Area	T0000 to T4095	T0000 to T4093	
Counter Area	C0000 to C4095	C0000 to C4093	
DM Area	D00000 to D32767	D00000 to D32765	
EM Area without bank	E00000 to E32767	E00000 to E32765	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32765 (n = 0 to C)	
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	

Area	C	S_1	S_2
Constants	See previous page.	See previous page.	---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

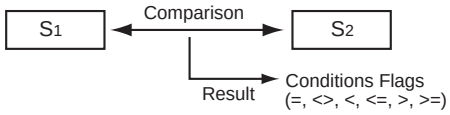
The time comparison instruction compares the unmasked values (corresponding bit of C set to 0) of the present time data in S_1 to S_1+2 with the comparison time data in S_2 to S_2+2 and creates an ON execution condition when the comparison condition is true. At the same time, the result of a time comparison instruction is reflected in the arithmetic flags (=, <>, <, <=, >, >=).

There are 18 possible combinations of time comparison instructions.

Any time values that are masked in the control word (C) are not included in the comparison.

The following table shows the ON/OFF status of each flag for each comparison result.

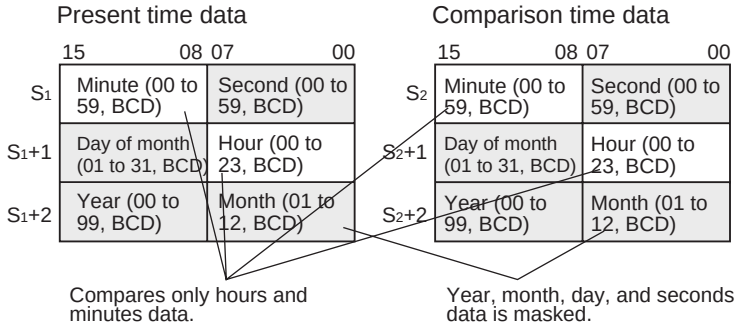
Result	Flag status					
	=	<>	<	<=	>	>=
$S_1 = S_2$	ON	OFF	OFF	ON	OFF	ON
$S_1 > S_2$	OFF	ON	OFF	OFF	ON	ON
$S_1 < S_2$	OFF	ON	ON	ON	OFF	OFF



Masking Time Values

Time values can be masked individually and excluded from the comparison operation. To mask a time value, set the corresponding bit in the control word (C) to 1. Bits 00 to 05 of C mask the seconds, minutes, hours, day, month, and year, respectively.

Example:
When C = 39 hex, the rightmost 6 bits are 111001 (year=1, month=1, day=1, hours=0, minutes=0, and seconds=1) so only the hours and minutes are compared. This mask setting can be used to perform a particular operation at a given time (hour and minute) each day.



Previous data comparison instructions compared data in 16-bit units. The time comparison instructions are limited to comparing 8-bit time values.

The following table shows the structure of the CPU Unit's internal Calendar/Clock Area.

Addresses	Contents
A35100 to A35107	Second (00 to 59, BCD)
A35108 to A35115	Minute (00 to 59, BCD)
A35200 to A35207	Hour (00 to 23, BCD)
A35208 to A35215	Day of month (01 to 31, BCD)
A35300 to A35307	Month (01 to 12, BCD)
A35308 to A35315	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).

Summary of Time Comparison Instructions

The following table shows the function codes, mnemonics, names, and functions of the 18 time comparison instructions.

Code	Mnemonic	Name	Function
341	LD= DT	LOAD EQUAL	True if S1 = S2
	AND=DT	AND EQUAL	
	OR=DT	OR EQUAL	
342	LD<>DT	LOAD NOT EQUAL	True if S1 ≠ S2
	AND<>DT	AND NOT EQUAL	
	OR<>DT	OR NOT EQUAL	
343	LD<DT	LOAD LESS THAN	True if S1 < S2
	AND<DT	AND LESS THAN	
	OR<DT	OR LESS THAN	
344	LD<=DT	LOAD LESS THAN OR EQUAL	True if S1 ≤ S2
	AND<=DT	AND LESS THAN OR EQUAL	
	OR<=DT	OR LESS THAN OR EQUAL	
345	LD>DT	LOAD GREATER THAN	True if S1 > S2
	AND>DT	AND GREATER THAN	
	OR>DT	OR GREATER THAN	
346	LD>=DT	LOAD GREATER THAN OR EQUAL	True if S1 ≥ S2
	AND>=DT	AND GREATER THAN OR EQUAL	
	OR>=DT	OR GREATER THAN OR EQUAL	

Flags

Name	Label	Operation
Error Flag	ER	ON if all 6 of the mask bits (C bits 00 to 05) are ON. OFF in all other cases.
Greater Than Flag	>	ON if $S_1 > S_2$. OFF in all other cases.
Greater Than or Equal Flag	> =	ON if $S_1 \geq S_2$. OFF in all other cases.
Equal Flag	=	ON if $S_1 = S_2$. OFF in all other cases.
Not Equal Flag	=	ON if $S_1 \neq S_2$. OFF in all other cases.
Less Than Flag	<	ON if $S_1 < S_2$. OFF in all other cases.
Less Than or Equal Flag	< =	ON if $S_1 \leq S_2$. OFF in all other cases.
Negative Flag	N	Unchanged (See note.)

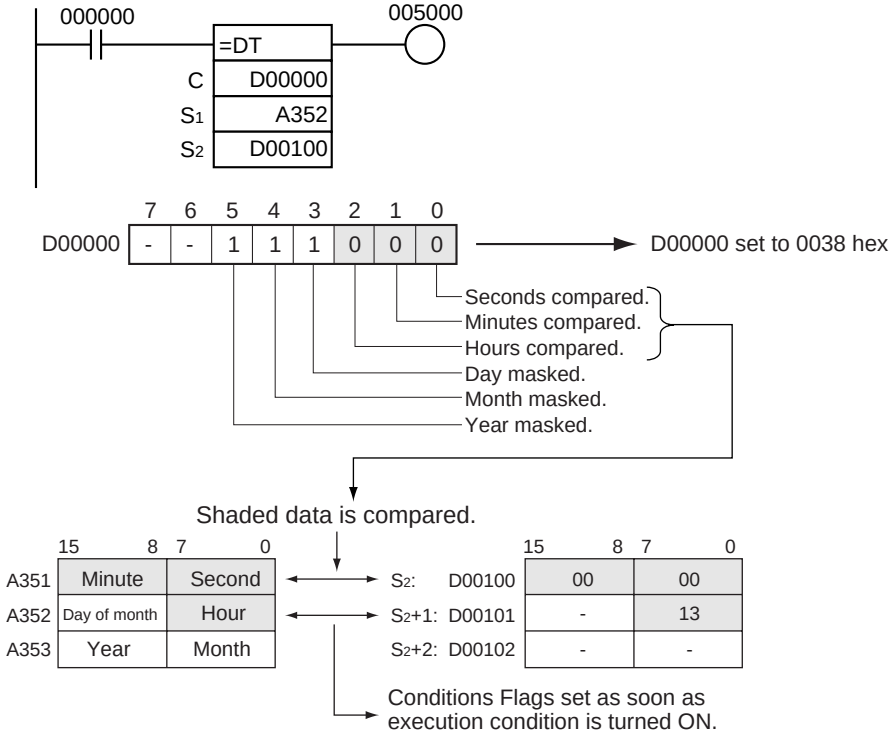
Note In CS1-H, CJ1-H, CJ1M, and CS1D (for Single-CPU System) CPU Units, these Flags are left unchanged.
In CS1 and CJ1 CPU Units, these Flags are turned OFF.

Precautions

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

Example

When CIO 000000 is ON and the time is 13:00:00, CIO 005000 is turned ON. The contents of A351 to A353 (the CPU Unit's internal calendar/clock data) are used as the present time data and the contents of D00100 to D00102 are used as the comparison time data. The year, month, and day values are masked, so only the hour, minute, and second data are compared.

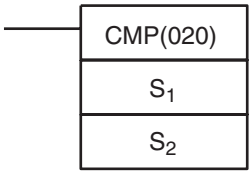


3-7-3 COMPARE: CMP(020)

Purpose

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

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations		
	Executed Each Cycle for ON Condition	CMP(020)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification (See note.)		!CMP(020)

Note Immediate refreshing is not supported by CS1D CPU Units for Duplex-CPU Systems.

Applicable Program Areas

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

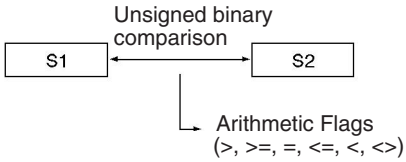
Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF (binary)	
Data Registers	DR0 to DR15	

Area	S_1	S_2
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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



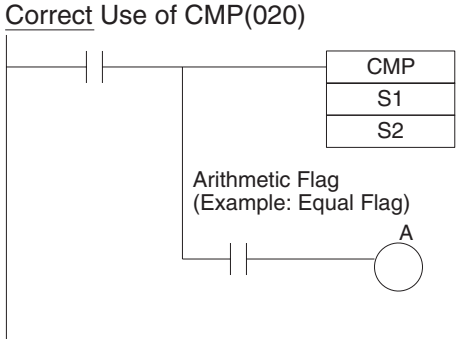
Condition Flag Status

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

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

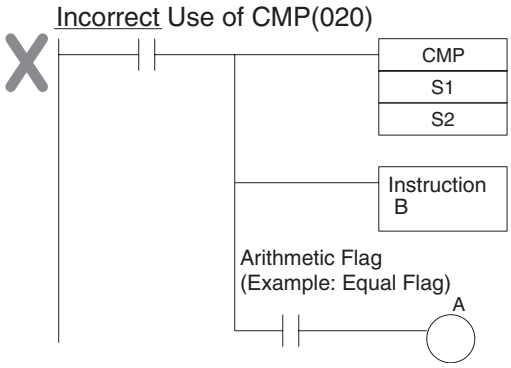
Using CMP(020) Results in the Program

When CMP(020) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls CMP(020), as shown in the following diagram. In this case, the Equals Flag and output A will be turned ON when $S_1 = S_2$.



Using CMP(020) Results in the Program

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



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.)

Flags

Name	CX-Programmer label	Programming Console label	Operation
Error Flag	P_ER	ER	Unchanged (See note.)
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.
Negative Flag	P_N	N	Unchanged (See note.)

Note In 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.

Precautions

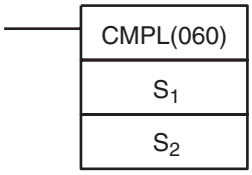
Do not program another instruction between CMP(020) and an input condition that accesses the result of CMP(020) because the other instruction might change the status of the Arithmetic Flags.

3-7-4 DOUBLE COMPARE: CMPL(060)

Purpose

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

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

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

Applicable Program Areas

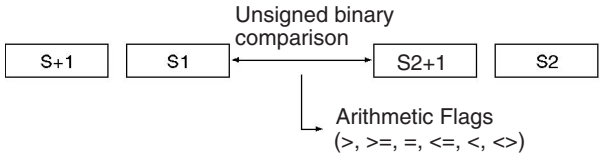
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	
Data Registers	---	
Index Registers	IR0 to IR15	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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



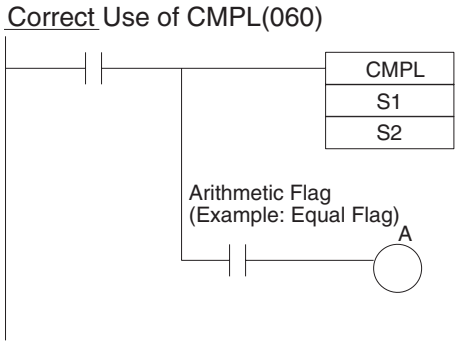
Arithmetic Flag Status

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

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

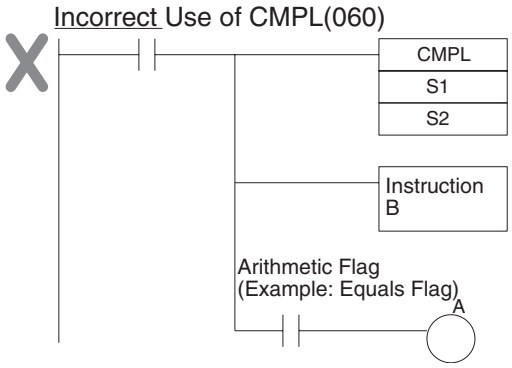
Using CMPL(060) Results in the Program

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



Using CMPL(060) Results in the Program

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



Flags

Name	CX-Programmer label	Programming Console label	Operation
Error Flag	P_ER	ER	Unchanged (See note.)
Greater Than Flag	P_GT	>	ON if S_1 +1, S_1 > S_2+1, S_2. OFF in all other cases.
Greater Than or Equal Flag	P_GE	> =	ON if S_1 +1, $S_1 \geq S_2$+1, S_2. OFF in all other cases.
Equal Flag	P_EQ	=	ON if S_1 +1, S_1 = S_2+1, S_2. OFF in all other cases.
Not Equal Flag	P_NE	<>	ON if S_1 +1, $S_1 \neq S_2$+1, S_2. OFF in all other cases.
Less Than Flag	P_LT	<	ON if S_1 +1, S_1 < S_2+1, S_2. OFF in all other cases.
Less Than or Equal Flag	P_LE	< =	ON if S_1 +1, $S_1 \leq S_2$+1, S_2. OFF in all other cases.
Negative Flag	P_N	N	Unchanged (See note.)

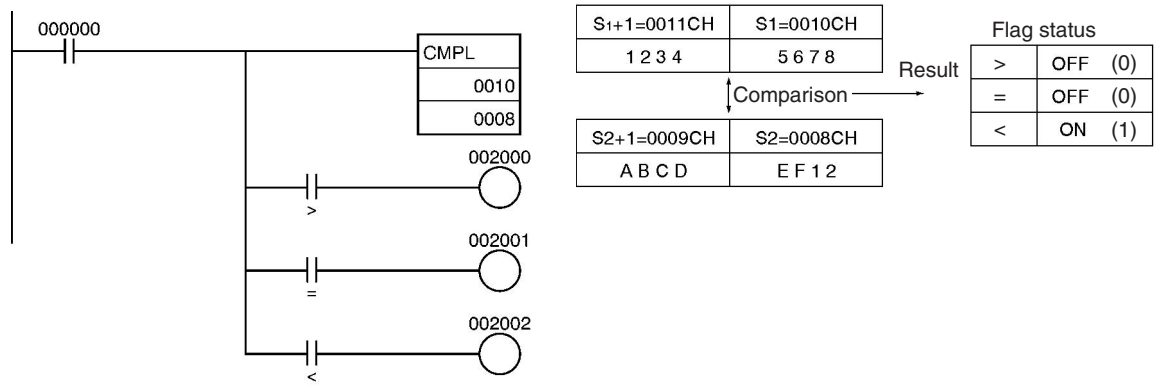
Note In CS1-H, CJ1-H, CJ1M, and CS1D (for Single-CPU System) CPU Units, these Flags are left unchanged.
In CS1 and CJ1 CPU Units, these Flags are turned OFF.

Precautions

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

Example

When CIO 000000 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 000200 (Greater Than), CIO 000201 (Equals), and CIO 000202 (Less Than).

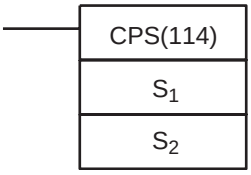


3-7-5 SIGNED BINARY COMPARE: CPS(114)

Purpose

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

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations		
	Executed Each Cycle for ON Condition	CPS(114)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification (See note.)		!CPS(114)

Note Immediate refreshing is not supported by CS1D CPU Units.

Applicable Program Areas

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

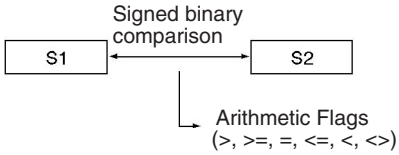
Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	

Area	S_1	S_2
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF (binary)	
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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



Note CPS(114) treats the data in S_1 and S_2 as signed binary data which ranges from 8000 to 7FFF (–32,768 to 32,767 decimal).

Arithmetic Flag Status

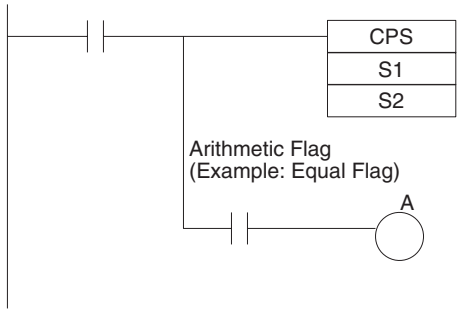
The following table shows the status of the Arithmetic Flags after execution of CPS(114). (A status of "---" indicates that the Flag may be ON or OFF.)

CPS(114) Result	Flag status					
	>	> =	=	< =	<	< >
$S_1 > S_2$	ON	ON	OFF	OFF	OFF	ON
$S_1 = S_2$	OFF	ON	ON	ON	OFF	OFF
$S_1 < S_2$	OFF	OFF	OFF	ON	ON	ON

Using CPS(114) Results in the Program

When CPS(114) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls CPS(114), as shown in the following diagram. In this case, the Equals Flag and output A will be turned ON when $S_1 = S_2$.

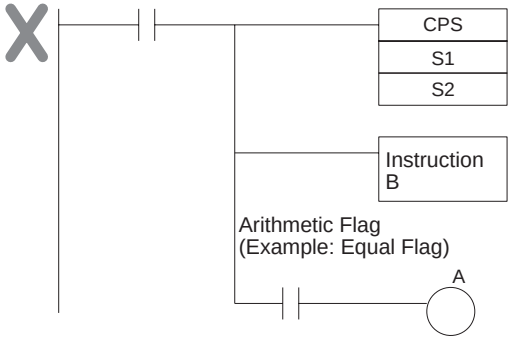
Correct Use of CPS(114)



Using CPS(114) Results in the Program

Do not program another instruction between CPS(114) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag. In this case, the results of instruction B might change the results of CPS(114).

Incorrect Use of CPS(114)



The immediate-refreshing variation (!CPS(114)) can be used with words allocated to external inputs specified in S_1 and/or S_2. When !CPS(114) is executed, input refreshing will be performed for the external input word specified in S_1 and/or S_2 and that refreshed value will be compared. (Immediate refreshing cannot be performed on inputs allocated to Group-2 High-density I/O Units or Units mounted to Slave Racks.)

Flags

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

Note In 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.

Precautions

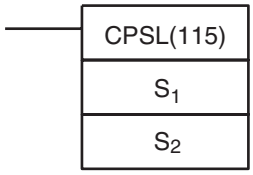
Do not program another instruction between CPS(114) and an input condition that accesses the result of CPS(114) because the other instruction might change the status of the Arithmetic Flags.

3-7-6 DOUBLE SIGNED BINARY COMPARE: CPSL(115)

Purpose

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

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations		
	Executed Each Cycle for ON Condition	CPSL(115)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

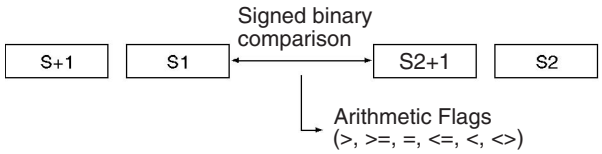
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

CPSL(115) compares the double signed binary data in S_1 +1, S_1 and S_2+1, S_2 and outputs the result to Arithmetic Flags (the Greater Than, Greater Than or Equal, Equal, Less Than or Equal, Less Than, and Not Equal Flags) in the Auxiliary Area.



Note CPSL(115) treats the data in S_1 and S_2 as double signed binary data which ranges from 8000 0000 to 7FFF FFFF (–2,147,483,648 to 2,147,483,647 decimal).

Arithmetic Flag Status

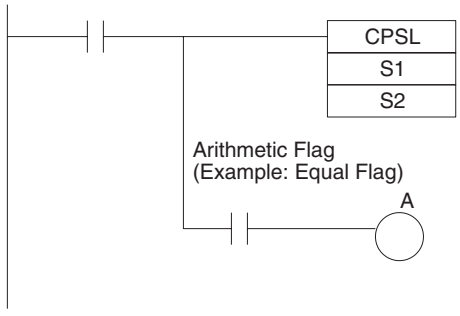
The following table shows the status of the Arithmetic Flags after execution of CPSL(115). (A status of "---" indicates that the Flag may be ON or OFF.)

CPSL(115)Result	Flag status					
	>	> =	=	< =	<	< >
S_1 +1, S_1 > S_2+1, S_2	ON	ON	OFF	OFF	OFF	ON
S_1+1, S_1 = S_2+1, S_2	OFF	ON	ON	ON	OFF	OFF
S_1+1, S_1 < S_2+1, S_2	OFF	OFF	OFF	ON	ON	ON

Using CPSL(115) Results in the Program

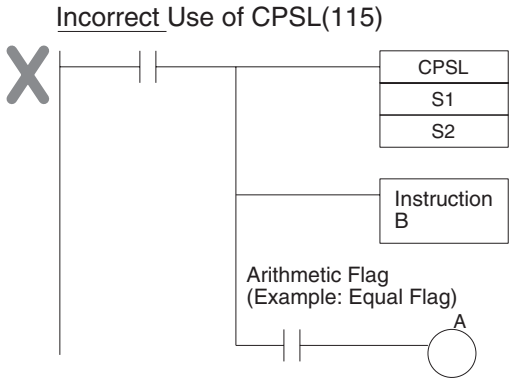
When CPSL(115) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls CPSL(115), as shown in the following diagram. Here, the Equals Flag and output A will be turned ON when S_1 +1, S_1 = S_2+1, S_2.

Correct Use of CPSL(115)



Using CPSL(115) Results in the Program

Do not program another instruction between CPSL(115) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag. In this case, the results of instruction B might change the results of CPSL(115).



Flags

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

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

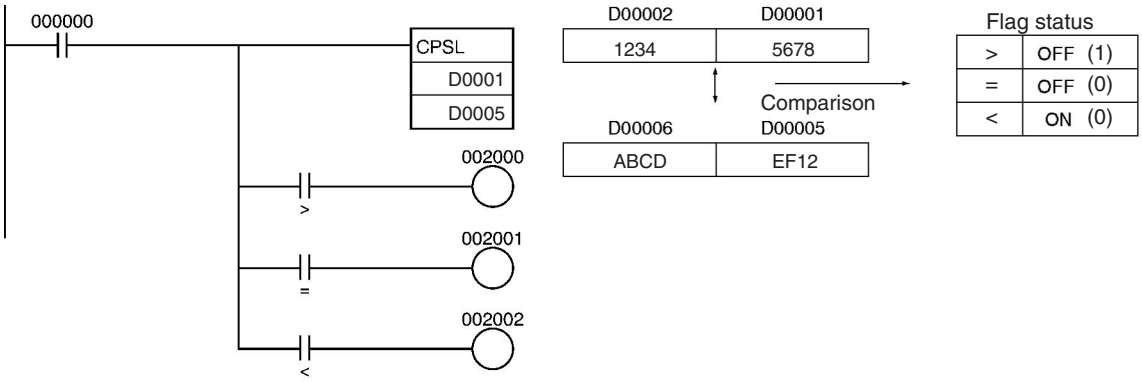
Precautions

Do not program another instruction between CPSL(115) and an input condition that accesses the result of CPSL(115) because the other instruction might change the status of the Arithmetic Flags.

Example

When CIO 000000 is ON in the following example, the eight-digit signed binary data in D00002 and D00001 is compared to the eight-digit signed binary data in D00006 and D00005 and the result is output to the Arithmetic Flags.

- If the content of D00002 and D00001 is greater than that of D00006 and D00005, the Greater Than Flag will be turned ON, causing CIO 002000 to be turned ON.
- If the content of D00002 and D00001 is equal to that of D00006 and D00005, the Equals Flag will be turned ON, causing CIO 002001 to be turned ON.
- If the content of D00002 and D00001 is less than that of D00006 and D00005, the Less Than Flag will be turned ON, causing CIO 002002 to be turned ON.

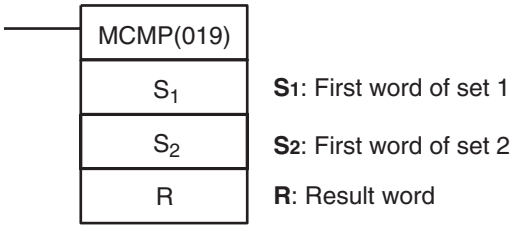


3-7-7 MULTIPLE COMPARE: MCMP(019)

Purpose

Compares 16 consecutive words with another 16 consecutive words and turns ON the corresponding bit in the result word where the contents of the words **are not** equal.

Ladder Symbol



Variations

Variations	Executed Each Cycle for ON Condition	MCMP(019)
	Executed Once for Upward Differentiation	@MCMP(019)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S_1: First word of set 1

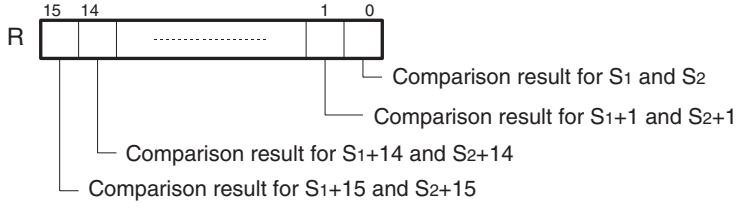
Specifies the beginning of the first 16-word range. S_1 and S_1+15 must be in the same data area.

S_2: First word of set 2

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

R: Result word

Each bit of R contains the result of a comparison between two words in the 16-word sets. Bit n of R (n = 00 to 15) contains the result of the comparison between words S_1+n and S_2+n.



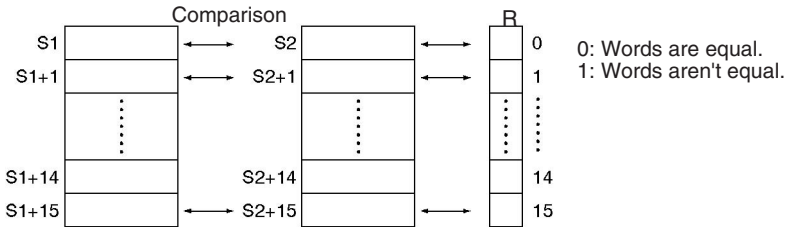
Operand Specifications

Area	S_1	S_2	R
CIO Area	CIO 0000 to CIO 6128		CIO 0000 to CIO 6143
Work Area	W000 to W496		W000 to W511
Holding Bit Area	H000 to H496		H000 to H511
Auxiliary Bit Area	A000 to A944		A448 to A959
Timer Area	T0000 to T4080		T0000 to T4095
Counter Area	C0000 to C4080		C0000 to C4095
DM Area	D00000 to D32752		D00000 to D32767
EM Area without bank	E00000 to E32752		E00000 to E32767
EM Area with bank	En_00000 to 32752 (n = 0 to C)		En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

MCMP(019) compares the contents of the 16 words S_1 through S_1+15 to the contents of the 16 words S_2 through S_2+15, and turns ON the corresponding bit in word R when the contents **are not** equal.

The content of S_1 is compared to the content of S_2, the content of S_1+1 to the content of S_2+1, ..., and the content of S_1+15 to the content of S_2+15. Bit n of R is turned OFF if the content of S_1+n is equal to the content of S_2+n; bit n of R is turned ON if the contents are not equal. If the contents of all 16 pairs of words are the same, the Equals Flag will turn ON after the instruction has been executed.

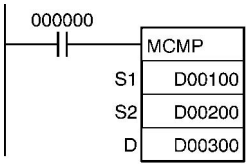


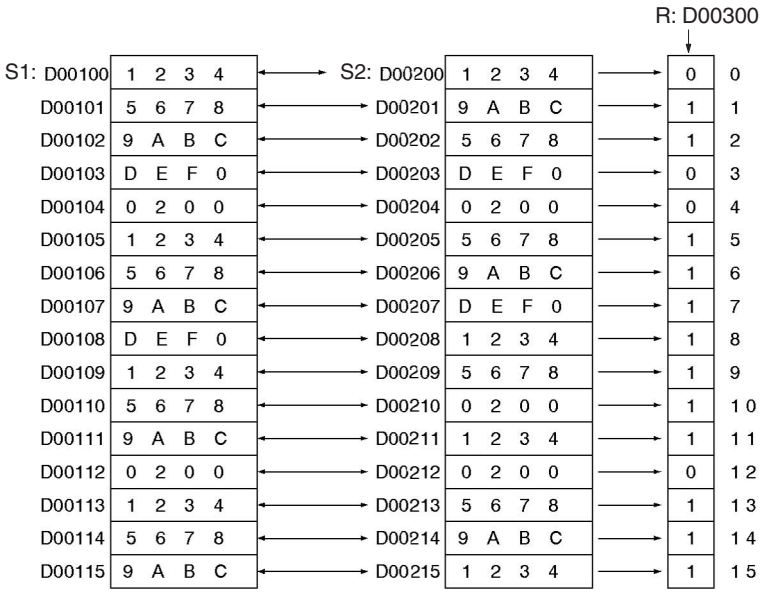
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result word is 0000. (The two 16-word sets contain the same data.) OFF in all other cases.

Example

When CIO 000000 is ON in the following example, MCMP(019) compares words D00100 through D00115 in order to words D00200 through D00215 and turns ON the corresponding bits in D00300 when the words **are not** equal.



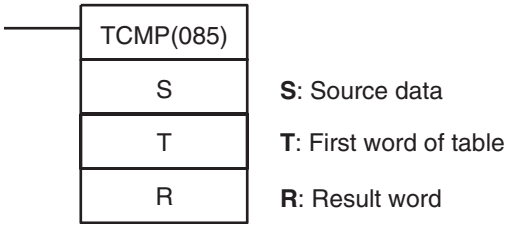


3-7-8 TABLE COMPARE: TCMP(085)

Purpose

Compares the source data to the contents of 16 consecutive words and turns ON the corresponding bit in the result word when the contents of the words **are** equal.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	TCMP(085)
	Executed Once for Upward Differentiation	@TCMP(085)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

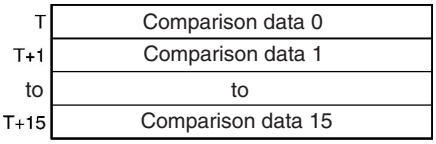
Operands

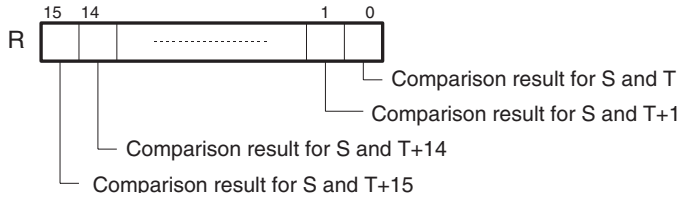
T: First word of table

Specifies the beginning of the 16-word table. T and T+15 must be in the same data area.

R: Result word

Each bit of R contains the result of a comparison between S and a word in the 16-word table. Bit n of R (n = 00 to 15) contains the result of the comparison between S and T+n.





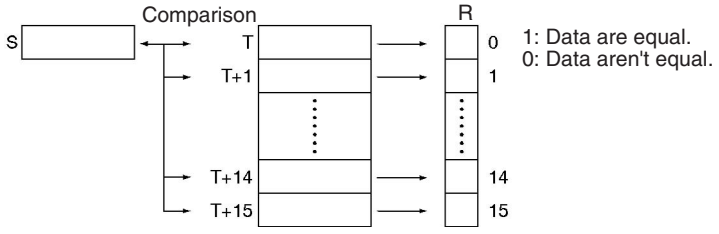
Operand Specifications

Area	S	T	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6128	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W496	W000 to W511
Holding Bit Area	H000 to H511	H000 to H496	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A944	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4080	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4080	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32752	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32752	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32752 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

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

S is compared to the content of T and bit 00 of R is turned ON if they are equal or OFF if they are not equal, S is compared to the content of T+1 and bit 01 of R is turned ON if they are equal or OFF if they are not equal, ..., and S is compared to the content of T+15 and bit 15 of R is turned ON if they are equal or OFF if they are not equal.

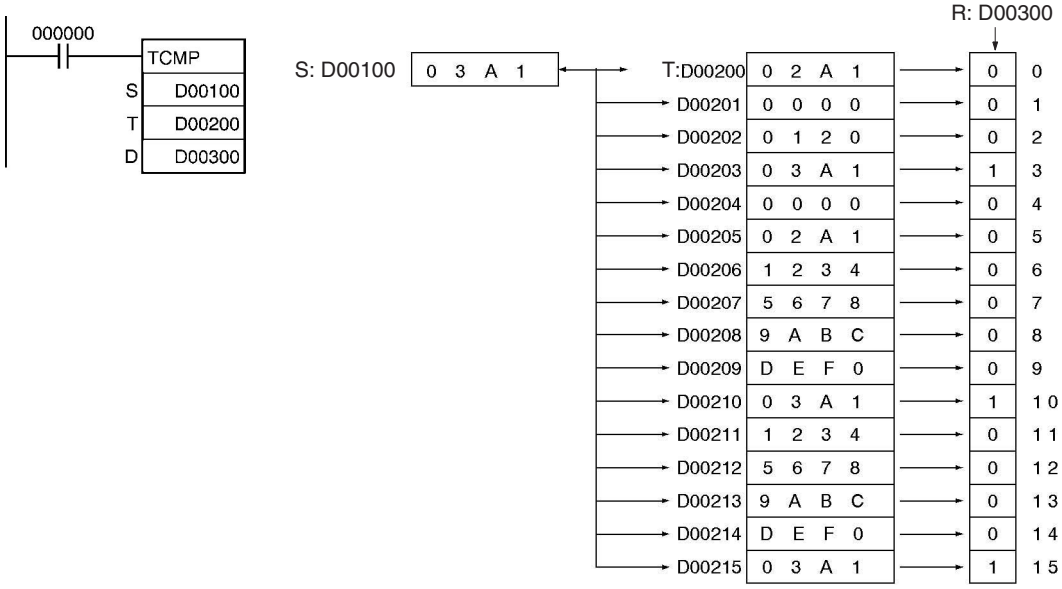


Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result word is 0000. (None of the 16 words in the table equals S.) OFF in all other cases.

Example

When CIO 000000 is ON in the following example, TCMP(085) compares the content of D00100 with the contents of words D00200 through D00215 and turns ON the corresponding bits in D00300 when the contents are equal or OFF when the contents are not equal.

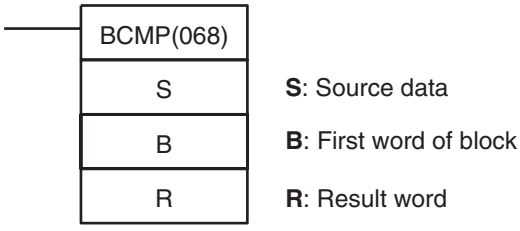


3-7-9 BLOCK COMPARE: BCMP(068)

Purpose

Compares the source data to 16 ranges (defined by 16 lower limits and 16 upper limits) and turns ON the corresponding bit in the result word when the source data is within a range.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	BCMP(068)
	Executed Once for Upward Differentiation	@BCMP(068)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

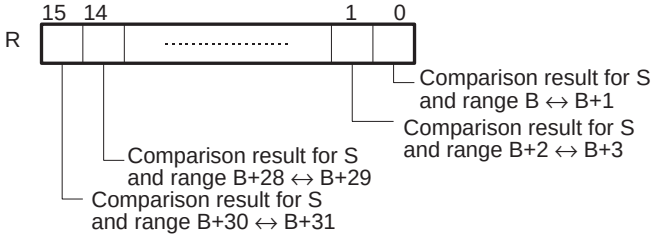
Operands

B: First word of block

Specifies the beginning of a 32-word block (16 lower/upper limit pairs). B and B+31 must be in the same data area.

R: Result word

Each bit of R contains the result of a comparison between S and one of the 16 ranges defined the 32-word block. Bit n of R (n = 00 to 15) contains the result of the comparison between S and the n^{th} pair of words.



Operand Specifications

Area	S	B	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6112	CIO 0000 to CIO 6143
Work Area	W000 to W511	W0000 to W480	W000 to W511
Holding Bit Area	H000 to H511	H000 to H480	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A928	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4064	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4064	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32736	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32736	E00000 to E32767

Area	S	B	R
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32736 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

BCMP(068) compares the source data (S) to the 16 ranges defined by pairs of lower and upper limit values in B through B+31. The first word in each pair (B+2n) provides the lower limit and the second word (B+2n+1) provides the upper limit of range n (n = 0 to 15). If S is within any of these ranges (inclusive of the upper and lower limits), the corresponding bit in R is turned ON. The rest of the bits in R will be turned OFF.

B	≤ S ≤	B+1	Bit 00 of R
B+2	≤ S ≤	B+3	Bit 01 of R
B+4	≤ S ≤	B+5	Bit 02 of R
B+6	≤ S ≤	B+7	Bit 03 of R
B+8	≤ S ≤	B+9	Bit 04 of R
B+10	≤ S ≤	B+11	Bit 05 of R
B+12	≤ S ≤	B+13	Bit 06 of R
B+14	≤ S ≤	B+15	Bit 07 of R
B+16	≤ S ≤	B+17	Bit 08 of R
B+18	≤ S ≤	B+19	Bit 09 of R
B+20	≤ S ≤	B+21	Bit 10 of R
B+22	≤ S ≤	B+23	Bit 11 of R
B+24	≤ S ≤	B+25	Bit 12 of R
B+26	≤ S ≤	B+27	Bit 13 of R
B+28	≤ S ≤	B+29	Bit 14 of R
B+30	≤ S ≤	B+31	Bit 15 of R

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

Flags

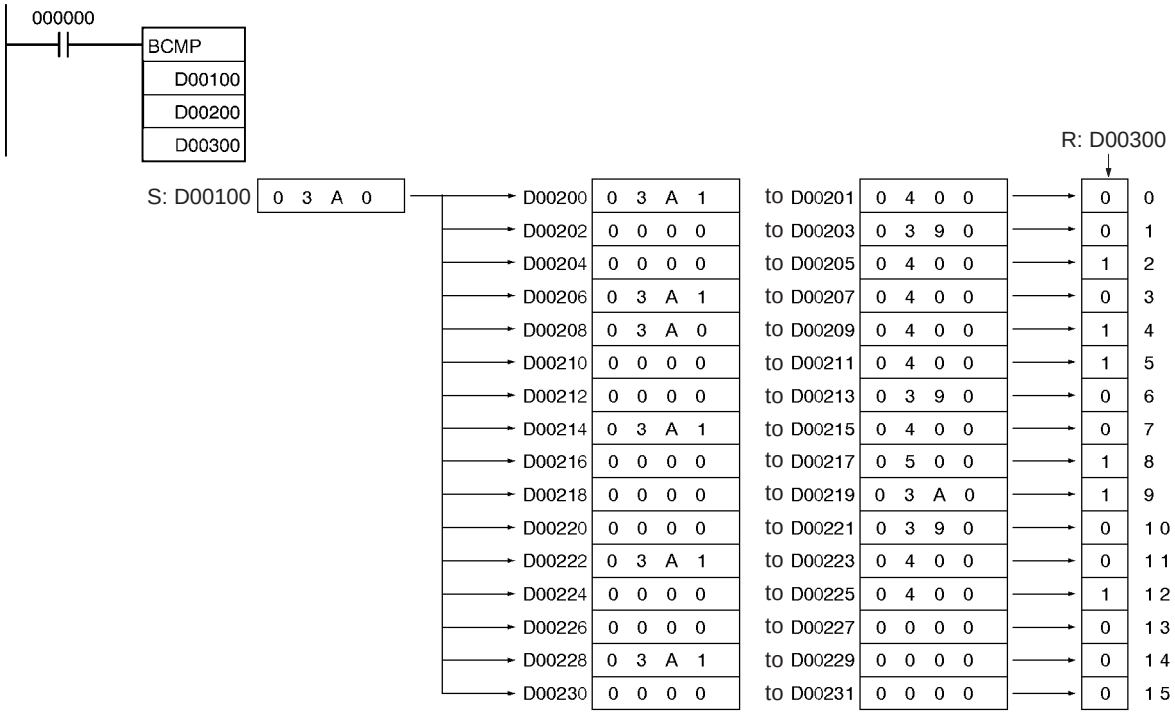
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result word is 0000. (S is not within any of the 16 ranges.) OFF in all other cases.

Precautions

An error will not occur if the lower limit is greater than the upper limit, but 0 (not within the range) will be output to the corresponding bit of R.

Example

When CIO 000000 is ON in the following example, BCMP(068) compares the content of D00100 with the 16 ranges defined in D00200 through D00231 and turns ON the corresponding bits in D00300 when S is within the range or OFF when S is not within the range.

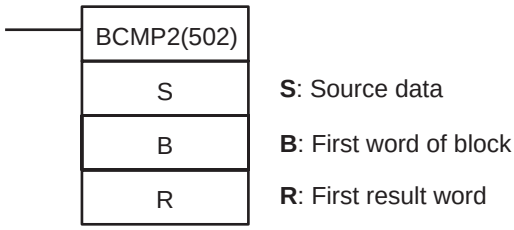


3-7-10 EXPANDED BLOCK COMPARE: BCMP2(502)

Purpose

Compares the source data to up to 256 ranges (defined by 256 lower limits and 256 upper limits) and turns ON the corresponding bit in the result word when the source data is within a range. BCMP2(502) is supported only by the CS1-H, CJ1-H, and CS1D CPU Unit Ver. 2.0 or later, and CJ1M CPU Unit (Pre-Ver. 2.0 or Unit Ver. 2.0 or later).

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	BCMP2(502)
	Executed Once for Upward Differentiation	@BCMP2(502)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

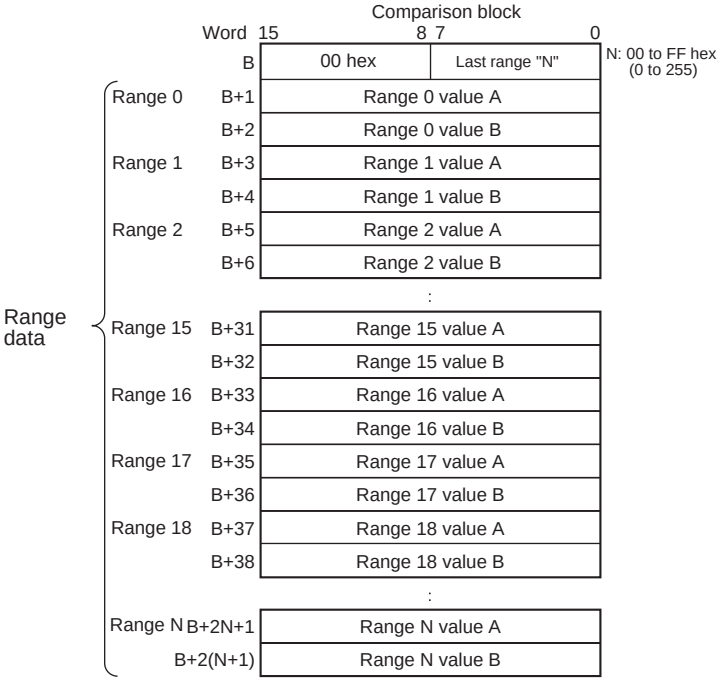
Applicable Program Areas

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

Operands

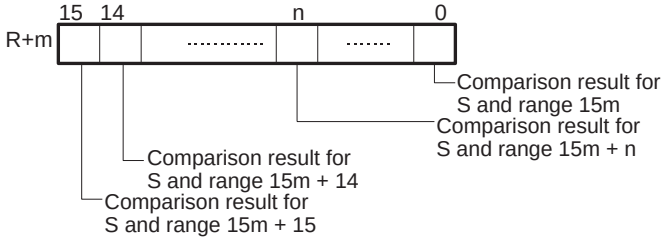
B: First word of block

Specifies the beginning of a comparison block containing up to 513 words including up to 256 lower/upper limit pairs). All words must be in the same data area.



R: First result word

Each bit of each R word contains the result of a comparison between S and one of the ranges defined the comparison block. The maximum number of result words is 16, i.e., m equals 0 to 15.



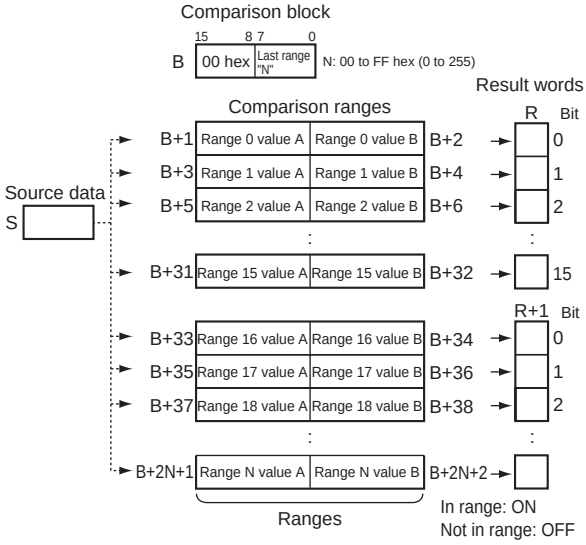
Operand Specifications

Area	S	B	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	---		
EM Area with bank	---		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767		
Indirect DM/EM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

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.

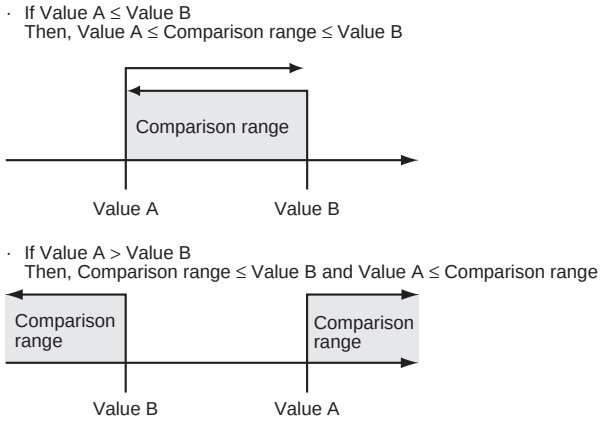


Number of Ranges

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.



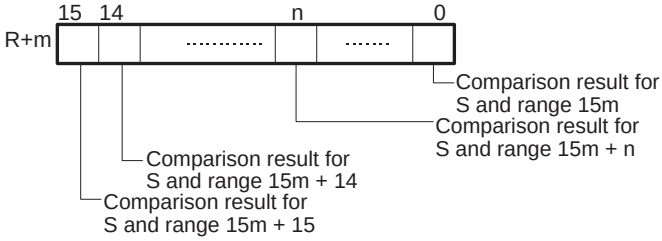
Example

When B+1 ≤ B+2
If B+1 ≤ S ≤ B+2, then bit 0 of R will turn ON,
If B+3 ≤ S ≤ B+4, then bit 1 of R will turn ON,
If S < B+5 and B+6 < S, then bit 2 of R will turn OFF, and
If S < B+7 and B+8 < S, then bit 3 of R will turn OFF.

When B+1 > B+2
If S ≤ B+2 and B+1 ≤ S, then bit 0 of R will turn ON,
If S ≤ B+4 and B+3 ≤ S, then bit 1 of R will turn ON,
If B+6 < S < B+5, then bit 2 of R will turn OFF, and
If B+8 < S < B+7, then bit 3 of R will turn OFF.

Results Storage Location

The results are output to corresponding bits in word R. If there are more than 16 comparison ranges, consecutive words following R will be used. The maximum number of result words is 16, i.e., m equals 0 to 15.



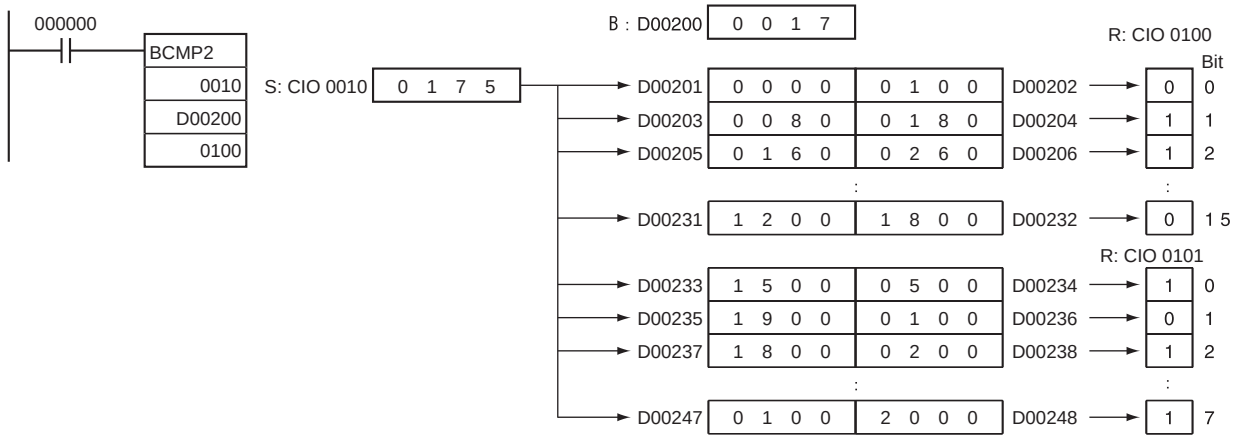
Flags

Name	Label	Operation
Error Flag	ER	OFF

Example

When CIO 000000 is ON in the following example, BCMP2(502) compares the content of CIO 0010 with the 24 ranges defined in D00200 through D00247 (N = 17 hex = 23 decimal, i.e., 24 ranges) and turns ON the corresponding bits in CIO 0100 and CIO 0101 when S is within the range and OFF when S is not within the range. For example, if the source data in CIO 0010 is in the range defined by D00201 and D00202, then bit 00 of CIO 0100 is turned ON and if it in not in the range, then bit 00 of CIO 0100 is turned OFF. Likewise, the source data in CIO 0010 is compared to the ranges defined by D00203 and D00204, D00247 and D00248, and the other words in the com-

parison block, and bit 1 in CIO 0100, bit 7 in CIO 1010, and the other bits in the result words are manipulated according to the results of comparison.



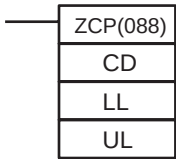
3-7-11 AREA RANGE COMPARE: ZCP(088)

Purpose

Compares a 16-bit unsigned binary value (CD) with the range defined by lower limit LL and upper limit UL. The results are output to the Arithmetic Flags.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



CD: Comparison Data
LL: Lower limit of range
UL: Upper limit of range

Variations

Variations	**Executed Each Cycle for ON Condition**	ZCP(088)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications

Area	CD	LL	UL
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		

Area	CD	LL	UL
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

ZCP(088) compares the 16-bit signed binary data in CD with the range defined by LL and UL and outputs the result to the Greater Than, Equals, and Less Than Flags in the Auxiliary Area. (The Less Than or Equal, Greater Than or Equal, and Not Equal Flags are left unchanged.)

Arithmetic Flag Status

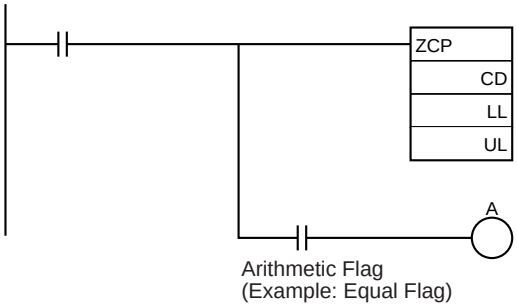
The following table shows the status of the Arithmetic Flags after execution of ZCP(088).

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

Using ZCP(088) Results in the Program

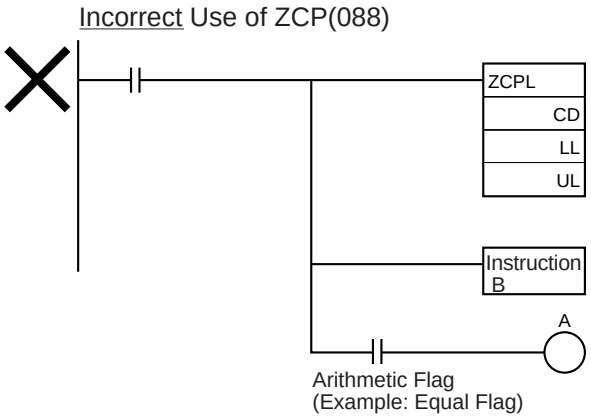
When ZCP(088) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls ZCP(088), as shown in the following diagram. In this case, the Equals Flag and output A will be turned ON when LL ≤ CD ≤ UL.

Correct Use of ZCP(088)



Do not program another instruction between ZCP(088) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag. In this case, the results of instruction B might change the results of ZCP(088).

Incorrect Use of ZCP(088)



Flags

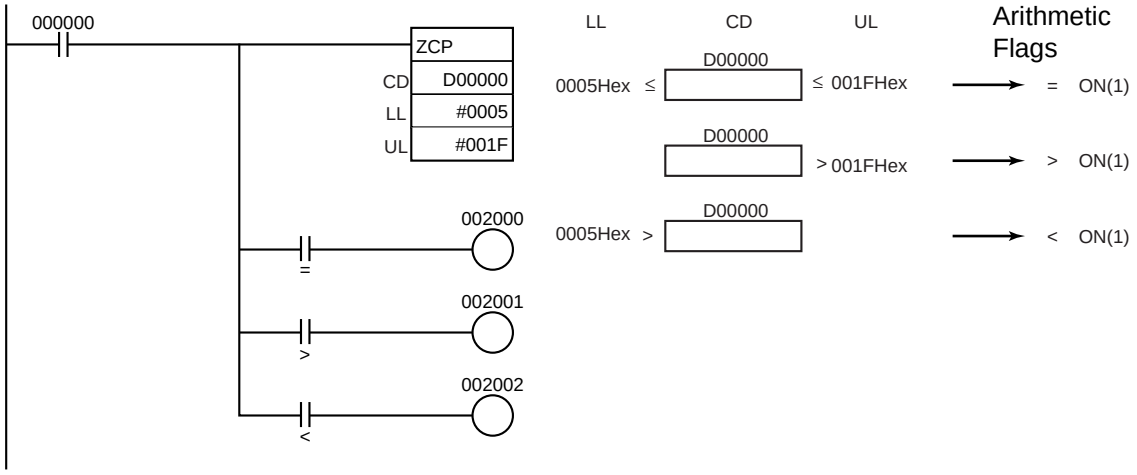
Name	Label	Operation
Error Flag	ER	ON if LL > UL.
Greater Than Flag	>	ON if CD > UL. OFF in all other cases.
Greater Than or Equal Flag	> =	Left unchanged.
Equal Flag	=	ON if LL ≤ CD ≤ UL. OFF in all other cases.
Not Equal Flag	<>	Left unchanged.
Less Than Flag	<	ON if CD < LL. OFF in all other cases.
Less Than or Equal Flag	< =	Left unchanged.
Negative Flag	N	Left unchanged.

Precautions

Do not program another instruction between ZCP(088) and an input condition that accesses the result of ZCP(088) because the other instruction might change the status of the Arithmetic Flags.

Example

When CIO 000000 is ON in the following example, the 16-bit unsigned binary data in D00000 is compared to the range 0005 to 001F hex (5 to 31 decimal) and the result is output to the Arithmetic Flags.
CIO 000200 is turned ON if 0005 hex ≤ content of D00000 ≤ 001F hex.
CIO 000201 is turned ON if the content of D00000 > 001F hex.
CIO 000202 is turned ON if the content of D00000 < 0005 hex.



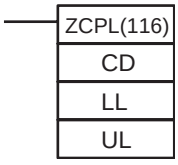
3-7-12 DOUBLE AREA RANGE COMPARE: ZCPL(116)

Purpose

Compares a 32-bit unsigned binary value (CD+1, CD) with the range defined by lower limit (LL+1, LL) and upper limit (UL+1, UL). The results are output to the Arithmetic Flags.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



CD: First word of Comparison Data
LL: First word of Lower Limit
UL: First word of Upper Limit

Variations

Variations		
	Executed Each Cycle for ON Condition	ZCP(088)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications

Area	CD	LL	UL
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		

Area	CD	LL	UL
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 0000 to #FFFF FFFF (binary)		
Data Registers	---		
Index Registers	IR0 to IR15		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

ZCPL(116) compares the 32-bit signed binary data in CD+1, CD with the range defined by LL+1, LL and UL+1, UL and outputs the result to the Greater Than, Equals, and Less Than Flags in the Auxiliary Area. (The Less Than or Equal, Greater Than or Equal, and Not Equal Flags are left unchanged.)

Arithmetic Flag Status

The following table shows the status of the Arithmetic Flags after execution of ZCPL(116).

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

Using ZCPL(116) Results in the Program

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

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

The operation of ZCPL(116) is almost identical to that of ZCP(088) except that ZCPL(116) compares 32-bit values instead of 16-bit values. Refer to *3-7-11 AREA RANGE COMPARE: ZCP(088)* for diagrams showing how to use results in the program and an example program section.

Flags

Name	Label	Operation
Error Flag	ER	ON if LL+1, LL > UL+1, UL.
Greater Than Flag	>	ON if CD > UL+1, UL. OFF in all other cases.

Name	Label	Operation
Greater Than or Equal Flag	> =	Left unchanged.
Equal Flag	=	ON if LL+1, LL ≤ CD+1, CD ≤ UL+1, UL. OFF in all other cases.
Not Equal Flag	<>	Left unchanged.
Less Than Flag	<	ON if CD+1, CD < LL+1, LL. OFF in all other cases.
Less Than or Equal Flag	< =	Left unchanged.
Negative Flag	N	Left unchanged.

Precautions

Do not program another instruction between ZCPL(116) and an input condition that accesses the result of ZCPL(116) because the other instruction might change the status of the Arithmetic Flags.

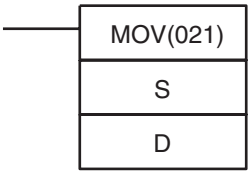
3-8 Data Movement Instructions

3-8-1 MOVE: MOV(021)

Purpose

Transfers a word of data to the specified word.

Ladder Symbol



S: Source

D: Destination

Variations

Variations	**Executed Each Cycle for ON Condition**	MOV(021)
	Executed Once for Upward Differentiation	@MOV(021)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification (See note.)		!MOV(021)
Combined Variations	**Executed Once and Destination Refreshed Immediately for Upward Differentiation (See note.)**	!@MOV(021)

Note Immediate refreshing is not supported by CS1D CPU Units.

Applicable Program Areas

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

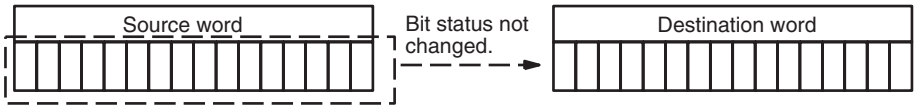
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	

Area	S	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15	

Description

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



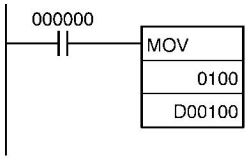
MOV(021) has an immediate refreshing variation (!MOV(021)). An external input bits can be specified for S and external output bits can be specified for D. Input bits used for S will refreshed just before, and output bits used for D will be refreshed just after execution unless the bits are allocated 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.

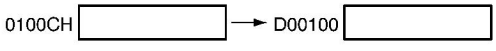
Flags

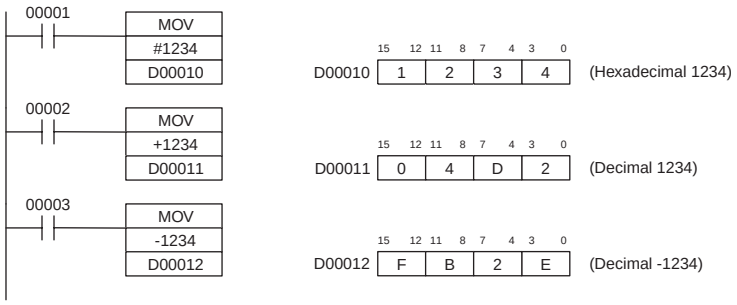
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the data being transferred is 0000. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the data being transferred is 1. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, the content of CIO 0100 is copied to D00100.





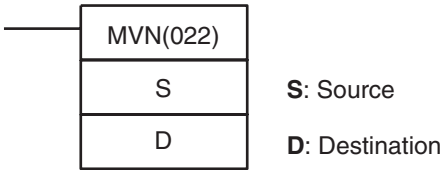


3-8-2 MOVE NOT: MVN(022)

Purpose

Transfers the complement of a word of data to the specified word.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	MVN(022)
	Executed Once for Upward Differentiation	@MVN(022)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	

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

Description

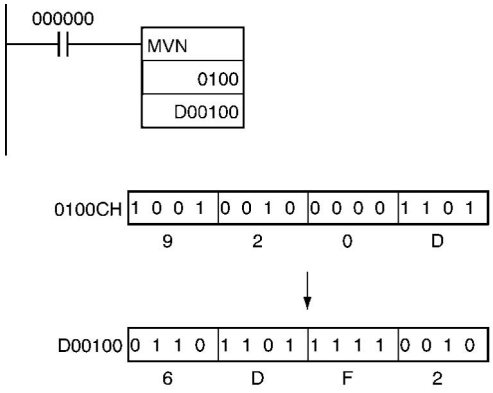
MVN(022) inverts the bits in S and transfers the result to D. The content of S is left unchanged.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the content of D is 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of D is 1 after execution. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, the status of the bits in CIO 0100 is inverted and the result is copied to D00100.

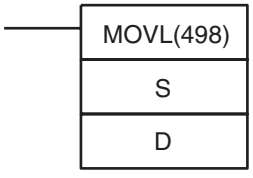


3-8-3 DOUBLE MOVE: MOVL(498)

Purpose

Transfers two words of data to the specified words.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVL(498)
	Executed Once for Upward Differentiation	@MOVL(498)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

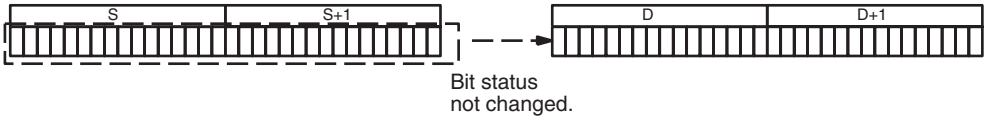
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	IR0 to IR15	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, 1–(– –) IR5	

Description

MOVL(498) transfers S+1 and S to D+1 and D. If S+1 and S are constants, the value can be used for a data setting.

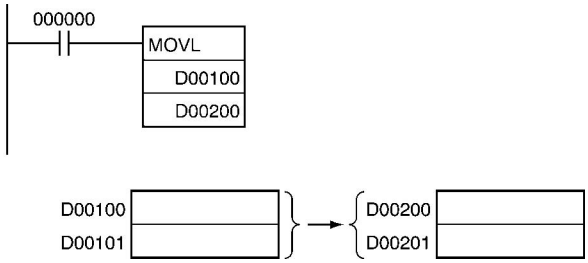


Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the contents of D+1 and D are 0000 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of D+1 is 1 after execution. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, the content of D00101 and D00100 are copied to D00201 and D00200.

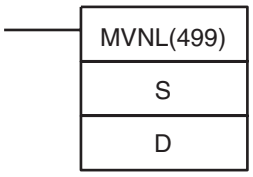


3-8-4 DOUBLE MOVE NOT: MVNL(499)

Purpose

Transfers the complement of two words of data to the specified words.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	MVNL(499)
	Executed Once for Upward Differentiation	@MVNL(499)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

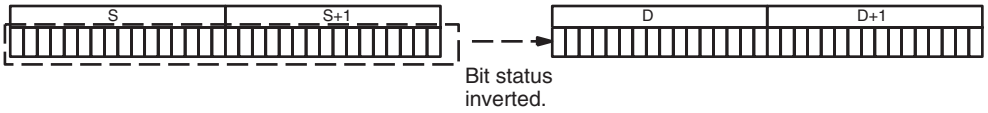
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---

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

Description

MVNL(499) inverts the bits in S+1 and S and transfers the result to D+1 and D. The contents of S+1 and S are left unchanged.

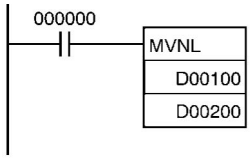


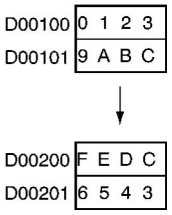
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the contents of D+1 and D are 0000 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of D+1 is 1 after execution. OFF in all other cases.

Examples

When CIO 000000 is ON in the following example, the status of the bits in D00101 and D00100 are inverted and the result is copied to D00201 and D00200. (The original contents of D00101 and D00100 are left unchanged.)



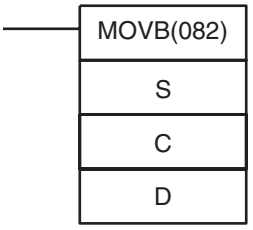


3-8-5 MOVE BIT: MOVB(082)

Purpose

Transfers the specified bit.

Ladder Symbol



S: Source word or data

C: Control word

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVB(082)
	Executed Once for Upward Differentiation	@MOVB(082)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

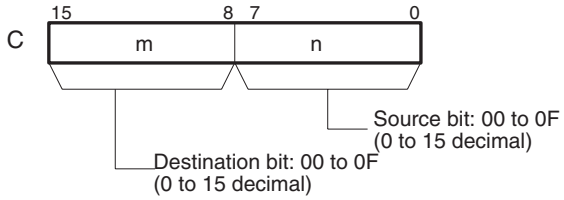
Applicable Program Areas

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

Operands

C: Control Word

The rightmost two digits of C indicate which bit of S is the source bit and the leftmost two digits of C indicate which bit of D is the destination bit.

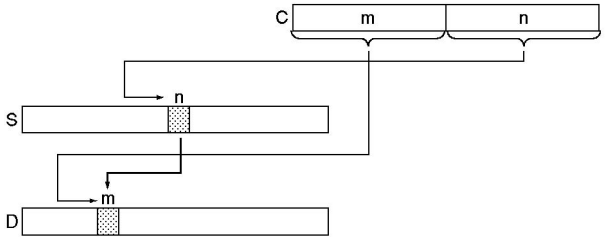


Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	Specified values only	---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

MOVB(082) copies the specified bit (n) from S to the specified bit (m) in D. The other bits in the destination word are left unchanged.



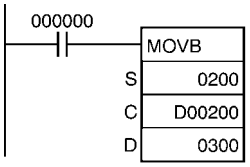
Note The same word can be specified for both S and D to copy a bit within a word.

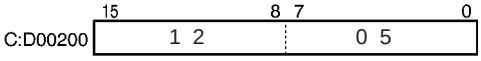
Flags

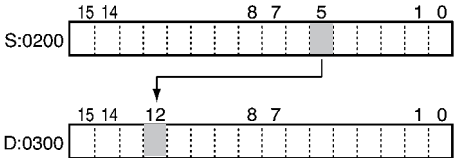
Name	Label	Operation
Error Flag	ER	ON if the rightmost and leftmost two digits of C are not within the specified range of 00 to 0F. OFF in all other cases.

Examples

When CIO 000000 is ON in the following example, the 5th bit of the source word (CIO 0200) is copied to the 12th bit of the destination word (CIO 0300) in accordance with the control word's value of 0C05.





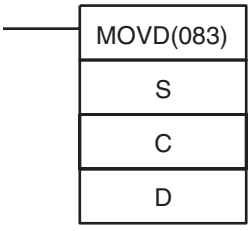


3-8-6 MOVE DIGIT: MOVD(083)

Purpose

Transfers the specified digit or digits. (Each digit is made up of 4 bits.)

Ladder Symbol



S: Source word or data

C: Control word

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVD(083)
	Executed Once for Upward Differentiation	@MOVD(083)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

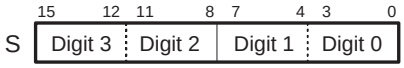
Applicable Program Areas

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

Operands

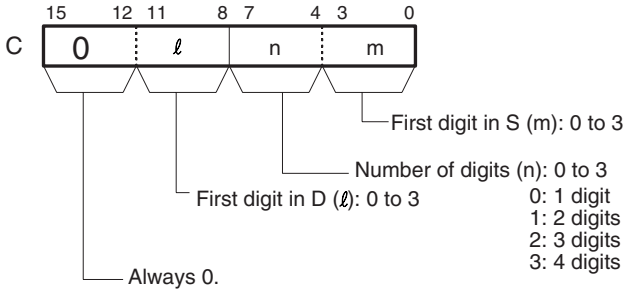
S: Source Word

The source digits are read from right to left, wrapping back to the rightmost digit (digit 0) if necessary.



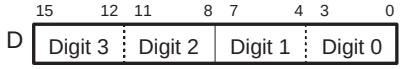
C: Control Word

The first three digits of C indicate the first source digit (m), the number of digits to transfer (n), and the first destination digit (ℓ), as shown in the following diagram.



D: Destination Word

The destination digits are written from right to left, wrapping back to the rightmost digit (digit 0) if necessary.



Operand Specifications

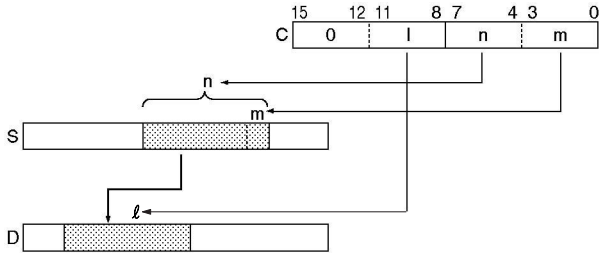
Area	S	C	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	Specified values only	---
Data Registers	DR0 to DR15		

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

Description

MOVD(083) copies the content of n digits from S (beginning at digit m) to D (beginning at digit ℓ). Only the specified digits are changed; the rest are left unchanged.

If the number of digits being read or written exceeds the leftmost digit of S or D, MOVD(083) will wrap to the rightmost digit of the same word.



Note The same word can be specified for both S and D to copy a bit within a word.

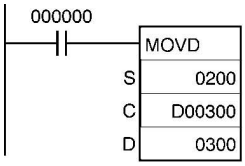
Flags

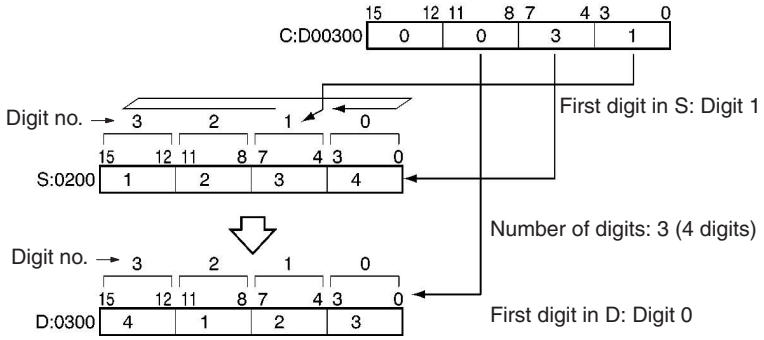
Name	Label	Operation
Error Flag	ER	ON if one of the first three digits of C is not within the specified range of 0 to 3. OFF in all other cases.

Examples

Four-digit Transfer

When CIO 000000 is ON in the following example, four digits of data are copied from CIO 0200 to CIO 0300. The transfer begins with the digit 1 of CIO 0200 and digit 0 or CIO 0300, in accordance with the control word's value of 0031.

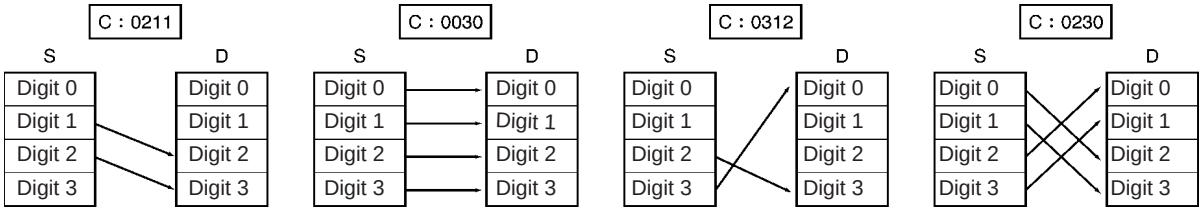




Note After reading the leftmost digit of S (digit 3), MOVD(083) wraps to the rightmost digit (digit 0).

Examples of C

The following diagram shows examples of data transfers for various values of C.

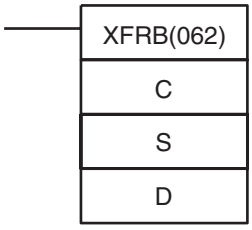


3-8-7 MULTIPLE BIT TRANSFER: XFRB(062)

Purpose

Transfers the specified number of consecutive bits.

Ladder Symbol



C: Control word

S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	XFRB(062)
	Executed Once for Upward Differentiation	@XFRB(062)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

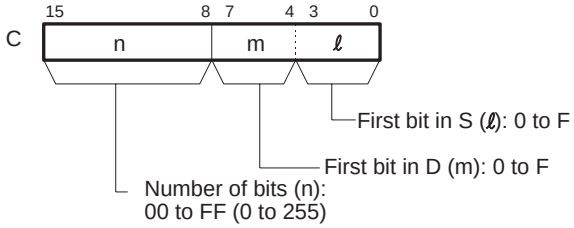
Applicable Program Areas

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

Operands

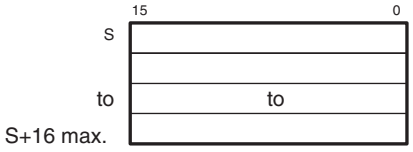
C: Control Word

The first three digits of C indicate the first destination bit (m), the number of bits to transfer (n), and the first source digit (ℓ), as shown in the following diagram.



S: First Source Word

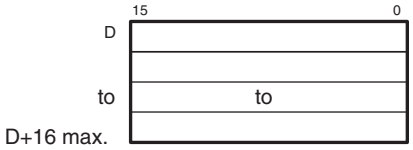
Specifies the first source word. Bits are read from right to left, continuing with consecutive words (up to S+16) when necessary.



Note The source words must be in the same data area.

D: First Destination Word

Specifies the first destination word. Bits are written from right to left, continuing with consecutive words (up to D+16) when necessary.



Note The destination words must be in the same data area.

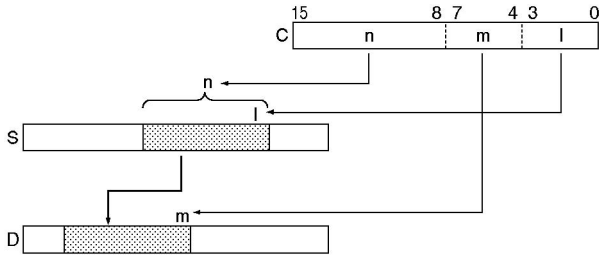
Operand Specifications

Area	C	S	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	Specified values only	---	---
Data Registers	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to 5+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

XFRB(062) transfers up to 255 consecutive bits from the source words (beginning with bit ℓ of S) to the destination words (beginning with bit m of D). Bits in the destination words that are not overwritten by the source bits are left unchanged.

The beginning bits and number of bits are specified in C, as shown in the following diagram.



It is possible for the source words and destination words to overlap. By transferring data overlapping several words, the data can be packed more efficiently in the data area. (This is particularly useful when handling position data for position control.)

Since the source words and destination words can overlap, XFRB(062) can be combined with ANDW(034) to shift m bits by n spaces.

Flags

Name	Label	Operation
Error Flag	ER	OFF

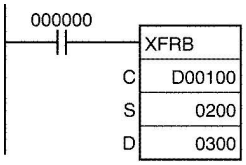
Precautions

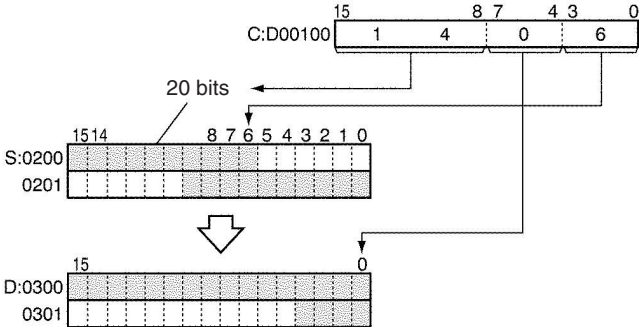
Up to 255 bits of data can be transferred per execution of XFRB(062).

Be sure that the source words and destination words do not exceed the end of the data area.

Examples

When CIO 000000 is ON in the following example, the 20 bits beginning with CIO 020006 are copied to the 20 bits beginning with CIO 030000.



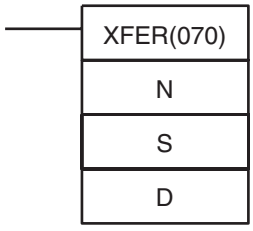


3-8-8 BLOCK TRANSFER: XFER(070)

Purpose

Transfers the specified number of consecutive words.

Ladder Symbol



N: Number of words

S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	XFER(070)
	Executed Once for Upward Differentiation	@XFER(070)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

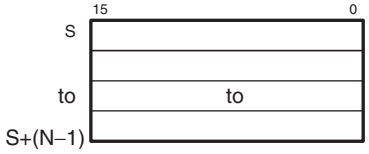
Operands

N: Number of Words

Specifies the number of words to be transferred. The possible range for N is 0000 to FFFF (0 to 65,535 decimal).

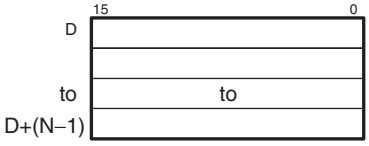
S: First Source Word

Specifies the first source word.



D: First Destination Word

Specifies the first destination word.



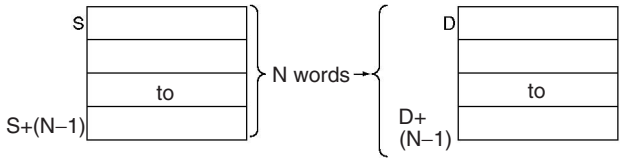
Operand Specifications

Area	N	S	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary) or &0 to &65535	---	---
Data Registers	DR0 to DR15	---	

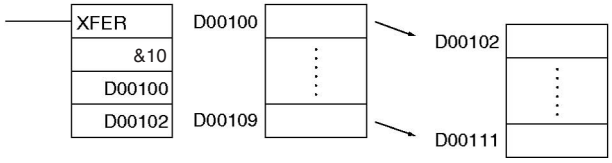
Area	N	S	D
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

XFER(070) copies N words beginning with S (S to S+(N–1)) to the N words beginning with D (D to D+(N–1)).



It is possible for the source words and destination words to overlap, so XFER(070) can perform word-shift operations.



Flags

Name	Label	Operation
Error Flag	ER	OFF

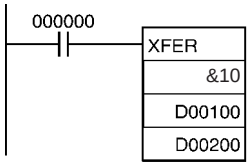
Precautions

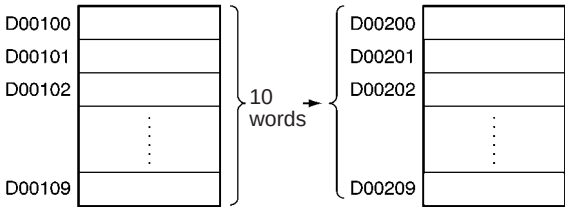
Be sure that the source words (S to S+N–1) and destination words (D to D+N–1) do not exceed the end of the data area.

Some time will be required to complete XFER(070) when a large number of words is being transferred. In this case, the XFER(070) transfer might not be completed if a power interruption occurs during execution of the instruction.

Example

When CIO 000000 is ON in the following example, the 10 words D00100 through D00109 are copied to D00200 through D00209.



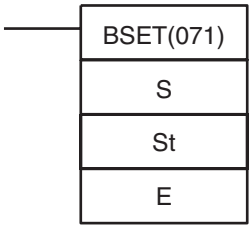


3-8-9 BLOCK SET: BSET(071)

Purpose

Copies the same word to a range of consecutive words.

Ladder Symbol



S: Source word

St: Starting word

E: End word

Variations

Variations		
	Executed Each Cycle for ON Condition	BSET(071)
	Executed Once for Upward Differentiation	@BSET(071)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: Source Word

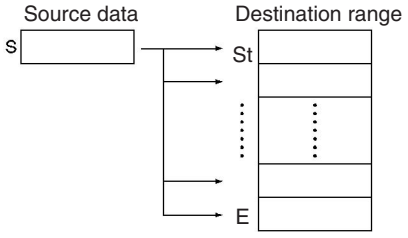
Specifies the source data or the word containing the source data.

St: Starting Word

Specifies the first word in the destination range.

E: End Word

Specifies the last word in the destination range.



Note St and E must be in the same data area.

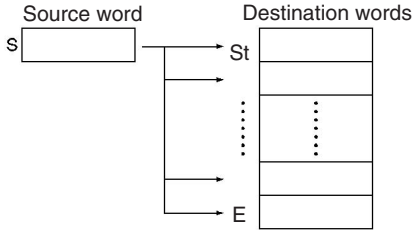
Operand Specifications

Area	S	St	E
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959	A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		

Area	S	St	E
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, 15–(– –) IR		

Description

BSET(071) copies the same source word (S) to all of the destination words in the range St to E.



Flags

Name	Label	Operation
Error Flag	ER	ON if St is greater than E. OFF in all other cases.

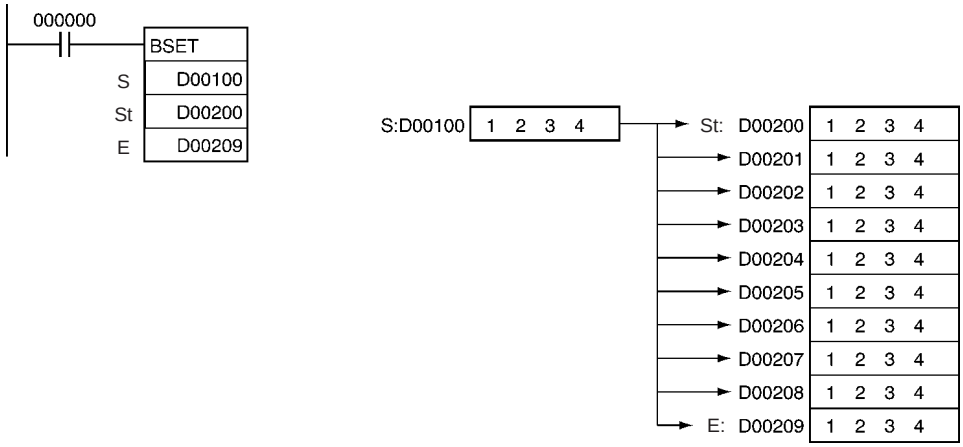
Precautions

Be sure that the starting word (St) and end word (E) are in the same data area and that St ≤ E.

Some time will be required to complete BSET(071) when the source data is being transferred to a large number of words. In this case, the BSET(071) transfer might not be completed if a power interruption occurs during execution of the instruction.

Example

When CIO 000000 is ON in the following example, the source data in D00100 is copied to D00200 through D00209.

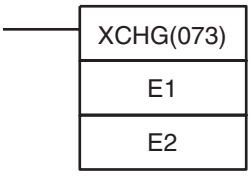


3-8-10 DATA EXCHANGE: XCHG(073)

Purpose

Exchanges the contents of the two specified words.

Ladder Symbol



E1: First exchange word

E2: Second exchange word

Variations

Variations		
	Executed Each Cycle for ON Condition	XCHG(073)
	Executed Once for Upward Differentiation	@XCHG(073)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

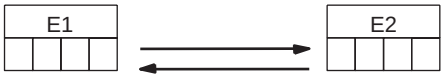
Operand Specifications

Area	E1	E2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	

Area	E1	E2
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15	

Description

XCHG(073) exchanges the contents of E1 and E2.



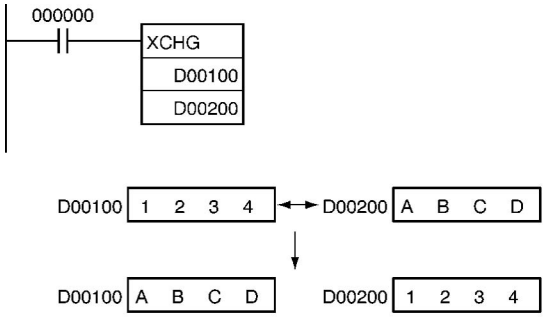
Flags

Name	Label	Operation
Error Flag	ER	Unchanged (See note.)
Equals Flag	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

Note In 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.

Example

When CIO 000000 is ON in the following example, the content of D00100 is exchanged with the content of D00200.

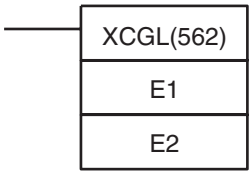


3-8-11 DOUBLE DATA EXCHANGE: XCGL(562)

Purpose

Exchanges the contents of a pair of consecutive words with another pair of consecutive words.

Ladder Symbol



E1: First exchange word

E2: Second exchange word

Variations

Variations		
	Executed Each Cycle for ON Condition	XCGL(562)
	Executed Once for Upward Differentiation	@XCGL(562)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

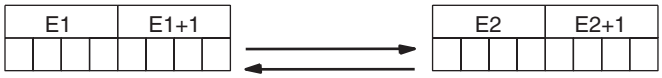
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

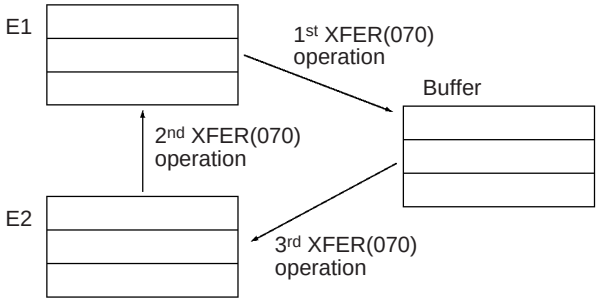
Area	E1	E2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A448 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	---
Data Registers	---	
Index Registers	IR0 to IR15	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15	

Description

XCHG(073) exchanges the contents of E1+1 and E1 with the contents of E2+1 and E2.



To exchange 3 or more words, use XFER(070) to transfer the words to a third set of words (a buffer) as shown in the following diagram.

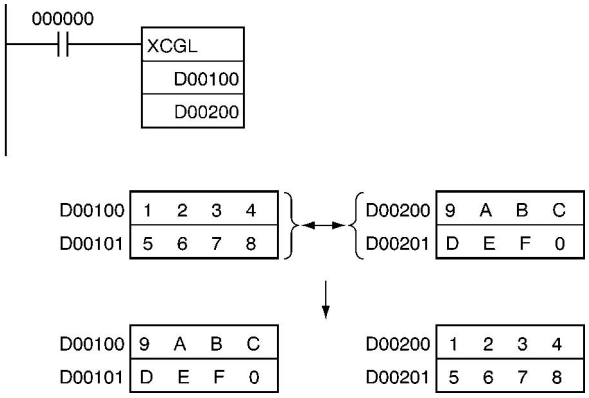


Flags

Name	Label	Operation
Error Flag	ER	Unchanged (See note.)
Equals Flag	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

Note In 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.

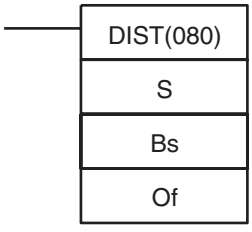
Example When CIO 000000 is ON in the following example, the contents of D00100 and D00101 are exchanged with the contents of D00200 and D00201.



3-8-12 SINGLE WORD DISTRIBUTE: DIST(080)

Purpose Transfers the source word to a destination word calculated by adding an offset value to the base address.

Ladder Symbol



S: Source word

Bs: Destination base address

Of: Offset

Variations

Variations		
	Executed Each Cycle for ON Condition	DIST(080)
	Executed Once for Upward Differentiation	@DIST(080)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

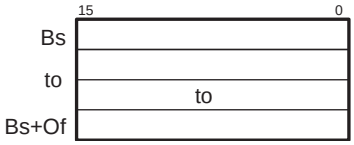
Operands

Bs: Destination Base Address

Specifies the destination base address. The offset is added to this address to calculate the destination word.

Of: Offset

This value is added to the base address to calculate the destination word. The offset can be any value from 0000 to FFFF (0 to 65,535 decimal), but Bs and Bs+Of must be in the same data area.

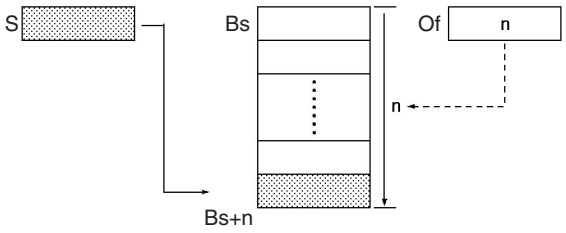


Operand Specifications

Area	S	Bs	Of
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959	A448 to A959	A000 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	#0000 to #FFFF (binary) or &0 to &65535
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

DIST(080) copies S to the destination word calculated by adding Of to Bs. The same DIST(080) instruction can be used to distribute the source word to various words in the data area by changing the value of Of.



Flags

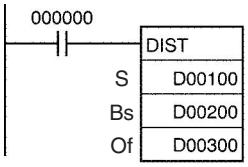
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the source data is 1. OFF in all other cases.

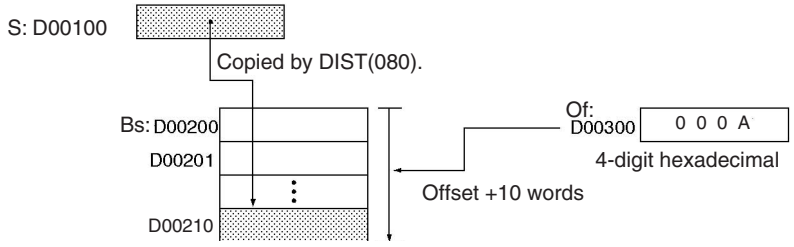
Precautions

Be sure that the offset does not exceed the end of the data area, i.e., Bs and Bs+Of are in the same data area.

Example

When CIO 000000 is ON in the following example, the contents of D00100 will be copied to D00210 (D00200 + 10) if the contents of D00300 is 10 (0A hexadecimal). The contents of D00100 can be copied to other words by changing the offset in D00300.



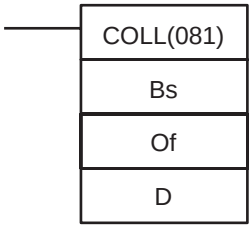


3-8-13 DATA COLLECT: COLL(081)

Purpose

Transfers the source word (calculated by adding an offset value to the base address) to the destination word.

Ladder Symbol



Bs: Source base address

Of: Offset

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	COLL(081)
	Executed Once for Upward Differentiation	@COLL(081)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

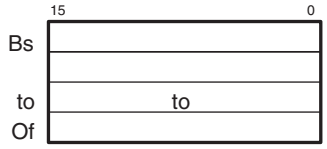
Operands

Bs: Source Base Address

Specifies the source base address. The offset is added to this address to calculate the source word.

Of: Offset

This value is added to the base address to calculate the source word. The offset can be any value from 0000 to FFFF (0 to 65,535 decimal), but Bs and Bs+Of must be in the same data area.

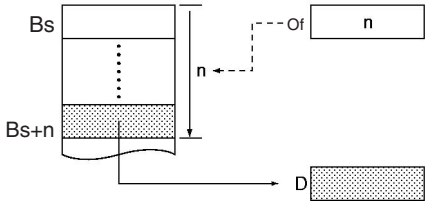


Operand Specifications

Area	Bs	Of	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0000 to #FFFF (binary) or &0 to &65535	---
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

COLL(081) copies the source word (calculated by adding Of to Bs) to the destination word. The same COLL(081) instruction can be used to collect data from various source words in the data area by changing the value of Of.



Flags

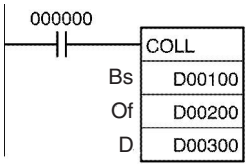
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the source data is 1. OFF in all other cases.

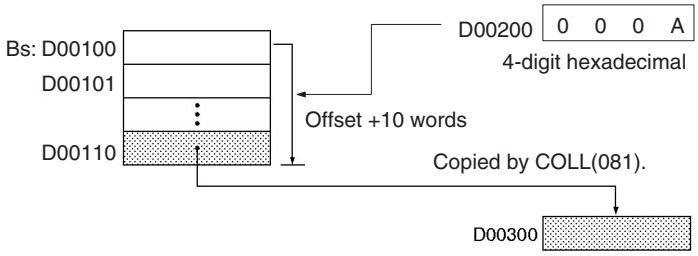
Precautions

Be sure that the offset does not exceed the end of the data area, i.e., Bs and Bs+Of are in the same data area.

Example

When CIO 000000 is ON in the following example, the contents of D00110 (D00100 + 10) will be copied to D00300 if the content of D00200 is 10 (0A hexadecimal). The contents of other words can be copied to D00300 by changing the offset in D00200.



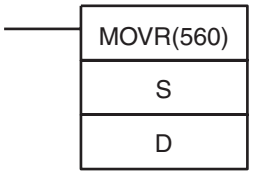


3-8-14 MOVE TO REGISTER: MOVR(560)

Purpose

Sets the PLC memory address of the specified word, bit, or timer/counter Completion Flag in the specified Index Register. (Use MOVRW(561) to set the PLC memory address of a timer/counter PV in an Index Register.)

Ladder Symbol



S: Source (desired word or bit)

D: Destination (Index Register)

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVR(560)
	Executed Once for Upward Differentiation	@MOVR(560)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

D: Destination

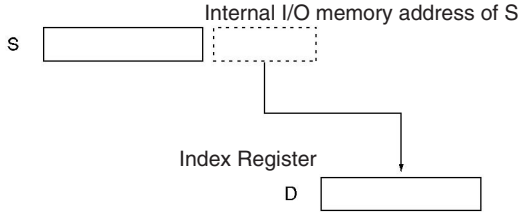
The destination must be an Index Register (IR0 to IR15).

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143 CIO 000000 to CIO 614315	---
Work Area	W000 to W511 W00000 to W51115	---
Holding Bit Area	H000 to H511 H00000 to H51115	---
Auxiliary Bit Area	A000 to A447 A448 to A959 A00000 to A44715 A44800 to A95915	---
Timer Area	T0000 to T4095 (Completion Flag)	---
Counter Area	C0000 to C4095 (Completion Flag)	---
Task Flag	TK0000 to TK0031	---
DM Area	D00000 to D32767	---
EM Area without bank	E00000 to E32767	---
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	---
Indirect DM/EM addresses in binary	---	
Indirect DM/EM addresses in BCD	---	
Constants	---	
Data Registers	---	
Index Registers	---	IR0 to IR15
Indirect addressing using Index Registers	---	

Description

MOVR(560) finds the PLC memory address (absolute address) of S and writes that address in D (an Index Register).



If a timer or counter is specified in S, MOVR(560) will write the PLC memory address of the timer/counter Completion Flag in D. Use MOVRW(561) to write the PLC memory address of the timer/counter PV in D.

Flags

Name	Label	Operation
Error Flag	ER	Unchanged (See note.)
Equals Flag	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

Note In 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.

Precautions

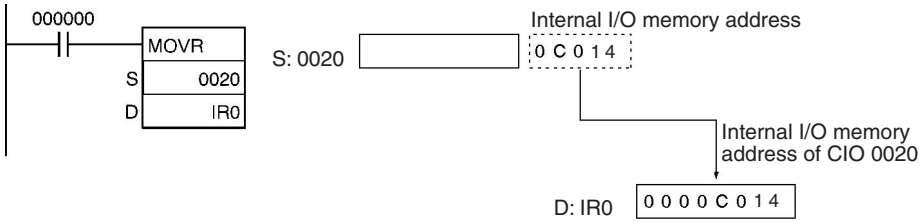
MOVR(560) cannot set the PLC memory addresses of timer/counter PVs. Use MOVRW(561) to set the PLC memory addresses of timer/counter PVs.

The contents of an index register in an interrupt task is not predictable until it is set. Be sure to set a register using MOVR(560) in an interrupt task before using the register.

Any changes to the contents of an IR or DR made in an interrupt task will not affect the contents of the register in a cyclic task.

Example

When CIO 000000 is ON in the following example, MOVR(560) writes the PLC memory address of CIO 0020 to IR0.

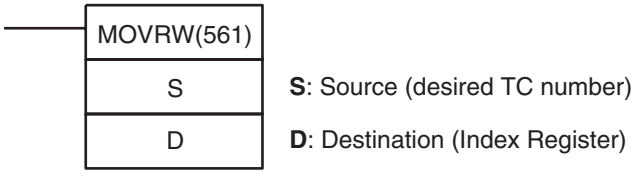


3-8-15 MOVE TIMER/COUNTER PV TO REGISTER: MOVRW(561)

Purpose

Sets the PLC memory address of the specified timer or counter's PV in the specified Index Register. (Use MOVR(560) to set the PLC memory address of a word, bit, or timer/counter Completion Flag in an Index Register.)

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	MOVR(561)
	Executed Once for Upward Differentiation	@MOVR(561)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

D: Destination

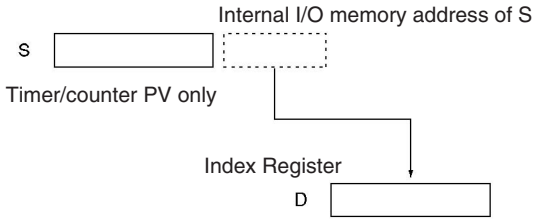
The destination must be an Index Register (IR0 to IR15).

Operand Specifications

Area	S	D
CIO Area	---	
Work Area	---	
Holding Bit Area	---	
Auxiliary Bit Area	---	
Timer Area	T0000 to T4095 (present value)	---
Counter Area	C0000 to C4095 (present value)	---
DM Area	---	
EM Area without bank	---	
EM Area with bank	---	
Indirect DM/EM addresses in binary	---	
Indirect DM/EM addresses in BCD	---	
Constants	---	
Data Registers	---	
Index Registers	---	IR0 to IR15
Indirect addressing using Index Registers	---	

Description

MOVRW(561) finds the PLC memory address for the PV of the timer or counter specified in S and writes that address in D (an Index Register).



MOVRW(561) will set the PLC memory address of the timer or counter's PV in D. Use MOVR(560) to set the PLC memory address of the timer or counter Completion Flag.

Flags

Name	Label	Operation
Error Flag	ER	Unchanged (See note.)
Equals Flag	=	Unchanged (See note.)
Negative Flag	N	Unchanged (See note.)

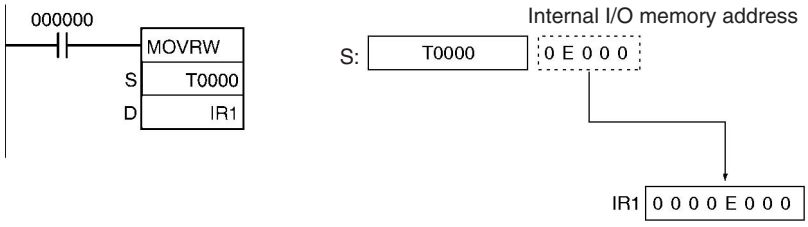
Note In CS1-H, CJ1-H, CJ1M, and CS1D (for Single-CPU System) CPU Units, these Flags are left unchanged.
In CS1 and CJ1 CPU Units, these Flags are turned OFF.

Precautions

MOVRW(561) cannot set the PLC memory addresses of data area words, bits, or timer/counter Completion Flags. Use MOVR(560) to set these PLC memory addresses.

Example

When CIO 000000 is ON in the following example, MOVRW(561) writes the PLC memory address for the PV of timer T0000 to IR1.



3-9 Data Shift Instructions

This section describes instructions used to shift data within or between words, but in differing amounts and directions.

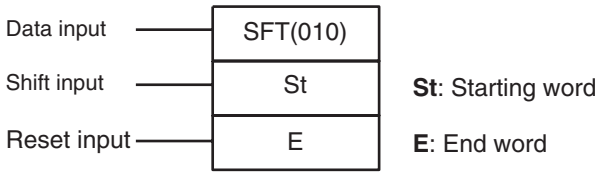
Instruction	Mnemonic	Function code	Page
SHIFT REGISTER	SFT	010	361
REVERSIBLE SHIFT REGISTER	SFTR	084	362
ASYNCHRONOUS SHIFT REGISTER	ASFT	017	365
WORD SHIFT	WSFT	016	368
ARITHMETIC SHIFT LEFT	ASL	025	370
DOUBLE SHIFT LEFT	ASLL	570	371
ARITHMETIC SHIFT RIGHT	ASR	026	373
DOUBLE SHIFT RIGHT	ASRL	571	374
ROTATE LEFT	ROL	027	376
DOUBLE ROTATE LEFT	ROLL	572	378
ROTATE LEFT WITHOUT CARRY	RLNC	574	383
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	576	385
ROTATE RIGHT	ROR	028	380
DOUBLE ROTATE RIGHT	RORL	573	381
ROTATE RIGHT WITHOUT CARRY	RRNC	575	387
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	577	388
ONE DIGIT SHIFT LEFT	SLD	074	390
ONE DIGIT SHIFT RIGHT	SRD	075	392
SHIFT N-BIT DATA LEFT	NSFL	578	393
SHIFT N-BIT DATA RIGHT	NSFR	579	395
SHIFT N-BITS LEFT	NASL	580	397
DOUBLE SHIFT N-BITS LEFT	NSLL	582	400
SHIFT N-BITS RIGHT	NASR	581	403
DOUBLE SHIFT N-BITS RIGHT	NSRL	583	405

3-9-1 SHIFT REGISTER: SFT(010)

Purpose

Operates a shift register.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	SFT(010)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

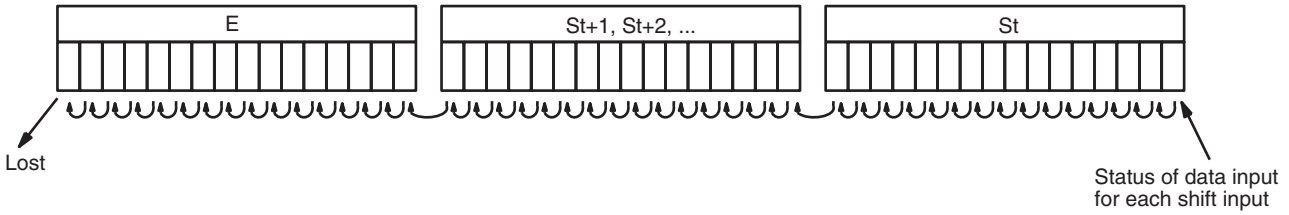
Note St and E must be in the same data area.

Operand Specifications

Area	St	E
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	---	
Counter Area	---	
DM Area	---	
EM Area without bank	---	
EM Area with bank	---	
Indirect DM/EM addresses in binary	---	
Indirect DM/EM addresses in BCD	---	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

When the execution condition on the shift input changes from OFF to ON, all the data from St to E is shifted to the left by one bit (from the rightmost bit to the leftmost bit), and the ON/OFF status of the data input is placed in the rightmost bit.



Flags

Name	Label	Operation
Error Flag	ER	ON if the indirect IR address for St and E is not in the CIO, AR, HR, or WR data areas. OFF in all other cases.

Precautions

The results will not be predictable if two SFT(010) instructions are used with overlapping shift registers. All words in the range ST to E must be used in only one SFT(010) instruction.

The bit data shifted out of the shift register is discarded.

When the reset input turns ON, all bits in the shift register from the rightmost designated word (St) to the leftmost designated word (E) will be reset (i.e., set to 0). The reset input takes priority over other inputs.

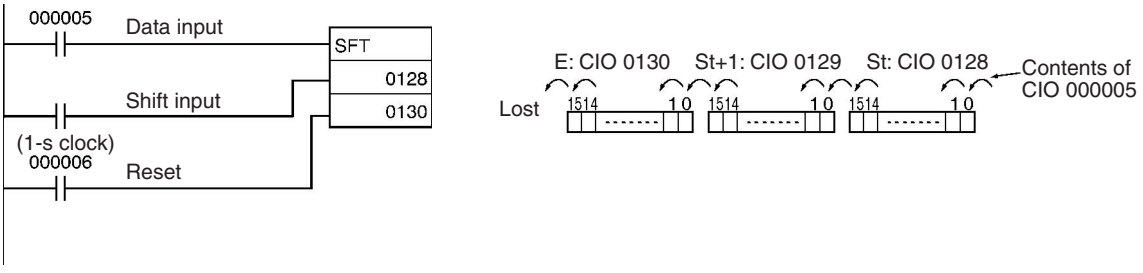
St must be less than or equal to E, but even when St is set to greater than E an error will not occur and one word of data in St will be shifted.

When St and E are designated indirectly using index registers and the actual addresses in I/O memory are not within memory areas for data, an error will occur and the Error Flag will turn ON.

Examples

Shift Register Exceeding 16 Bits

The following example shows a 48-bit shift register using words CIO 0128 to CIO 0130. A 1-s clock pulse is used so that the execution condition produced by CIO 000005 is shifted into a 3-word register between CIO 012800 and CIO 013015 every second.

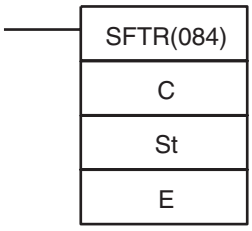


3-9-2 REVERSIBLE SHIFT REGISTER: SFTR(084)

Purpose

Creates a shift register that shifts data to either the right or the left.

Ladder Symbol



C: Control word

St: Starting word

E: End word

Variations

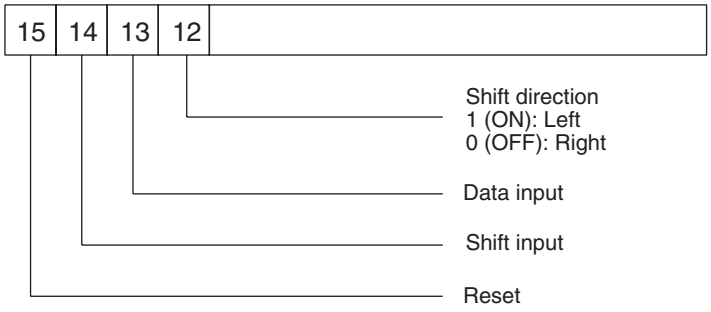
Variations		
	Executed Each Cycle for ON Condition	SFTR(084)
	Executed Once for Upward Differentiation	@SFTR(084)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word



Note St and E must be in the same data area.

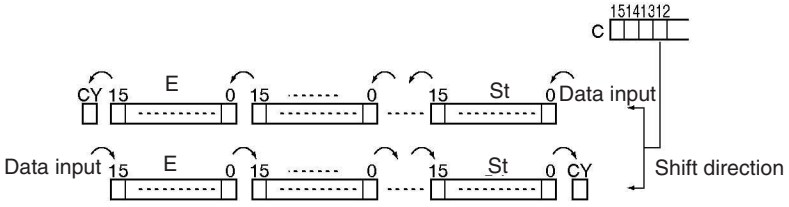
Operand Specifications

Area	C	St	E
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959	A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	DR0 to DR15	---	

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

Description

When the execution condition of the shift input bit (bit 14 of C) changes to ON, all the data from St to E is moved in the designated shift direction (designated by bit 12 of C) by 1 bit, and the ON/OFF status of the data input is placed in the rightmost or leftmost bit. The bit data shifted out of the shift register is placed in the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into it. OFF when 0 is shifted into it. OFF when reset is set to 1.

Precautions

The above shift operations are applicable when the reset bit (bit 15 of C) is set to OFF.

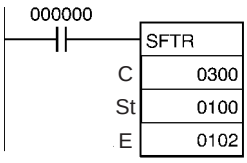
When reset (bit 15 of C) turns ON all bits in the shift register, from St to E will be reset (i.e., set to 0).

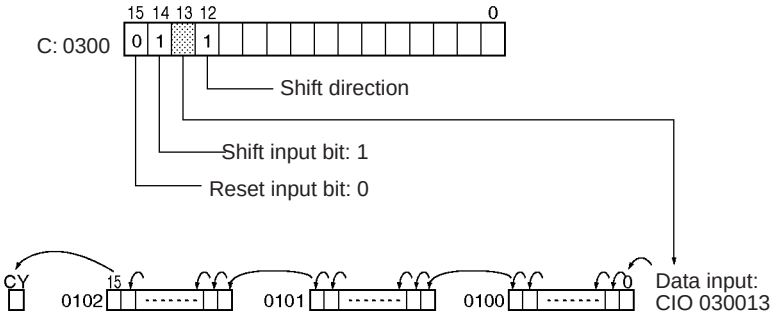
When St is greater than E, an error will be generated and the Error Flag will turn ON.

Examples

Shifting Data

If shift input CIO 030014 goes ON when CIO 000000 is ON, and the reset bit CIO 030015 is OFF, words CIO 0100 through CIO 0102 will shift one bit in the direction designated by CIO 030012 (e.g., 1: Right) and the contents of input bit CIO 030013 will be shifted into the rightmost bit, CIO 010000. The contents of CIO 010215 will be shifted to the Carry Flag (CY).





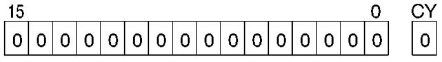
Resetting Data

If CIO 030014 is ON when CIO 000000 is ON, and the reset bit, CIO 030015, is ON, words CIO 0100 through CIO 0102 and the Carry Flag will be reset to OFF.

Controlling Data

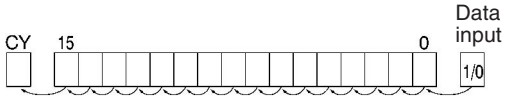
Resetting Data

All bits from St to E and the Carry Flag are set to 0 and no other data can be received when the reset input bit (bit 15 of C) is ON.



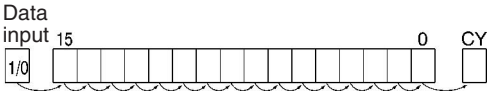
Shifting Data Left (from Rightmost to Leftmost Bit)

When the shift input bit (bit 14 of C) is ON, the contents of the input bit (bit 13 of C) is shifted to bit 00 of the starting word, and each bit thereafter is shifted one bit to the left. The status of bit 15 of the end word is shifted to the Carry Flag.



Shifting Data Right (from Leftmost to Rightmost Bit

When the shift input bit (bit 14 of C) is ON, the contents of the input bit (bit 13 of C) (I/O) is shifted to bit 15 on the end word, and each bit thereafter is shifted one bit to the right. The status of bit 00 of the starting word is shifted to the Carry Flag.

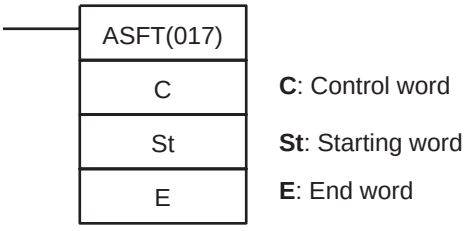


3-9-3 ASYNCHRONOUS SHIFT REGISTER: ASFT(017)

Purpose

Shifts all non-zero word data within the specified word range either towards St or toward E, replacing 0000Hex word data.

Ladder Symbol



Variations

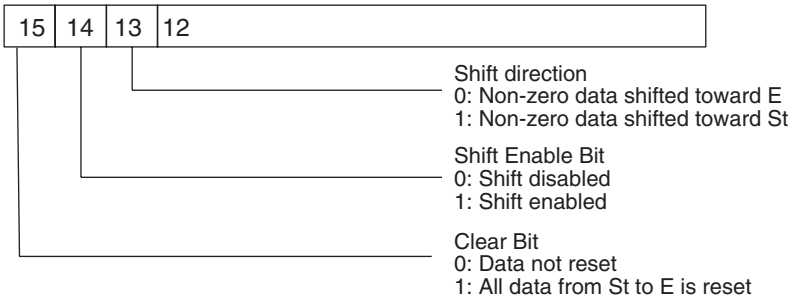
Variations		
	Executed Each Cycle for ON Condition	ASFT(017)
	Executed Once for Upward Differentiation	@ASFT(017)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word



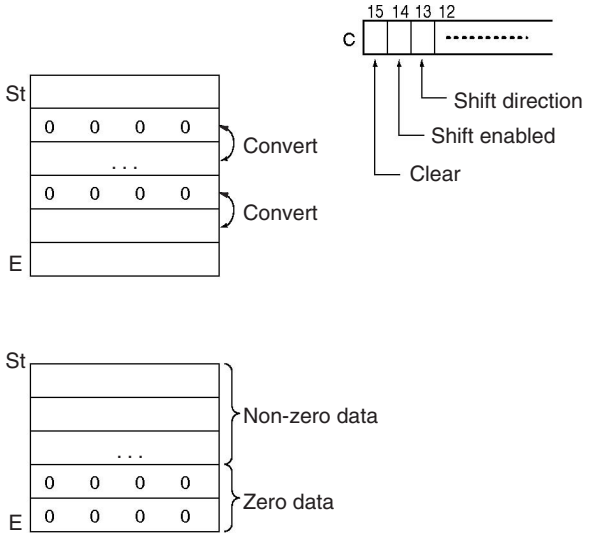
Note St and E must be in the same data area.

Operand Specifications

Area	C	St	E
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959	A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

When the 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 ASFT(017) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

Precautions

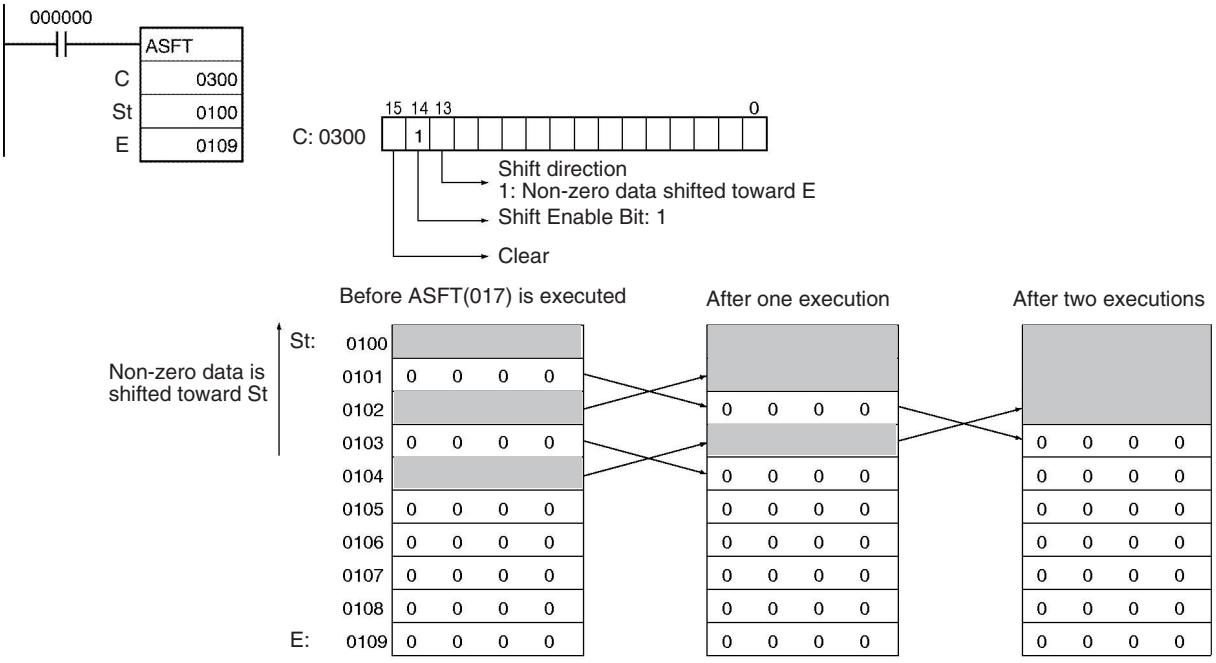
When the Clear Flag (bit 15 of C) goes ON, all bits in the shift register, from St to E, will be reset (i.e., set to 0). The Clear Flag has priority over the Shift Enable Bit (bit 14 of C).

When St is greater than E an error will be generated and the Error Flag will turn ON.

Examples

Shifting Data:

If the Shift Enable Bit, CIO 030014, goes ON when CIO 000000 is ON, all words with non-zero data content from CIO 0100 through CIO 0109 will be shifted in the direction designated by the Shift Direction Bit, CIO 030013 (e.g., 1: Toward St) if the word to the left of the non-zero data is all zeros.

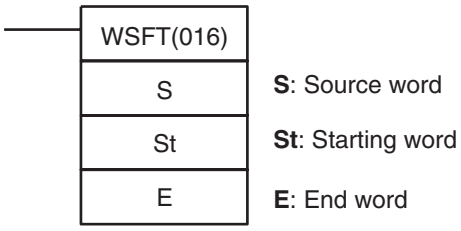


3-9-4 WORD SHIFT: WSFT(016)

Purpose

Shifts data between St and E in word units.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	WSFT(016)
	Executed Once for Upward Differentiation	@WSFT(016)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Note St and E must be in the same data area.

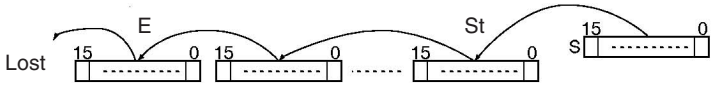
Operand Specifications

Area	S	St	E
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959	A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		

Area	S	St	E
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

WSFT(016) shifts data from St to E in word units and the data from the source word S is places into St. The contents of E is lost.



Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. OFF in all other cases.

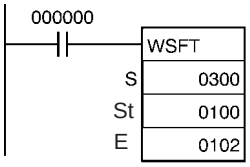
Precautions

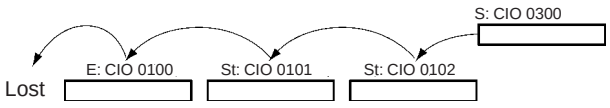
When St is greater than E, an error will be generated and the Error Flag will turn ON.

Note When large amounts of data are shifted, the instruction execution time is quite long. Be sure that the power is not cut while WSFT(016) is being executed, causing the shift operation to stop halfway through.

Examples

When CIO 000000 is ON, data from CIO 0100 through CIO 0102 will be shifted one word toward E. The contents of CIO 0300 will be stored in CIO 0100 and the contents of CIO 0102 will be lost.



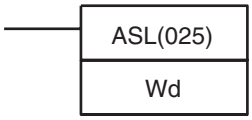


3-9-5 ARITHMETIC SHIFT LEFT: ASL(025)

Purpose

Shifts the contents of Wd one bit to the left.

Ladder Symbol



Wd: Word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	ASL(025)
	Executed Once for Upward Differentiation	@ASL(025)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

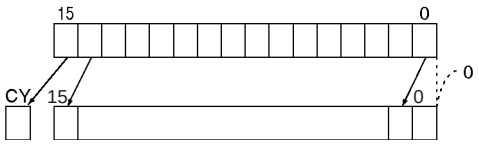
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

ASL(025) shifts the contents of Wd one bit to the left (from rightmost bit to leftmost bit). “0” is placed in the rightmost bit and the data from the leftmost bit is shifted into the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

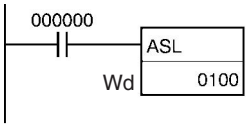
When ASL(025) is executed, the Error Flag will turn OFF.

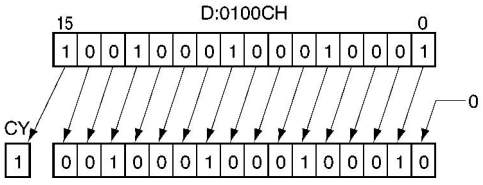
If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, CIO 0100 will be shifted one bit to the left. “0” will be placed in CIO 010000 and the contents of CIO 010115 will be shifted to the Carry Flag (CY).



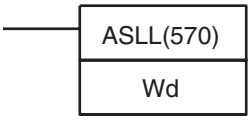


3-9-6 DOUBLE SHIFT LEFT: ASLL(570)

Purpose

Shifts the contents of Wd and Wd +1 one bit to the left.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASLL(570)
	Executed Once for Upward Differentiation	@ASLL(570)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

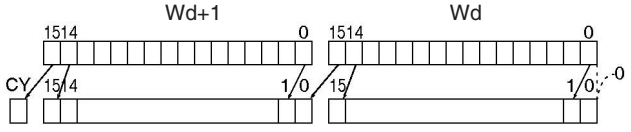
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510

Area	Wd
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

ASLL(570) shifts the contents of Wd and Wd +1 one bit to the left (from rightmost bit to leftmost bit). “0” is placed in the rightmost bit of Wd and the contents of the leftmost bit of Wd and Wd +1 are shifted into the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

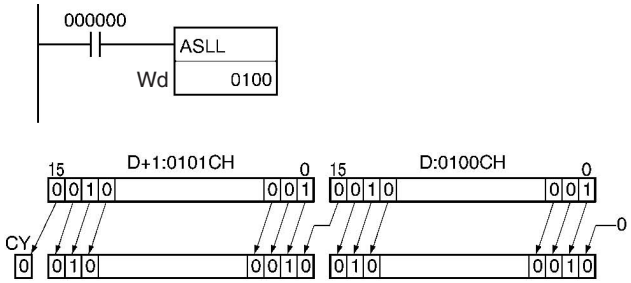
When ASLL(570) is executed, the Error Flag will turn OFF.

If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd +1 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, word CIO 0100 and CIO 0101 will shift one bit to the left. “0” is placed into CIO 010000 and the contents of CIO 010015 will be shifted to the Carry Flag (CY).



3-9-7 ARITHMETIC SHIFT RIGHT: ASR(026)

Purpose

Shifts the contents of Wd one bit to the right.

Ladder Symbol

Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASR(026)
	Executed Once for Upward Differentiation	@ASR(026)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

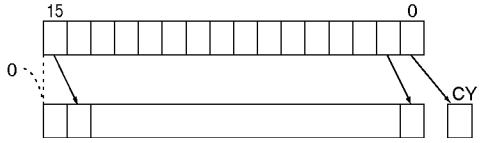
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	DR0 to DR15

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

Description

ASR(026) shifts the contents of Wd one bit to the right (from leftmost bit to rightmost bit). “0” will be placed in the leftmost bit and the contents of the rightmost bit will be shifted into the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	OFF

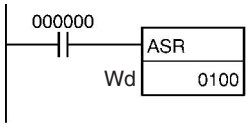
Precautions

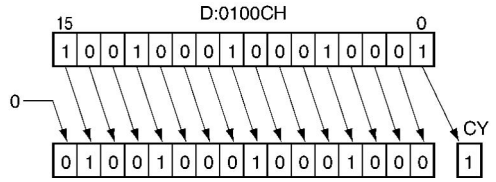
When ASR(026) is executed, the Error Flag and the Negative Flag will turn OFF.

If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

Examples

When CIO 000000 is ON, word CIO 0100 will shift one bit to the right. “0” will be placed in CIO 010015 and the contents of CIO 010000 will be shifted to the Carry Flag (CY).



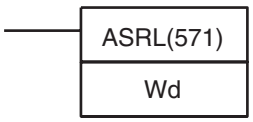


3-9-8 DOUBLE SHIFT RIGHT: ASRL(571)

Purpose

Shifts the contents of Wd and Wd +1 one bit to the right.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASRL(571)
	Executed Once for Upward Differentiation	@ASRL(571)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

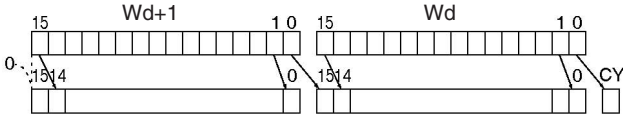
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

ASRL(571) shifts the contents of Wd and Wd +1 one bit to the right (from leftmost bit to rightmost bit). “0” will be placed in the leftmost bit of Wd +1 and the contents of the rightmost bit of Wd will be shifted into the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.

Name	Label	Operation
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	OFF

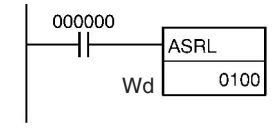
Precautions

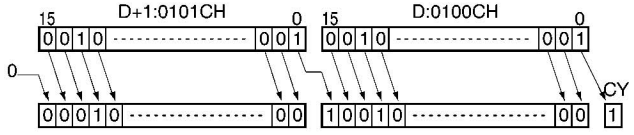
When ASRL (571) is executed, the Error Flag and the Negative Flag will turn OFF.

If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

Examples

When CIO 000000 is ON, word CIO 0100 and CIO 0101 will shift one bit to the right. “0” will be placed into CIO 010115 and the contents of CIO 010000 will be shifted to the Carry Flag (CY).



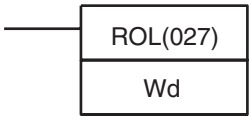


3-9-9 ROTATE LEFT: ROL(027)

Purpose

Shifts all Wd bits one bit to the left including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	ROL(027)
	Executed Once for Upward Differentiation	@ROL(027)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

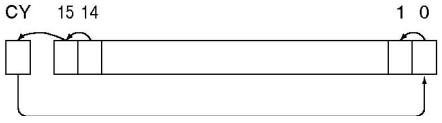
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)

Area	Wd
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

ROL(027) shifts all bits of Wd including the Carry Flag (CY) to the left (from rightmost bit to leftmost bit).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

When ROL(027) is executed, the Error Flag will turn OFF.

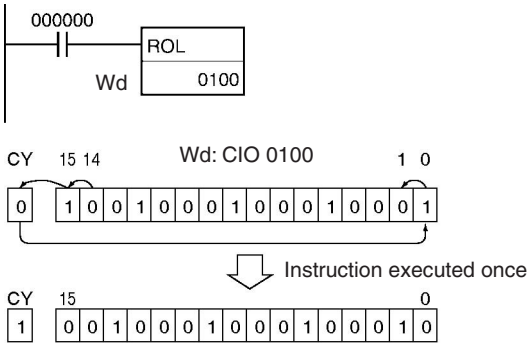
If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

Note It is possible to set the Carry Flag contents to 1 or 0 immediately before executing this instruction, by using the Set Carry (STC(040)) or Clear Carry (CLC(041)) instructions.

Examples

When CIO 000000 is ON, word CIO 0100 and the Carry Flag (CY) will shift one bit to the left. The contents of CIO 010015 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 010000.

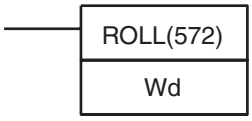


3-9-10 DOUBLE ROTATE LEFT: ROLL(572)

Purpose

Shifts all Wd and Wd +1 bits one bit to the left including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations	**Executed Each Cycle for ON Condition**	ROLL(572)
	Executed Once for Upward Differentiation	@ROLL(572)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

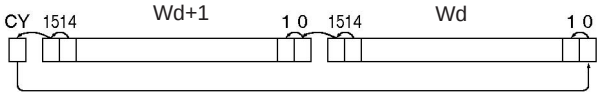
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---

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

Description

ROLL(572) shifts all bits of Wd and Wd +1 including the Carry Flag (CY) to the left (from rightmost bit to leftmost bit).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

When ROLL(572) is executed, the Error Flag will turn OFF.

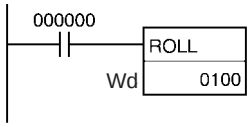
If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

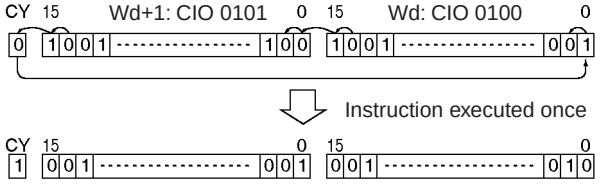
If as a result of the shift the contents of the leftmost bit of Wd + 1 is 1, the Negative Flag will turn ON.

Note It is possible to set the Carry Flag contents to 1 or 0 immediately before executing this instruction, by using the Set Carry (STC(040)) or Clear Carry (CLC(041)) instructions.

Examples

When CIO 000000 is ON, word CIO 0100, CIO 0101 and the Carry Flag (CY) will shift one bit to the left. The contents of CIO 010015 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 010000.



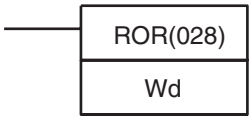


3-9-11 ROTATE RIGHT: ROR(028)

Purpose

Shifts all Wd bits one bit to the right including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	ROR(028)
	Executed Once for Upward Differentiation	@ROR(028)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

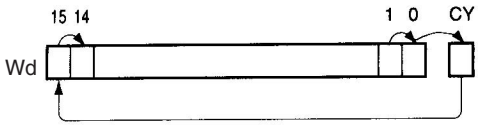
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

ROR(028) shifts all bits of Wd including the Carry Flag (CY) to the right (from leftmost bit to rightmost bit).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

When ROR(028) is executed, the Error Flag will turn OFF.

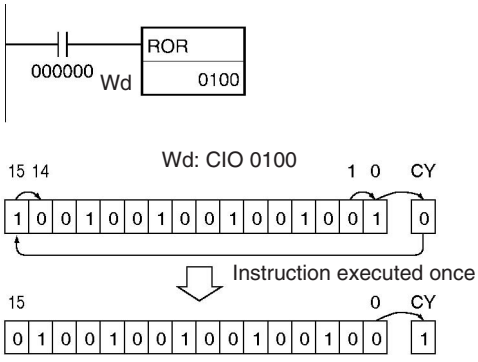
If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

Note It is possible to set the Carry Flag contents to 1 or 0 immediately before executing this instruction, by using the Set Carry (STC(040)) or Clear Carry (CLC(041)) instructions.

Examples

When CIO 000000 is ON, word CIO 0100 and the Carry Flag (CY) will shift one bit to the right. The contents of CIO 010000 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 010015.

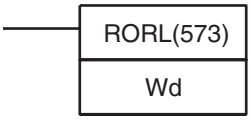


3-9-12 DOUBLE ROTATE RIGHT: RORL(573)

Purpose

Shifts all Wd and Wd +1 bits one bit to the right including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	RORL(573)
	Executed Once for Upward Differentiation	@RORL(573)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

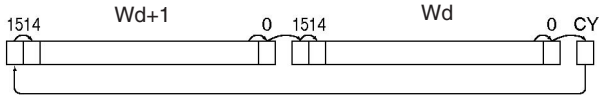
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

RORL(573) shifts all bits of Wd and Wd +1 including the Carry Flag (CY) to the right (from leftmost bit to rightmost bit).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

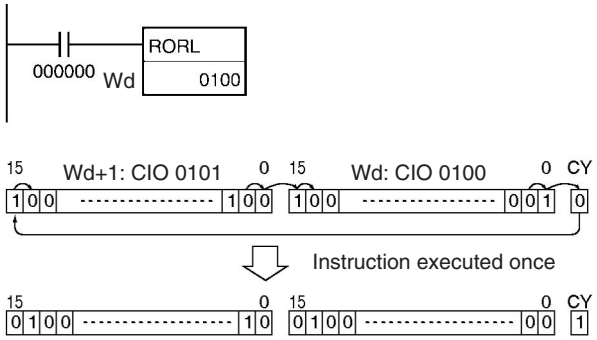
When RORL(573) is executed, the Error Flag will turn OFF.

If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd + 1 is 1, the Negative Flag will turn ON.

Note It is possible to set the Carry Flag contents to 1 or 0 immediately before executing this instruction, by using the Set Carry (STC(040)) or Clear Carry (CLC(041)) instructions.

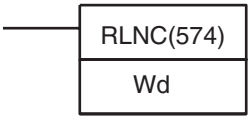
Examples When CIO 000000 is ON, word CIO 0100, CIO 0101 and the Carry Flag (CY) will shift one bit to the right. The contents of CIO 010000 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 010115.



3-9-13 ROTATE LEFT WITHOUT CARRY: RLNC(574)

Purpose Shifts all Wd bits one bit to the left not including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	RLNC(574)
	Executed Once for Upward Differentiation	@RLNC(574)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

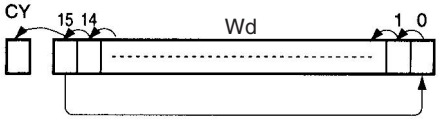
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)

Area	Wd
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

RLNC(574) shifts all bits of Wd to the left (from rightmost bit to leftmost bit). The contents of the leftmost bit of Wd shifts to the rightmost bit and to the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

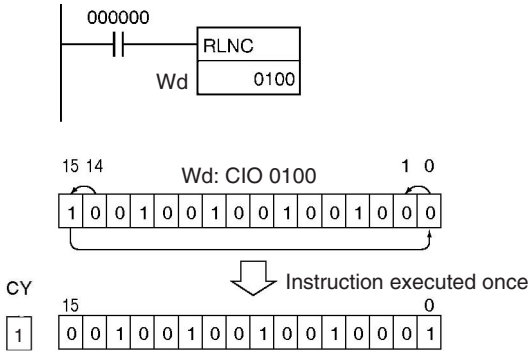
When RLNC(574) is executed, the Error Flag will turn OFF.

If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, word CIO 0100 will shift one bit to the left (excluding the Carry Flag (CY)). The contents of CIO 010015 will be shifted to CIO 010000.

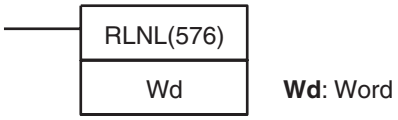


3-9-14 DOUBLE ROTATE LEFT WITHOUT CARRY: RLNL(576)

Purpose

Shifts all Wd and Wd +1 bits one bit to the left not including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	RLNL(576)
	Executed Once for Upward Differentiation	@RLNL(576)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

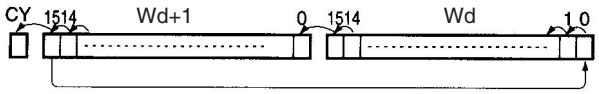
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---

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

Description

RLNL(576) shifts all bits of Wd and Wd +1 to the left (from rightmost bit to leftmost bit). The contents of the leftmost bit of Wd +1 is shifted to the rightmost bit of Wd, and to the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

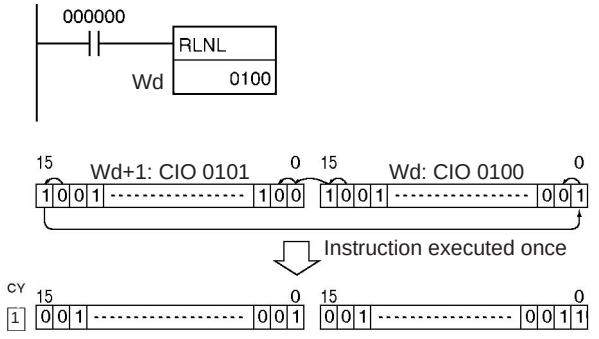
When RLNL(576) is executed, the Error Flag will turn OFF.

If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd + 1 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, word CIO 0100 and CIO 0101 will shift one bit to the left (excluding the Carry Flag (CY)). The contents of CIO 010115 will be shifted to CIO 010000.

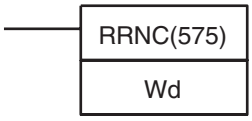


3-9-15 ROTATE RIGHT WITHOUT CARRY: RRNC(575)

Purpose

Shifts all Wd bits one bit to the right not including the Carry Flag (CY). The contents of the rightmost bit of Wd shifts to the leftmost bit and to the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	RRNC(575)
	Executed Once for Upward Differentiation	@RRNC(575)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

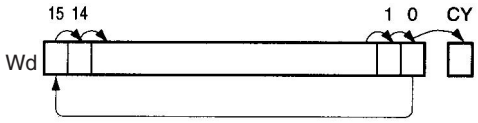
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

RRNC(575) shifts all bits of Wd to the right (from leftmost bit to rightmost bit) not including the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

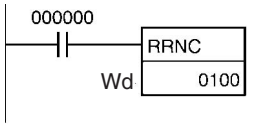
When RRNC(575) is executed, the Error Flag will turn OFF.

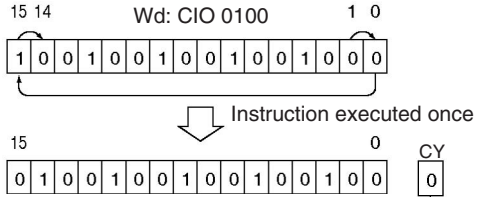
If as a result of the shift the contents of Wd is zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, word CIO 0100 will shift one bit to the right (excluding the Carry Flag (CY)). The contents of CIO 010000 will be shifted to CIO 010015.



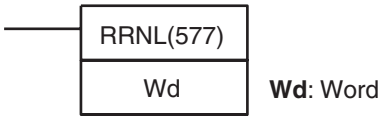


3-9-16 DOUBLE ROTATE RIGHT WITHOUT CARRY: RRNL(577)

Purpose

Shifts all Wd and Wd +1 bits one bit to the right not including the Carry Flag (CY). The contents of the rightmost bit of Wd +1 is shifted to the leftmost bit of Wd, and to the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	RRNL(577)
	Executed Once for Upward Differentiation	@RRNL(577)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

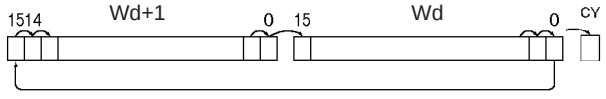
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

RRNL(577) shifts all bits of Wd and Wd +1 to the right (from leftmost bit to rightmost bit) not including the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

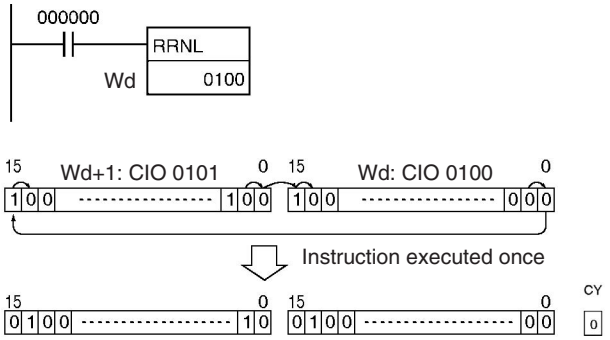
When RRNL(577) is executed, the Error Flag will turn OFF.

If as a result of the shift the contents of Wd and Wd +1 are zero, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of Wd + 1 is 1, the Negative Flag will turn ON.

Note It is possible to set the Carry Flag contents to 1 or 0 immediately before executing this instruction, by using the Set Carry (STC(040)) or Clear Carry (CLC(041)) instructions.

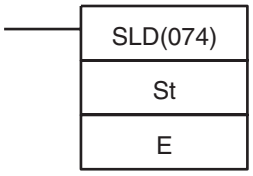
Examples When CIO 000000 is ON, words CIO 0100 and CIO 0101 will shift one bit to the right, (excluding the Carry Flag (CY)). The contents of CIO 010000 will be shifted to CIO 010115.



3-9-17 ONE DIGIT SHIFT LEFT: SLD(074)

Purpose Shifts data by one digit (4 bits) to the left.

Ladder Symbol



St: Starting word

E: End word

Variations

Variations	**Executed Each Cycle for ON Condition**	SLD(074)
	Executed Once for Upward Differentiation	@SLD(074)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Note St and E must be in the same data area.

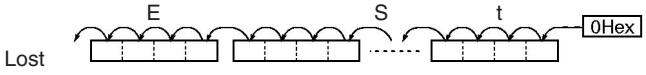
Operand Specifications

Area	St	E
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	

Area	St	E
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

SLD(074) shifts data between St and E by one digit (4 bits) to the left. “0” is placed in the rightmost digit (bits 3 to 0 of St), and the content of the leftmost digit (bits 15 to 12 of E) is lost.



Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. OFF in all other cases.

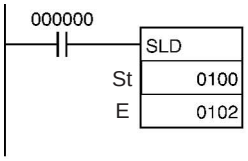
Precautions

When St is greater than E, an error will be generated and the Error Flag will turn ON.

Note When large amounts of data are shifted, the instruction execution time is quite long. Be sure that the power is not cut while SLD(074) is being executed, causing the shift operation to stop halfway through.

Examples

When CIO 000000 is ON, words CIO 0100 through CIO 0102 will shift by one digit (4 bits) to the left. A zero will be placed in bits 0 to 3 of word CIO 0100 and the contents of bits 12 to 15 of CIO 0102 will be lost.

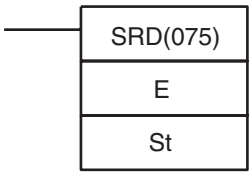


3-9-18 ONE DIGIT SHIFT RIGHT: SRD(075)

Purpose

Shifts data by one digit (4 bits) to the right.

Ladder Symbol



E: End word

St: Starting word

Variations

Variations	**Executed Each Cycle for ON Condition**	SRD(075)
	Executed Once for Upward Differentiation	@SRD(075)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

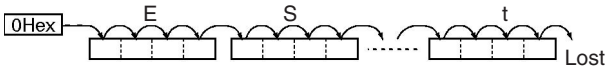
Note St and E must be in the same data area.

Operand Specifications

Area	St	E
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

SRD(075) shifts data between St and E by one digit (4 bits) to the right. “0” is placed in the leftmost digit (bits 15 to 12 of E), and the content of the rightmost digit (bits 3 to 0 of St) is lost.



Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. OFF in all other cases.

Precautions

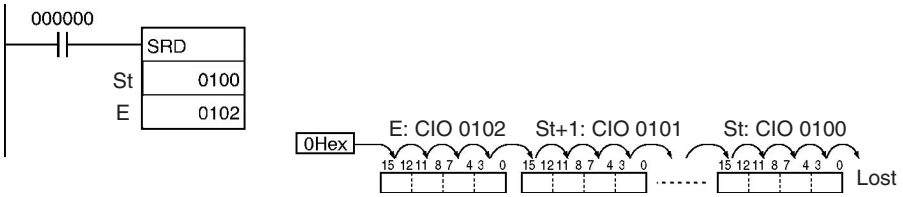
When St is greater than E, an error will be generated and the Error Flag will turn ON.

When SRD(075) is executed, the Equals Flag and Negative Flag will turn OFF.

Note When large amounts of data are shifted, the instruction execution time is quite long. Always take care that the power is not cut while SRD(075) is being executed, causing the shift operation to stop halfway through.

Examples

When CIO 000000 is ON, words CIO 0100 through CIO 0102 will shift by one digit (4 bits) to the right. A zero will be placed in bits 12 to 15 of CIO 0102 and the contents of bits 0 to 3 of word CIO 0100 will be lost.

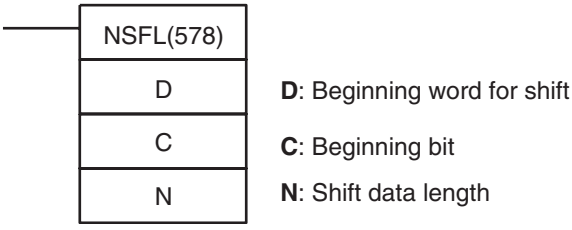


3-9-19 SHIFT N-BIT DATA LEFT: NSFL(578)

Purpose

Shifts the specified number of bits to the left.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	NSFL(578)
	Executed Once for Upward Differentiation	@NSFL(578)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: 0000 to 000F hex (0 to 15)
N: 0000 to FFFF hex (0 to 65535)

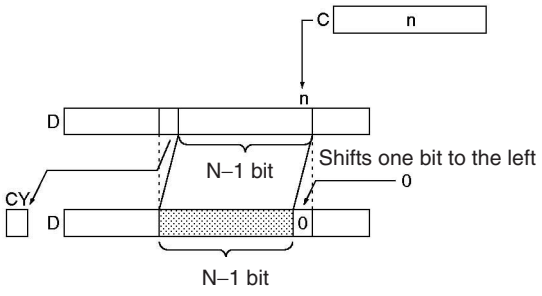
Note All words in the shift register must be in the same area.

Operand Specifications

Area	D	C	N
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A448 to A959	A000 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0000 to #000F (binary) or &0 to &15	#0000 to #FFFF (binary) or &0 to &65535
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

NSFL(578) shifts the specified number of bits by the shift data length (N) from the beginning bit (C) in the rightmost word, as designated by D one bit to the left (towards the leftmost word and the leftmost bit). “0” is place into the beginning bit and the contents of the leftmost bit in the shift area are shifted to the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	ON when C data is not between 0000 and 000F hex. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.

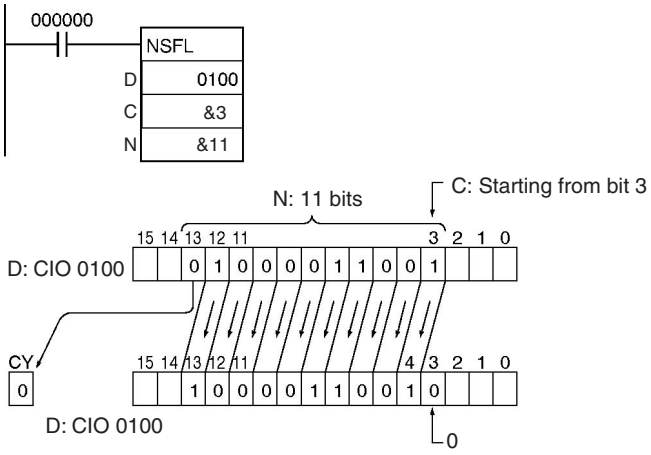
Precautions

When the shift data length (N) is 0, the contents of the beginning bit will be copied to the Carry Flag (CY), and its contents will not be changed.

Only the bits shifted into rightmost word in the shift area (i.e. leftmost word data) will be changed.

Examples

When CIO 000000 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 0100. The contents of the leftmost bit in the shift area (bit 13 of CIO 0100) are copied into the Carry Flag (CY).

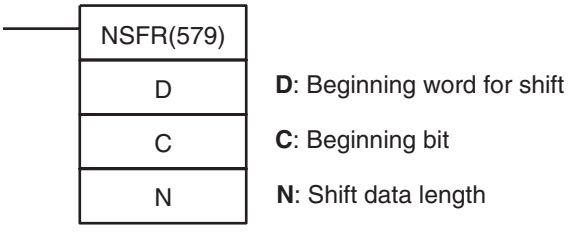


3-9-20 SHIFT N-BIT DATA RIGHT: NSFR(579)

Purpose

Shifts the specified number of bits to the right.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	NSFR(579)
	Executed Once for Upward Differentiation	@NSFR(579)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: 0000 to 000F hex (0 to 15)
N: 0000 to FFFF hex (0 to 65535)

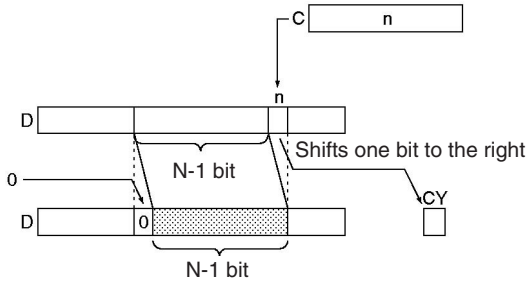
Note All words in the shift register must be in the same area.

Operand Specifications

Area	D	C	N
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A448 to A959	A000 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0000 to #000F (binary) or &0 to &15	#0000 to #FFFF (binary) or &0 to &65535
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

NSFR(579) shifts the specified number of bits by the shift data length (N) from the beginning bit (C) in the rightmost word as designated by D one bit to the right (towards the rightmost word and the rightmost bit). “0” will be placed into the beginning bit and the contents of the rightmost bit in the shift area will be shifted to the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	ON when C data is not between 0000 and 000F hex. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.

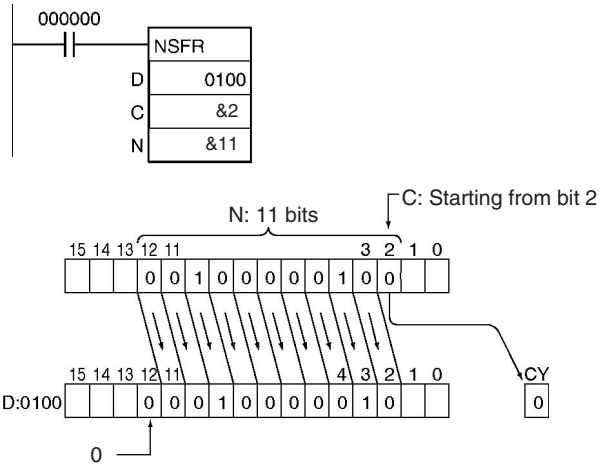
Precautions

When the shift data length (N) is 0, the contents of the beginning bit will be copied to the Carry Flag (CY), and its contents will not be changed.

Only the bits shifted into rightmost word in the shift area (i.e. leftmost word data) will be changed.

Examples

When CIO 000000 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 0100. The contents of the rightmost bit in the shift area (bit 2 of CIO 0100) are copied into the Carry Flag (CY).

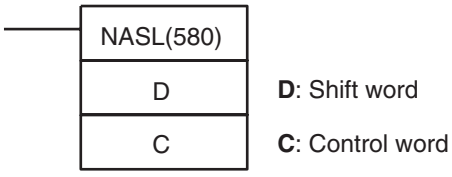


3-9-21 SHIFT N-BITS LEFT: NASL(580)

Purpose

Shifts the specified 16 bits of word data to the left by the specified number of bits.

Ladder Symbol



D: Shift word

C: Control word

Variations

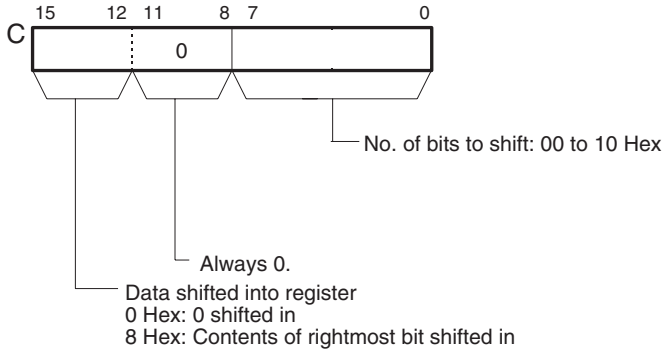
Variations		
	Executed Each Cycle for ON Condition	NASL(580)
	Executed Once for Upward Differentiation	@NASL(580)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word

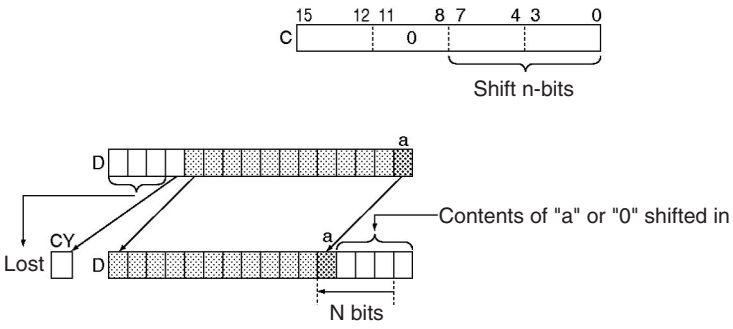


Operand Specifications

Area	D	C
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	A000 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	Specified values only
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

NASL(580) shifts D (the shift word) by the specified number of binary bits (specified in C) to the left (from the rightmost bit to the leftmost bit). Either zeros or the value of the rightmost bit will be placed into the specified number of bits of the shift word starting from the rightmost bit.



Flags

Name	Label	Operation
Error Flag	ER	ON when the control word C (the number of bits to shift) is not within range. OFF in all other cases.
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is lost.

When the number of bits to shift (specified in C) is “0,” the data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.

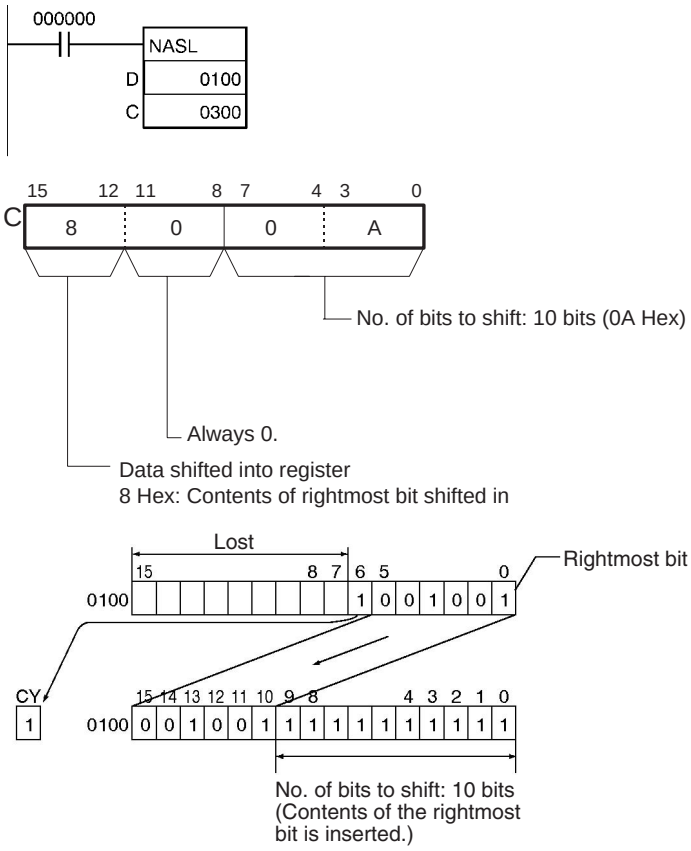
When the contents of the control word C is out of range, an error will be generated and the Error Flag will turn ON.

If as a result of the shift the contents of D is 0000 hex, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of D is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, The contents of CIO 0100 is shifted 10 bits to the left (from the rightmost bit to the leftmost bit). The number of bits to shift is specified in bits 0 to 7 of word CIO 0300 (control data). The contents of bit 0 of CIO 0100 is copied into bits from which data was shifted and the contents of the rightmost bit which was shifted out of range is shifted into the Carry Flag (CY). All other data is lost.

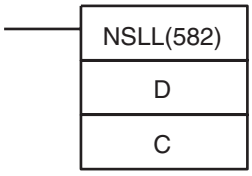


3-9-22 DOUBLE SHIFT N-BITS LEFT: NSLL(582)

Purpose

Shifts the specified 32 bits of word data to the left by the specified number of bits.

Ladder Symbol



D: Shift word

C: Control word

Variations

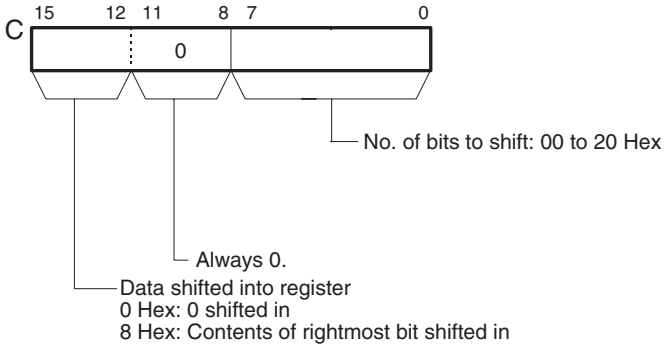
Variations	**Executed Each Cycle for ON Condition**	NSLL(582)
	Executed Once for Upward Differentiation	@NSLL(582)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word

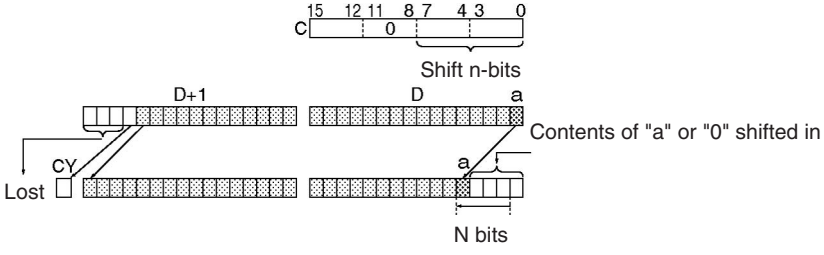


Operand Specifications

Area	D	C
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W510	W000 to W511
Holding Bit Area	H000 to H510	H000 to H511
Auxiliary Bit Area	A448 to A958	A000 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	Specified values only
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

NSLL(582) shifts D and D+1 (the shift words) by the specified number of binary bits (specified in C) to the left (from the rightmost bit to the leftmost bit). Either zeros or the value of the rightmost bit will be placed into the specified number of bits of the shift word starting from the rightmost bit.



Flags

Name	Label	Operation
Error Flag	ER	ON when the control word C (the number of bits to shift) is not within range. OFF in all other cases.
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is lost.

When the number of bits to shift (specified in C) is “0,” the data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.

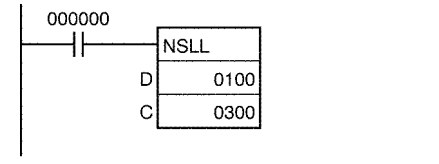
When the contents of the control word C are out of range, an error will be generated and the Error Flag will turn ON.

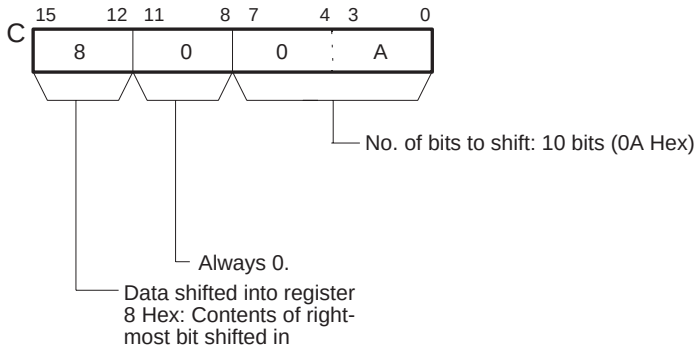
If as a result of the shift the contents of D is 0000, the Equals Flag will turn ON.

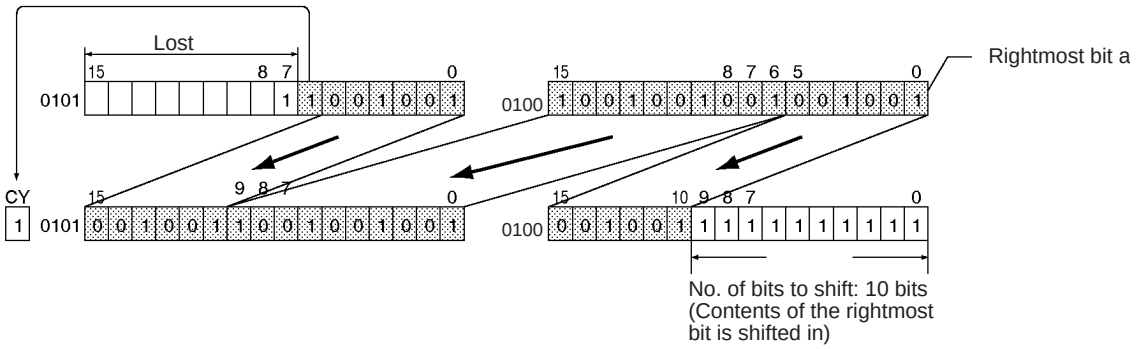
If as a result of the shift the contents of the leftmost bit of D, D +1 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, CIO 0100 and CIO 0101 will be shifted to the left (from the rightmost bit to the leftmost bit) by 10 bits. The number of bits to shift is specified in bits 0 to 7 of word CIO 0300 (control data). The contents of bit 0 of CIO 0100 is copied into bits from which data was shifted and the contents of the rightmost bit which was shifted out of range is shifted into the Carry Flag (CY). All other data is lost.





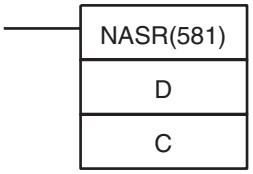


3-9-23 SHIFT N-BITS RIGHT: NASR(581)

Purpose

Shifts the specified 16 bits of word data to the right by the specified number of bits.

Ladder Symbol



D: Shift word

C: Control word

Variations

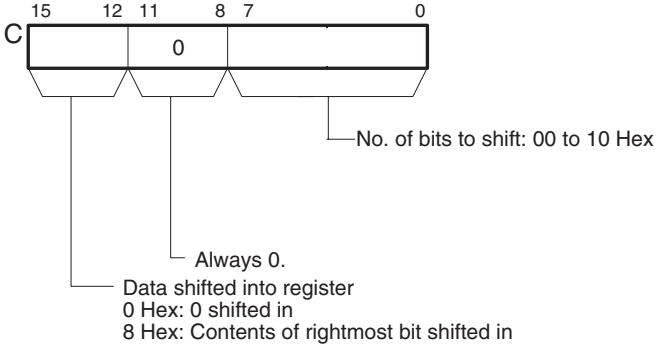
Variations		
	Executed Each Cycle for ON Condition	NASR(581)
	Executed Once for Upward Differentiation	@NASR(581)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word



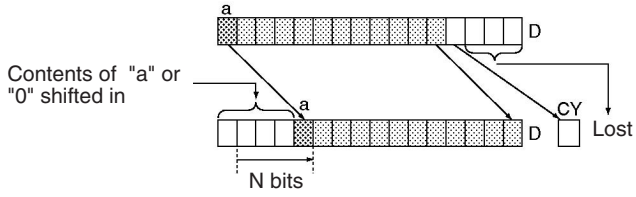
Operand Specifications

Area	D	C
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	A000 to A447 A448 to A959
Timer Area	T0000 to T4095	

Area	D	C
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	Specified values only
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

NASR(581) shifts D (the shift word) by the specified number of binary bits (specified in C) to the right (from the rightmost bit to the leftmost bit). Either zeros or the value of the rightmost bit will be placed into the specified number of bits of the shift word starting from the rightmost bit.



Flags

Name	Label	Operation
Error Flag	ER	ON when the control word C (the number of bits to shift) is not within range. OFF in all other cases.
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is discarded.

When the number of bits to shift (specified in C) is “0,” the data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.

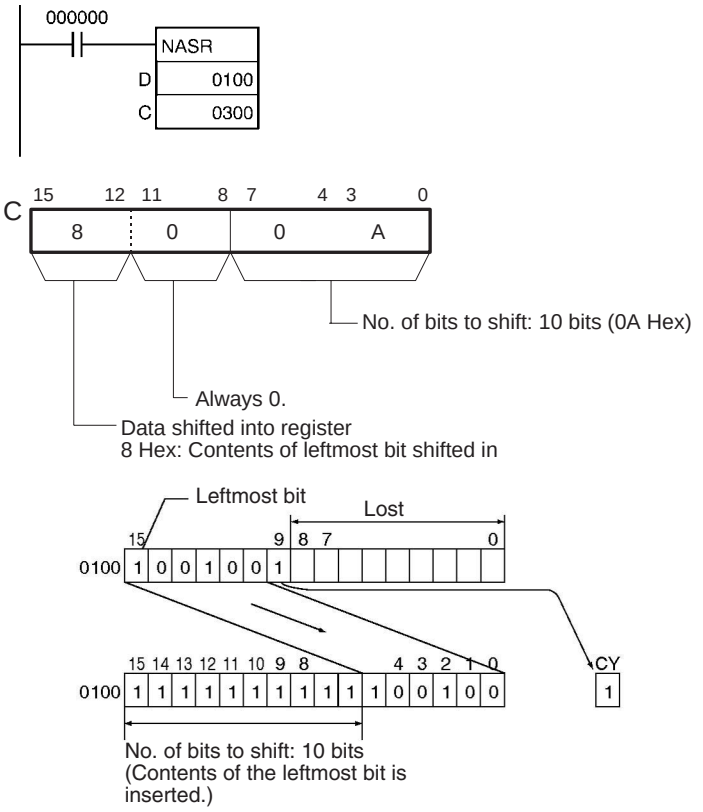
When the contents of the control word C are out of range, an error will be generated and the Error Flag will turn ON.

If as a result of the shift the contents of D is 0000 hex, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of D is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, CIO 0100 will be shifted 10 bits to the right (from the leftmost bit to the rightmost bit). The number of bits to shift is specified in bits 0 to 7 of word CIO 0300. The contents of bit 15 of CIO 0100 is copied into the bits from which data was shifted and the contents of the leftmost bit of data which was shifted out of range, is shifted into the Carry Flag (CY). All other data is lost.

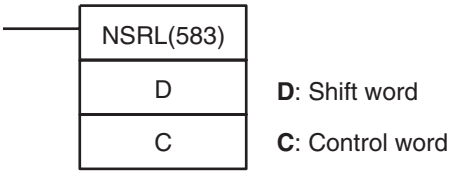


3-9-24 DOUBLE SHIFT N-BITS RIGHT: NSRL(583)

Purpose

Shifts the specified 32 bits of word data to the right by the specified number of bits.

Ladder Symbol



Variations

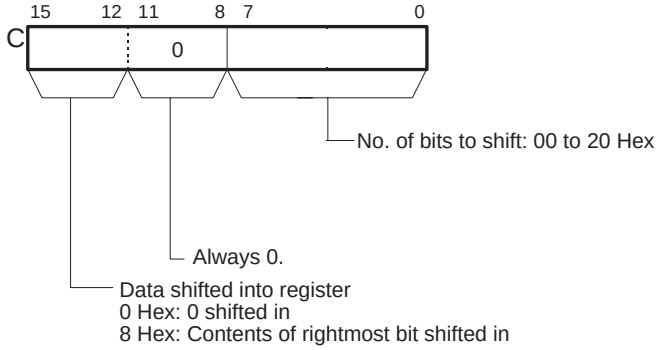
Variations		
	Executed Each Cycle for ON Condition	NSRL(583)
	Executed Once for Upward Differentiation	@NSRL(583)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word

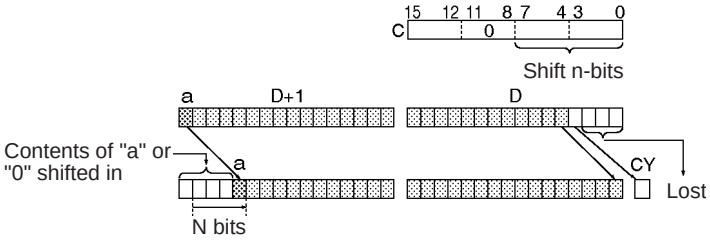


Operand Specifications

Area	D	C
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W510	W000 to W511
Holding Bit Area	H000 to H510	H000 to H511
Auxiliary Bit Area	A448 to A958	A000 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	Specified values only
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 -2048 to +2047 ,IR0 to -2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

NSRL(583) shifts D and D+1 (the shift words) by the specified number of binary bits (specified in C) to the right (from the leftmost bit to the rightmost bit). Either zeros or the value of the rightmost bit will be placed into the specified number of bits of the shift word starting from the rightmost bit.



Flags

Name	Label	Operation
Error Flag	ER	ON when the control word C (the number of bits to shift) is not within range. OFF in all other cases.
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is lost.

When the number of bits to shift (specified in C) is “0,” the data will not be shifted. The appropriate flags will turn ON or OFF, however, according to data in the specified word.

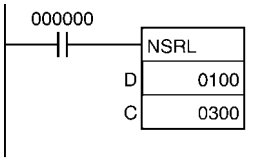
When the contents of the control word C are out of range, an error will be generated and the Error Flag will turn ON.

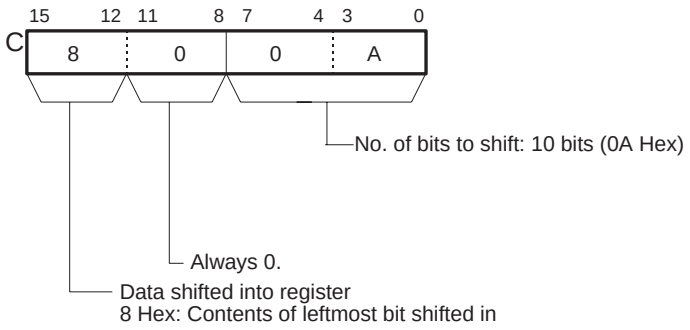
If as a result of the shift the contents of D +1 is 00000000 hex, the Equals Flag will turn ON.

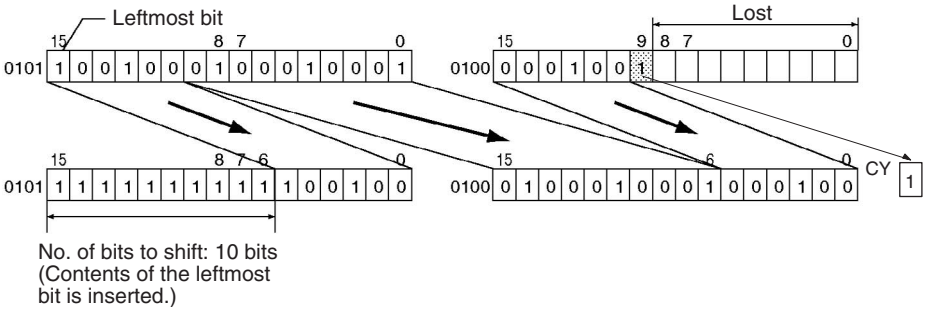
If as a result of the shift the contents of the leftmost bit of D +1 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, CIO 0100 and CIO 0101 will be shifted 10 bits to the right (from the leftmost bit to the rightmost bit). The number of bits to shift is specified in bits 0 to 7 of word CIO 0300 (control data). The contents of bit 15 of CIO will be copied into the bits from which data was shifted and the contents of the leftmost bit of data which was shifted out of range will be shifted into the Carry Flag (CY). All other data is lost.







No. of bits to shift: 10 bits
(Contents of the leftmost bit is inserted.)

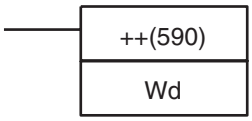
3-10 Increment/Decrement Instructions

3-10-1 INCREMENT BINARY: ++(590)

Purpose

Increments the 4-digit hexadecimal content of the specified word by 1.

Ladder Symbol



Wd: Word

Variations

Variations	**Executed Each Cycle for ON Condition**	++(590)
	Executed Once for Upward Differentiation	@++(590)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

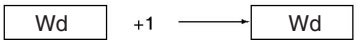
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

The ++(590) instruction adds 1 to the binary content of Wd. The specified word will be incremented by 1 every cycle as long as the execution condition of ++(590) is ON. When the up-differentiated variation of this instruction

(@++(590)) is used, the specified word is incremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000, the Carry Flag will be turned ON when a digit changes from F to 0, and the Negative Flag will be turned ON when bit 15 of Wd is ON in the result.

Both the Equals Flag and the Carry Flag will be turned ON when the content of Wd changes from FFFF to 0000.

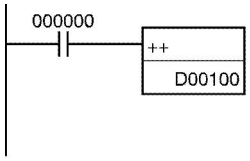
Flags

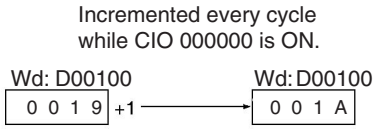
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the content of Wd is 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd went from F to 0 during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd is ON after execution. OFF in all other cases.

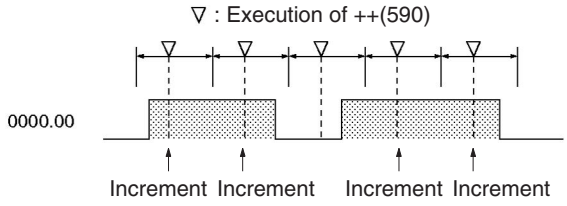
Examples

Operation of ++(590)

In the following example, the content of D00100 will be incremented by 1 every cycle as long as CIO 000000 is ON.

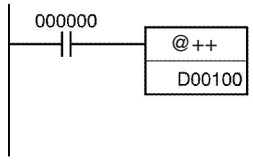


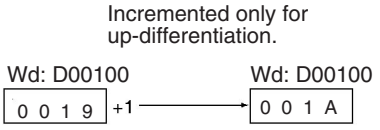


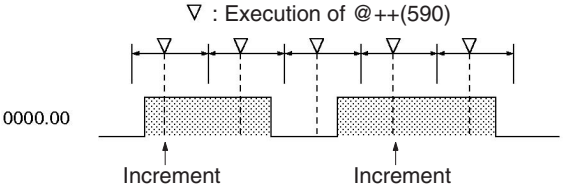


Operation of @++(590)

The up-differentiated variation is used in the following example, so the content of D00100 will be incremented by 1 only when CIO 000000 has gone from OFF to ON.





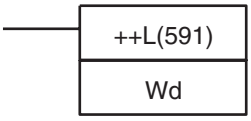


3-10-2 DOUBLE INCREMENT BINARY: ++L(591)

Purpose

Increments the 8-digit hexadecimal content of the specified words by 1.

Ladder Symbol



Wd: First word

Variations

Variations		
	Executed Each Cycle for ON Condition	++L(591)
	Executed Once for Upward Differentiation	@++L(591)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

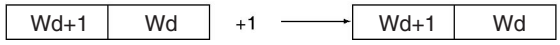
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	IR0 to IR15
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

The ++L(591) instruction adds 1 to the 8-digit hexadecimal content of Wd+1 and Wd. The content of the specified words will be incremented by 1 every cycle as long as the execution condition of ++L(591) is ON. When the up-differentiated variation of this instruction (@++L(591)) is used, the content of the

specified words is incremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 0000, the Carry Flag will be turned ON when a digit changes from F to 0, and the Negative Flag will be turned ON if bit 15 of Wd+1 is ON in the result.

Both the Equals Flag and the Carry Flag will be turned ON when the content of changes from FFFF FFFF to 0000 0000.

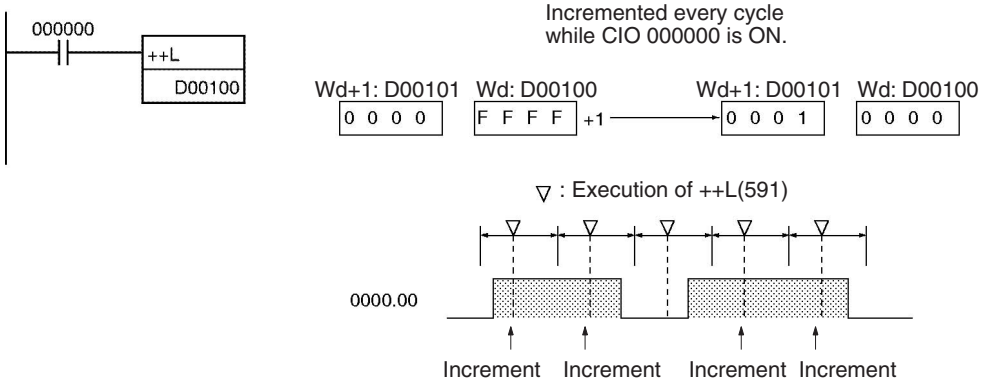
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd+1 or Wd went from F to 0 during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd+1 is ON after execution. OFF in all other cases.

Examples

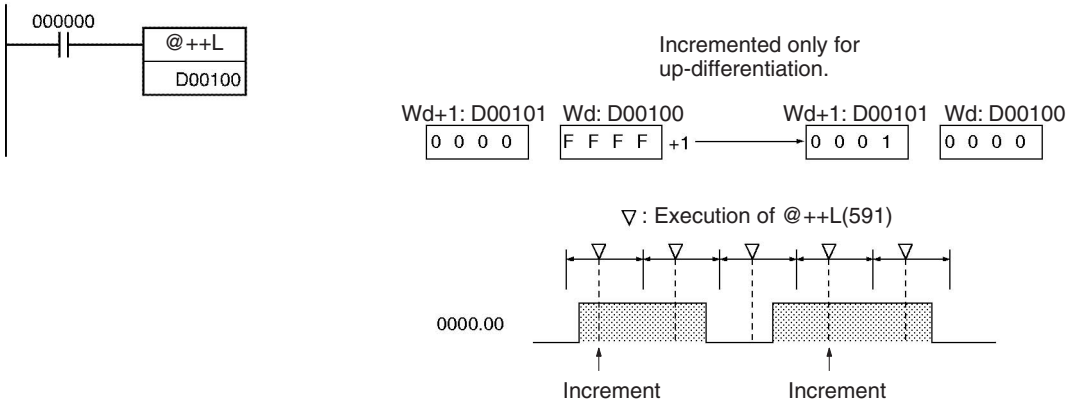
Operation of ++L(591)

In the following example, the 8-digit hexadecimal content of D00101 and D00100 will be incremented by 1 every cycle as long as CIO 000000 is ON.



Operation of @++L(591)

The up-differentiated variation is used in the following example, so the content of D00101 and D00100 will be incremented by 1 only when CIO 000000 has gone from OFF to ON.



3-10-3 DECREMENT BINARY: – –(592)

Purpose

Decrements the 4-digit hexadecimal content of the specified word by 1.

Ladder Symbol

Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	– – (592)
	Executed Once for Upward Differentiation	@– – (592)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

The – –(592) instruction subtracts 1 from the binary content of Wd. The specified word will be decremented by 1 every cycle as long as the execution condition of – –(592) is ON. When the up-differentiated variation of this instruction (@– –(592)) is used, the specified word is decremented only when the execution condition has gone from OFF to ON.

The Equals Flag will be turned ON if the result is 0000, the Carry Flag will be turned ON when a digit changes from 0 to F, and the Negative Flag will be turned ON if bit 15 of Wd is ON in the result.

Both the Carry Flag and the Negative Flag will be turned ON when the content of Wd changes from 0000 to FFFF.

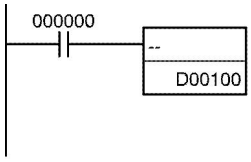
Flags

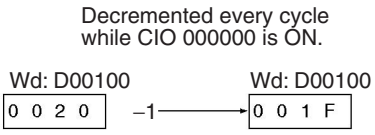
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the content of Wd is 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd went from 0 to F during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd is ON after execution. OFF in all other cases.

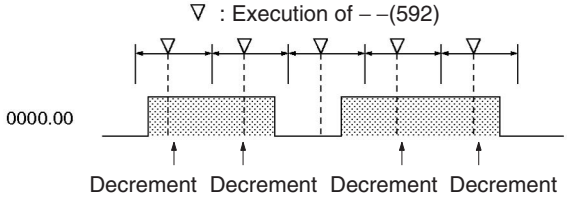
Examples

Operation of – –(592)

In the following example, the content of D00100 will be decremented by 1 every cycle as long as CIO 000000 is ON.

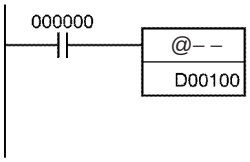


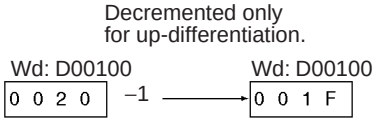


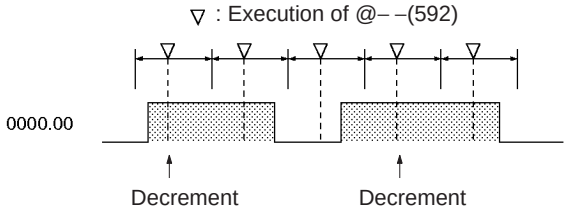


Operation of @– –(592)

The up-differentiated variation is used in the following example, so the content of D00100 will be decremented by 1 only when CIO 000000 has gone from OFF to ON.





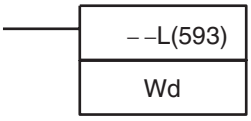


3-10-4 DOUBLE DECREMENT BINARY: – –L(593)

Purpose

Decrements the 8-digit hexadecimal content of the specified words by 1.

Ladder Symbol



Wd: First word

Variations

Variations		
	Executed Each Cycle for ON Condition	– –L(593)
	Executed Once for Upward Differentiation	@– –L(593)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

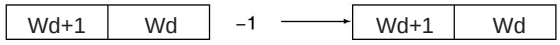
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	IR0 to IR15
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

The – –L(593) instruction subtracts 1 from the 8-digit hexadecimal content of Wd+1 and Wd. The content of the specified words will be decremented by 1 every cycle as long as the execution condition of – –L(593) is ON. When the up-differentiated variation of this instruction (@– –L(593)) is used, the content

of the specified words is decremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 0000, the Carry Flag will be turned ON when a digit changes from 0 to F, and the Negative Flag will be turned ON if bit 15 of Wd+1 is ON in the result.

Both the Carry Flag and the Negative Flag will be turned ON when the content changes from 0000 0000 to FFFF FFFF.

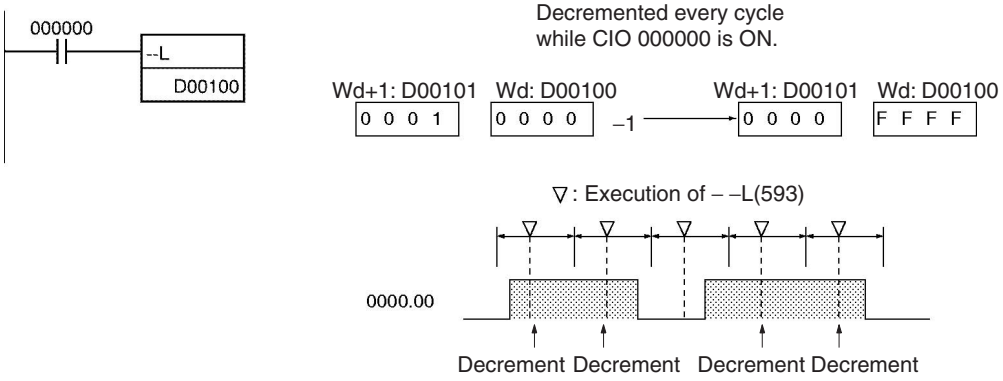
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd+1 or Wd went from 0 to F during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd+1 is ON after execution. OFF in all other cases.

Examples

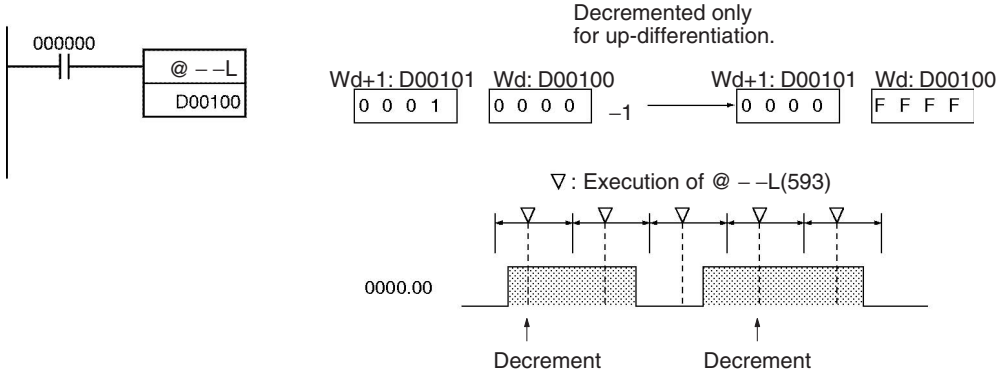
Operation of – –L(593)

In the following example, the 8-digit hexadecimal content of D00101 and D00100 will be decremented by 1 every cycle as long as CIO 000000 is ON.



Operation of @– –L(593)

The up-differentiated variation is used in the following example, so the content of D00101 and D00100 will be decremented by 1 only when CIO 000000 has gone from OFF to ON.



3-10-5 INCREMENT BCD: ++B(594)

Purpose

Increments the 4-digit BCD content of the specified word by 1.

Ladder Symbol

Wd: Word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	++B(594)
	Executed Once for Upward Differentiation	@++B(594)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

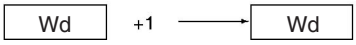
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n= 0 to C)
Indirect DM/EM addresses in BCD	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

The ++B(594) instruction adds 1 to the BCD content of Wd. The specified word will be incremented by 1 every cycle as long as the execution condition of ++B(594) is ON. When the up-differentiated variation of this instruction (@++B(594)) is used, the specified word is incremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 and the Carry Flag will be turned ON when a digit changes from 9 to 0.

Both the Equals Flag and the Carry Flag will be turned ON when the content of Wd changes from 9999 to 0000.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Wd is not BCD. OFF in all other cases.
Equals Flag	=	ON if the content of Wd is 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd went from 9 to 0 during execution. OFF in all other cases.

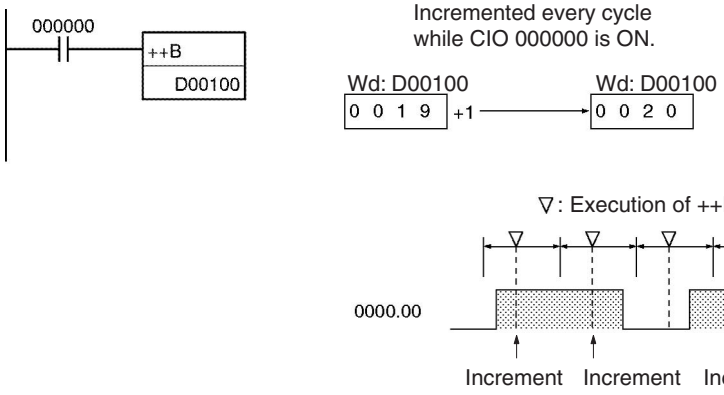
Precautions

The content of Wd must be BCD. If it is not BCD, an error will occur and the Error Flag will be turned ON.

Examples

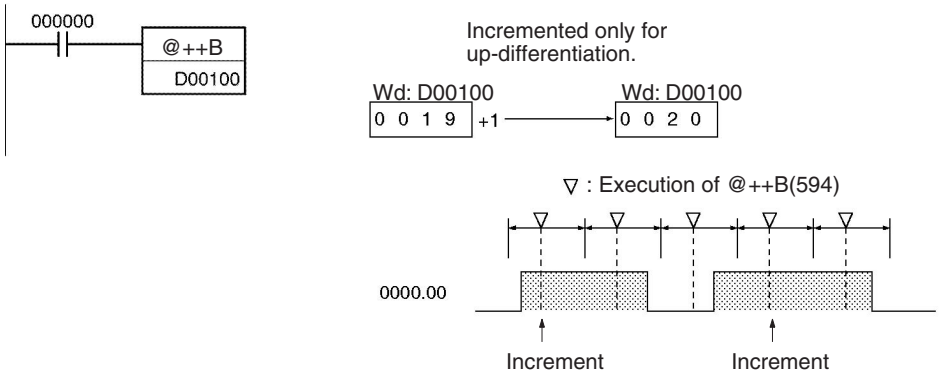
Operation of ++B(594)

In the following example, the BCD content of D00100 will be incremented by 1 every cycle as long as CIO 000000 is ON.



Operation of @++B(594)

The up-differentiated variation is used in the following example, so the content of D00100 will be incremented by 1 only when CIO 000000 has gone from OFF to ON.



3-10-6 DOUBLE INCREMENT BCD: ++BL(595)

Purpose

Increments the 8-digit BCD content of the specified words by 1.

Ladder Symbol

Wd: First word

Variations

Variations		
	Executed Each Cycle for ON Condition	++BL(595)
	Executed Once for Upward Differentiation	@++BL(595)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

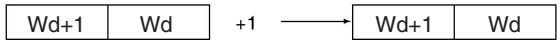
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in BCD	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

The ++BL(595) instruction adds 1 to the 8-digit BCD content of Wd+1 and Wd. The content of the specified words will be incremented by 1 every cycle as long as the execution condition of ++BL(595) is ON. When the up-differentiated variation of this instruction (@++BL(595)) is used, the content of the

specified words is incremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 0000 and the Carry Flag will be turned ON when a digit changes from 9 to 0.

Both the Equals Flag and the Carry Flag will be turned ON when the content of changes from 9999 9999 to 0000 0000.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Wd+1 and Wd is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd+1 or Wd went from 9 to 0 during execution. OFF in all other cases.

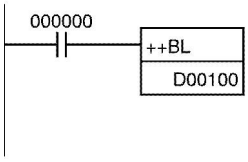
Precautions

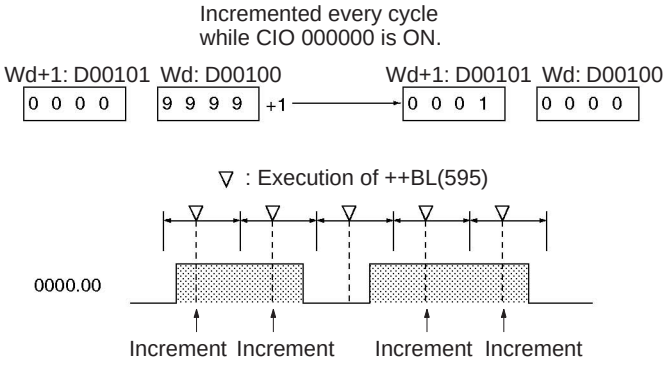
The content of Wd+1 and Wd must be BCD. If it is not BCD, an error will occur and the Error Flag will be turned ON.

Examples

Operation of ++BL(595)

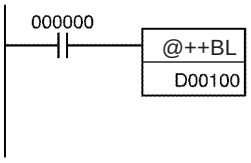
In the following example, the 8-digit BCD content of D00101 and D00100 will be incremented by 1 every cycle as long as CIO 000000 is ON.

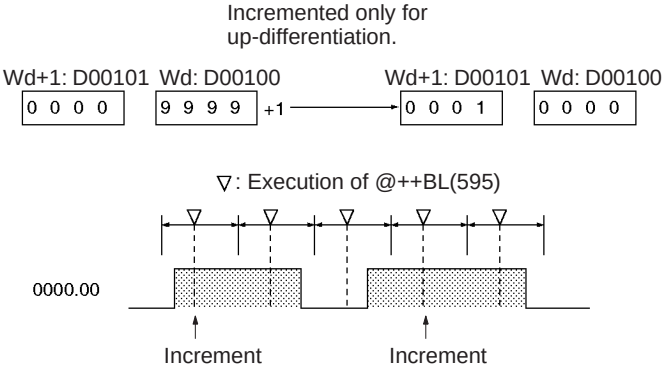




Operation of @++BL(595)

The up-differentiated variation is used in the following example, so the BCD content of D00101 and D00100 will be incremented by 1 only when CIO 000000 has gone from OFF to ON.



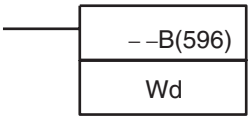


3-10-7 DECREMENT BCD: – –B(596)

Purpose

Decrements the 4-digit BCD content of the specified word by 1.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	– –B(596)
	Executed Once for Upward Differentiation	@– –B(596)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

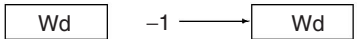
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

The – –B(596) instruction subtracts 1 from the BCD content of Wd. The specified word will be decremented by 1 every cycle as long as the execution condition of – –B(596) is ON. When the up-differentiated variation of this instruction (@– –B(596)) is used, the specified word is decremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 and the Carry Flag will be turned ON when a digit changes from 0 to 9.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Wd is not BCD. OFF in all other cases.
Equals Flag	=	ON if the content of Wd is 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd went from 0 to 9 during execution. OFF in all other cases.

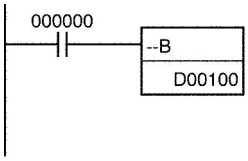
Precautions

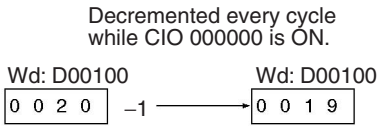
The content of Wd must be BCD. If it is not BCD, an error will occur and the Error Flag will be turned ON.

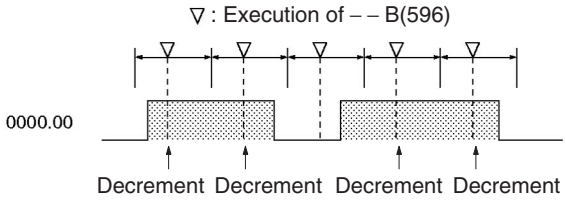
Examples

Operation of – –B(596)

In the following example, the BCD content of D00100 will be decremented by 1 every cycle as long as CIO 000000 is ON.

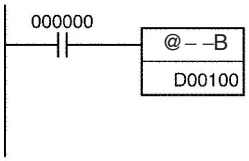


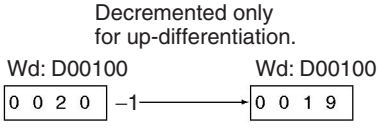


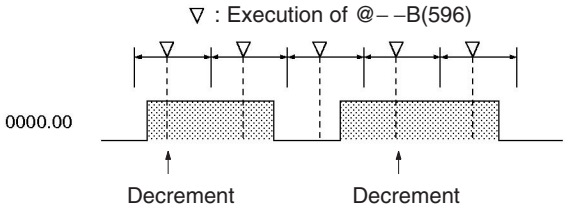


Operation of @– –B(596)

The up-differentiated variation is used in the following example, so the BCD content of D00100 will be decremented by 1 only when CIO 000000 has gone from OFF to ON.





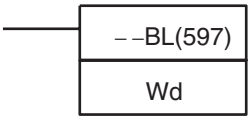


3-10-8 DOUBLE DECREMENT BCD: – –BL(597)

Purpose

Decrements the 8-digit BCD content of the specified words by 1.

Ladder Symbol



Wd: First word

Variations

Variations		
	Executed Each Cycle for ON Condition	– –BL(597)
	Executed Once for Upward Differentiation	@– –BL(597)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

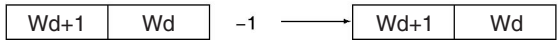
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in BCD	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

The – –BL(597) instruction subtracts 1 from the 8-digit BCD content of Wd+1 and Wd. The content of the specified words will be decremented by 1 every cycle as long as the execution condition of – –BL(597) is ON. When the up-differentiated variation of this instruction (@– –BL(597)) is used, the content

of the specified words is decremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 0000 and the Carry Flag will be turned ON when a digit changes from 0 to 9.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Wd+1 and Wd is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if a digit in Wd+1 or Wd went from 0 to 9 during execution. OFF in all other cases.

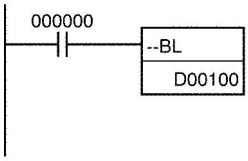
Precautions

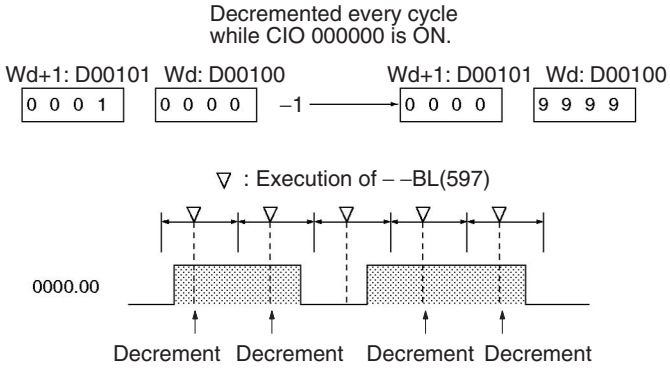
The content of Wd+1 and Wd must be BCD. If it is not BCD, an error will occur and the Error Flag will be turned ON.

Examples

Operation of – –BL(597)

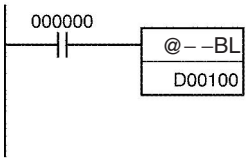
In the following example, the 8-digit BCD content of D00101 and D00100 will be decremented by 1 every cycle as long as CIO 000000 is ON.

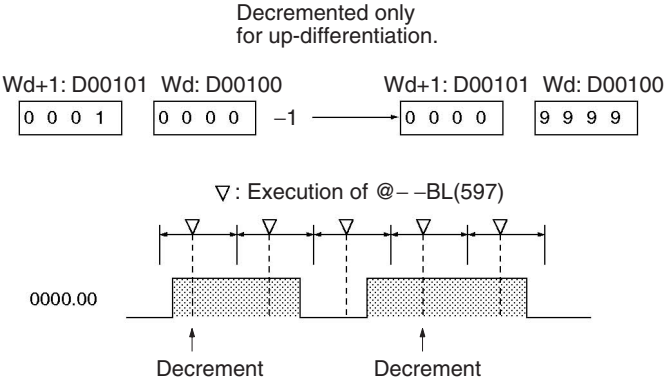




Operation of @– –BL(597)

The up-differentiated variation is used in the following example, so the BCD content of D00101 and D00100 will be decremented by 1 only when CIO 000000 has gone from OFF to ON.





3-11 Symbol Math Instructions

This section describes the Symbol Math Instructions, which perform arithmetic operations on BCD or binary data.

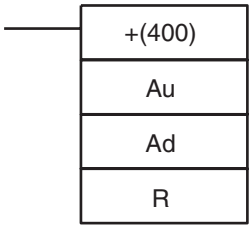
Instruction	Mnemonic	Function code	Page
SIGNED BINARY ADD WITHOUT CARRY	+	400	426
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	401	428
SIGNED BINARY ADD WITH CARRY	+C	402	430
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	403	432
BCD ADD WITHOUT CARRY	+B	404	434
DOUBLE BCD ADD WITHOUT CARRY	+BL	405	435
BCD ADD WITH CARRY	+BC	406	437
DOUBLE BCD ADD WITH CARRY	+BCL	407	439
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	410	440
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	411	442
SIGNED BINARY SUBTRACT WITH CARRY	–C	412	446
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	413	448
BCD SUBTRACT WITHOUT CARRY	–B	414	451
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	415	452
BCD SUBTRACT WITH CARRY	–BC	416	456
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	417	457
SIGNED BINARY MULTIPLY	*	420	459
DOUBLE SIGNED BINARY MULTIPLY	*L	421	461
UNSIGNED BINARY MULTIPLY	*U	422	463
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	423	465
BCD MULTIPLY	*B	424	467
DOUBLE BCD MULTIPLY	*BL	425	469
SIGNED BINARY DIVIDE	/	430	471
DOUBLE SIGNED BINARY DIVIDE	/L	431	473
UNSIGNED BINARY DIVIDE	/U	432	475
DOUBLE UNSIGNED BINARY DIVIDE	/UL	433	477
BCD DIVIDE	/B	434	479
DOUBLE BCD DIVIDE	/BL	435	481

3-11-1 SIGNED BINARY ADD WITHOUT CARRY: +(400)

Purpose

Adds 4-digit (single-word) hexadecimal data and/or constants.

Ladder Symbol



Au: Augend word

Ad: Addend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+(400)
	Executed Once for Upward Differentiation	@+(400)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

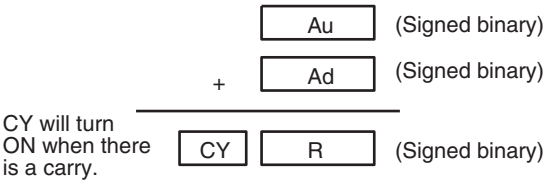
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+(400) adds the binary values in Au and Ad and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.
Overflow Flag	OF	ON when the result of adding two positive numbers is in the range 8000 to FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of adding two negative numbers is in the range 0000 to 7FFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When +(400) is executed, the Error Flag will turn OFF.

If as a result of the addition, the content of R is 0000 hex, the Equals Flag will turn ON.

If the addition results in a carry, the Carry Flag will turn ON.

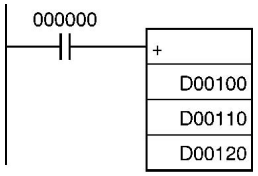
If the result of adding two positive numbers is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

If the result of adding two negative numbers is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

If as a result of the addition, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00100 and D00110 will be added as 4-digit signed binary values and the result will be output to D00120.

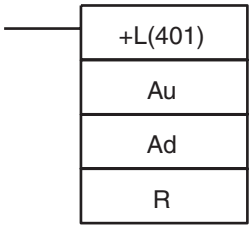


3-11-2 DOUBLE SIGNED BINARY ADD WITHOUT CARRY: +L(401)

Purpose

Adds 8-digit (double-word) hexadecimal data and/or constants.

Ladder Symbol



Au: 1st augend word

Ad: 1st addend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+L(401)
	Executed Once for Upward Differentiation	@+L(401)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

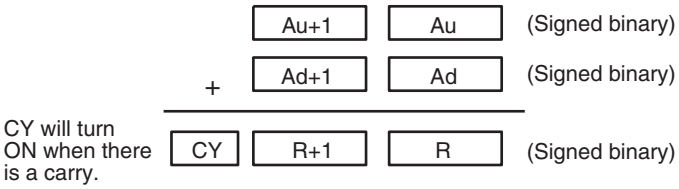
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	IR0 to IR15		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+L(401) adds the binary values in Au and Au+1 and Ad and Ad+1 and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.
Overflow Flag	OF	ON when the result of adding two positive numbers is in the range 80000000 to FFFFFFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of adding two negative numbers is in the range 00000000 to 7FFFFFFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When +L(401) is executed, the Error Flag will turn OFF.

If as a result of the addition, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If the addition results in a carry, the Carry Flag will turn ON.

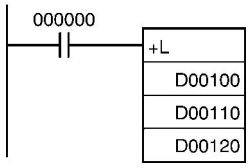
If the result of adding two positive numbers is negative (in the range 80000000 to FFFFFFFF hex), the Overflow Flag will turn ON.

If the result of adding two negative numbers is positive (in the range 00000000 to 7FFFFFFF hex), the Underflow Flag will turn ON.

If as a result of the addition, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, D00100 and D00110 and D00111 and D00110 will be added as 8-digit signed binary values and the result will be output to D00120 and D00120.

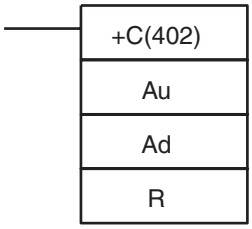


3-11-3 SIGNED BINARY ADD WITH CARRY: +C(402)

Purpose

Adds 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).

Ladder Symbol



Au: Augend word

Ad: Addend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+C(402)
	Executed Once for Upward Differentiation	@+C(402)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

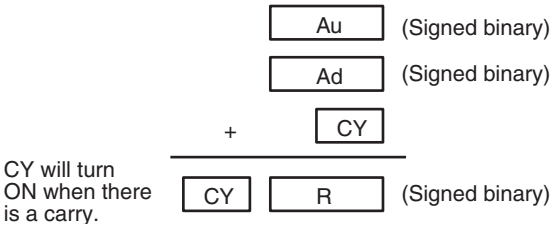
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+C(402) adds the binary values in Au, Ad, and CY and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the addition result is 0. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.
Overflow Flag	OF	ON when the addition result of adding two positive numbers and CY is in the range 8000 to FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the addition result of adding two negative numbers and CY is in the range 0000 to 7FFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When +C(402) is executed, the Error Flag will turn OFF.

If as a result of the addition, the content of R is 0000 hex, the Equals Flag will turn ON.

If the addition results in a carry, the Carry Flag will turn ON.

If the result of adding two positive numbers and CY is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

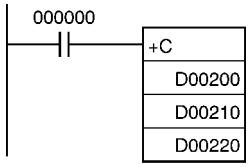
If the result of adding two negative numbers and CY is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

If as a result of the addition, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 000000 is ON, D00100, D00110, and CY will be added as 4-digit signed binary values and the result will be output to D00220.

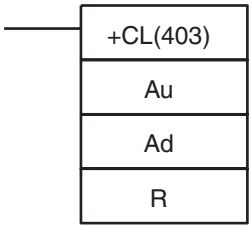


3-11-4 DOUBLE SIGNED BINARY ADD WITH CARRY: +CL(403)

Purpose

Adds 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).

Ladder Symbol



Au: 1st augend word

Ad: 1st addend word

R: 1st result word

Variations

Variations	**Executed Each Cycle for ON Condition**	+CL(403)
	Executed Once for Upward Differentiation	@+CL(403)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

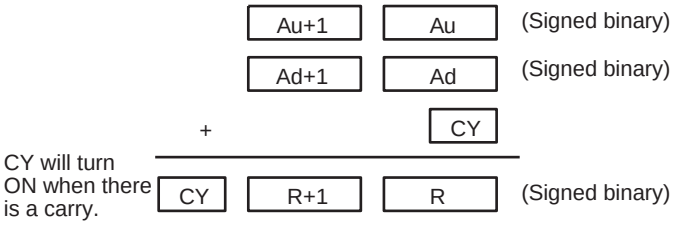
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+CL(403) adds the binary values in Au and Au+1, Ad and Ad+1, and CY and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the results in a carry. OFF in all other cases.
Overflow Flag	OF	ON when the result of adding two positive numbers and CY is in the range 80000000 to FFFFFFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of adding two negative numbers and CY is in the range 00000000 to 7FFFFFFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When +CL(403) is executed, the Error Flag will turn OFF.

If as a result of the addition, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If the addition results in a carry, the Carry Flag will turn ON.

If the result of adding two positive numbers and CY is negative (in the range 80000000 to FFFFFFFF hex), the Overflow Flag will turn ON.

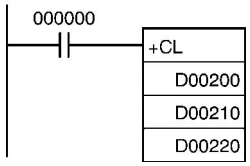
If the result of adding two negative numbers and CY is positive (in the range 00000000 to 7FFFFFFF hex), the Underflow Flag will turn ON.

If as a result of the addition, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 000000 is ON, D00201, D00200, D00211, D00210, and CY will be added as 8-digit signed binary values, and the result will be output to D00221 and D00220.

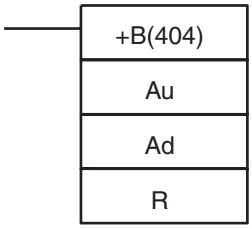


3-11-5 BCD ADD WITHOUT CARRY: +B(404)

Purpose

Adds 4-digit (single-word) BCD data and/or constants.

Ladder Symbol



Au: Augend word

Ad: Addend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+B(404)
	Executed Once for Upward Differentiation	@+B(404)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

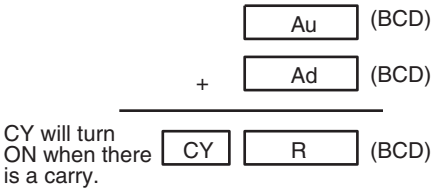
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	0000 to 9999 (BCD)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+B(404) adds the BCD values in Au and Ad and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	ON when Au is not BCD. ON when Ad is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

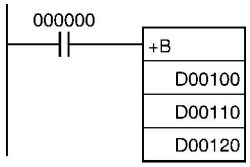
If Au or Ad is not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the addition, the content of R is 0000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00100 and D00110 will be added as 4-digit BCD values, and the result will be output to D00120.

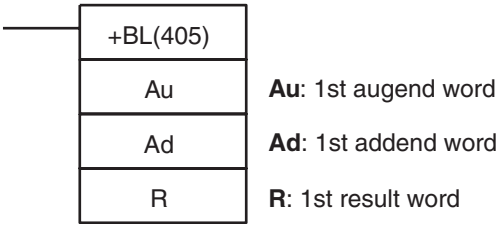


3-11-6 DOUBLE BCD ADD WITHOUT CARRY: +BL(405)

Purpose

Adds 8-digit (double-word) BCD data and/or constants.

Ladder Symbol



Au: 1st augend word

Ad: 1st addend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+BL(405)
	Executed Once for Upward Differentiation	@+BL(405)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

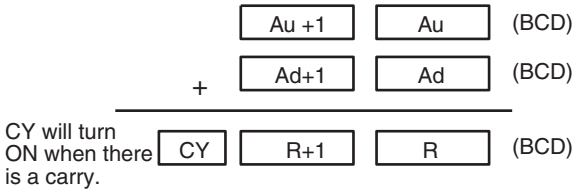
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #99999999 (BCD)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+BL(405) adds the BCD values in Au and Au+1 and Ad and Ad+1 and outputs the result to R, R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when Au, Au +1 is not BCD. ON when Ad, Ad +1 is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

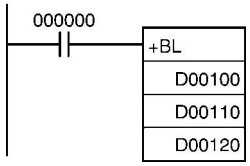
If Au, Au +1 or Ad, Ad +1 are not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the addition, the content of R, R +1 is 00000000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00101 and D00100 and D00111 and D00110 will be added as 8-digit BCD values, and the result will be output to D00121 and D00120.

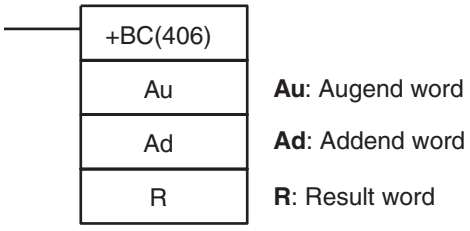


3-11-7 BCD ADD WITH CARRY: +BC(406)

Purpose

Adds 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	+BC(406)
	Executed Once for Upward Differentiation	@+BC(406)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

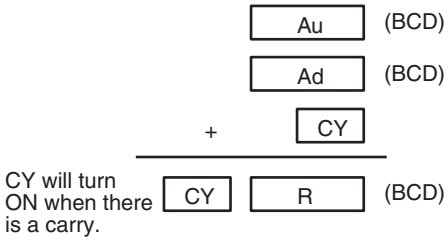
Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		

Area	Au	Ad	R
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to 9999 (BCD)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+BC(406) adds BCD values in Au, Ad, and CY and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	ON when Au is not BCD. ON when Ad is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

If Au or Ad is not BCD, an error is generated and the Error Flag will turn ON.

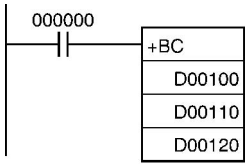
If as a result of the addition, the content of R is 0000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Note To clear the Carry Flay (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 000000 is ON in the following example, D00100, D00110, and CY will be added as 4-digit BCD values, and the result will be output to D00120.

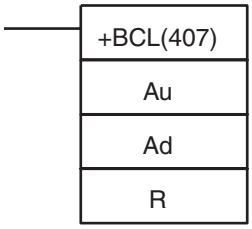


3-11-8 DOUBLE BCD ADD WITH CARRY: +BCL(407)

Purpose

Adds 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY).

Ladder Symbol



Au: 1st augend word

Ad: 1st addend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+BCL(407)
	Executed Once for Upward Differentiation	@+BCL(407)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

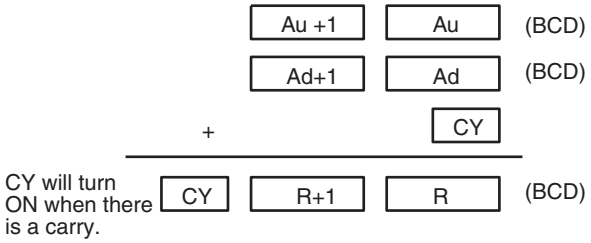
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #99999999 (BCD)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+BCL(407) adds the BCD values in Au and Au+1, Ad and Ad+1, and CY and outputs the result to R, R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when Au, Au +1 is not BCD. ON when Ad, Ad +1 is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

If Au, Au +1 or Ad, Ad +1 are not BCD, an error is generated and the Error Flag will turn ON.

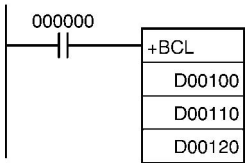
If as a result of the addition, the content of R, R +1 is 00000000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Note To clear the Carry Flay (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 000000 is ON in the following example, D00101, D00100, D00111, D00110, and CY will be added as 8-digit BCD values, and the result will be output to D00121 and D00120.

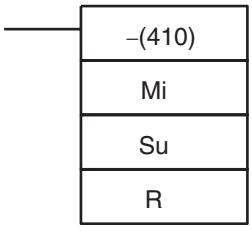


3-11-9 SIGNED BINARY SUBTRACT WITHOUT CARRY: –(410)

Purpose

Subtracts 4-digit (single-word) hexadecimal data and/or constants.

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–(410)
	Executed Once for Upward Differentiation	@–(410)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

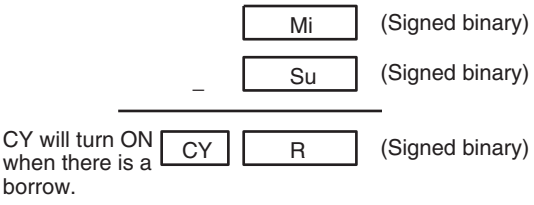
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D0000 to D4095		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–(400) subtracts the binary values in Su from Mi and outputs the result to R. When the result is negative, it is output to R as a 2's complement. (Refer to *3-11-10 DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY: –L(411)* for an example of handling 2's complements.)



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.

Name	Label	Operation
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.
Overflow Flag	OF	ON when the result of subtracting a negative number from a positive number is in the range 8000 to FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of subtracting a negative number from a positive number is in the range 0000 to 7FFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When –(410) is executed, the Error Flag will turn OFF.

If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If the subtraction results in a borrow, the Carry Flag will turn ON.

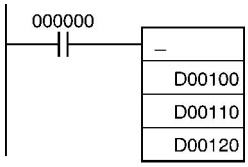
If the result of subtracting a negative number from a positive number is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

If the result of subtracting a positive number from a negative number is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

If as a result of the subtraction, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00110 will be subtracted from D00100 as 4-digit signed binary values and the result will be output to D00120.

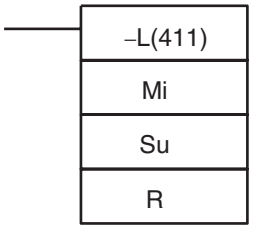


3-11-10 DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY: –L(411)

Purpose

Subtracts 8-digit (double-word) hexadecimal data and/or constants.

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–L(411)
	Executed Once for Upward Differentiation	@–L(411)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

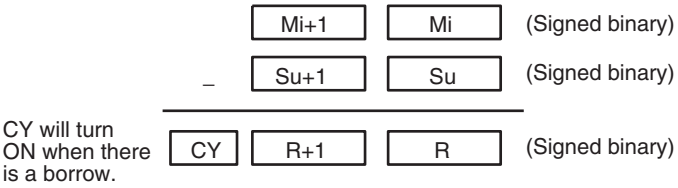
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	IR0 to IR15		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–L(411) subtracts the binary values in Su and Su+1 from Mi and Mi+1 and outputs the result to R, R+1. When the result is negative, it is output to R and R+1 as a 2's complement.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.
Overflow Flag	OF	ON when the result of subtracting a negative number from a positive number is in the range 80000000 to FFFFFFFF hex. OFF in all other cases.

Name	Label	Operation
Underflow Flag	UF	ON when the result of subtracting a positive number from a negative number is in the range 00000000 to 7FFFFFFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When –L(411) is executed, the Error Flag will turn OFF.

If as a result of the subtraction, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If the subtraction results in a borrow, the Carry Flag will turn ON.

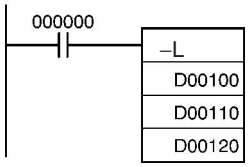
If the result of subtracting a negative number from a positive number is negative (in the range 80000000 to FFFFFFFF hex), the Overflow Flag will turn ON.

If the result of subtracting a positive number from a negative number is positive (in the range 00000000 to 7FFFFFFF hex), the Underflow Flag will turn ON.

If as a result of the subtraction, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00111 and D00110 will be subtracted from D00101 and D00100 as 8-digit signed binary values and the result will be output to D00121 and D00120.



Examples

If the result of the subtraction is a negative number (Mi<Su or Mi+1, Mi <Su+1, Su), the result is output as the 2's complement and the Carry Flag (CY) will turn ON to indicate that the result of the subtraction is negative. To convert the 2's complement to the true number, an instruction which subtracts the result from 0 is necessary using the Carry Flag (CY) as an execution condition.

Note 2's Complement

A 2's complement is the value obtained by subtracting each binary digit from 1 and adding one to the result. For example, the 2's complement for 1101 is calculated as follows: 1111 (F hexadecimal) – 1101 (D hexadecimal) + 1 (1 hexadecimal) = 0011 (3 hexadecimal). The 2's complement for 3039 (hexadecimal) is calculated as follows: FFFF (hexadecimal) – 3039 (hexadecimal) + 0001 (hexadecimal) – CFC7 (hexadecimal). Therefore, in case of 4-digit hexadecimal value, the 2's complement can be calculated as follows: FFFF (hexadecimal) – a (hexadecimal) + 0001 (hexadecimal) = b (hexadecimal). To obtain the true number from the 2's complement b (hexadecimal): a (hexadecimal) = 10000 (hexadecimal) – b (hexadecimal). For example, to obtain the true number from the 2's complement CFC7 (hexadecimal): 10000 (hexadecimal) – CFC7 = 3039.

Example 1

	Signed data	Unsigned data
FFFF Hex →	–1	65535
–) 0001 Hex →	–) +1	–) 1
FFFE Hex →	–2 Note 1	65534 Note 2
Negative Flag ON		
Carry Flag OFF		

Example 2

	Signed data	Unsigned data
FFFD Hex →	–3	65533
–)FFFF Hex →	–) –1	–) 65535
FFFE Hex →	–2 Note 3	65534 Note 4
Negative Flag ON		
Carry Flag OFF		

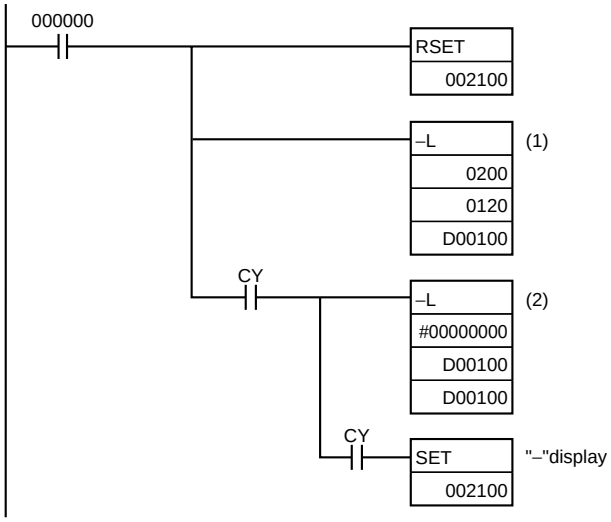
Note

1. Since the Negative Flag is ON, the result (FFFE hex) is a negative value (2's complement) and is thus –2.
2. Since the Carry Flag is OFF, the result (FFFE hex) is an unsigned positive value of 65534.
3. Since the Negative Flag is ON, the result (FFFE hex) is a negative value (2's complement) and is thus –2.
4. Since the Carry Flag is ON, the result (FFFE hex) is a negative value (2's complement) and becomes –2 when converted to a true value.

Program Example

20F55A10 – B8A360E3 = –97AE06D3.

In this example, the eight-digit binary value in CIO 0121 and CIO 0120 is subtracted from the value in CIO 0201 and CIO 0200, and the result is output in eight-digit binary to D00101 and D00100. If the result is negative, the instruction at (2) will be executed, and the actual result will then be output to D00101 and D00100.

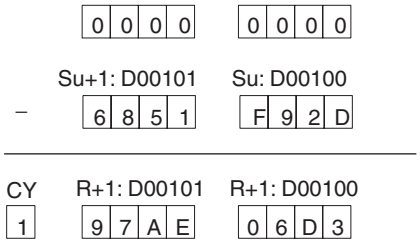


Subtraction at 1

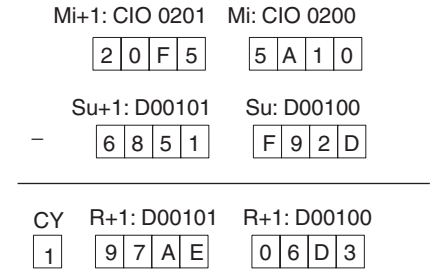
	Mi+1: CIO 0201				Mi: CIO 0200			
	2	0	F	5	5	A	1	0
	Su+1: CIO 0121				Su: CIO 0120			
–	B	8	A	3	6	0	E	3
CY	R+1: D00101				R+1: D00100			
1	6	8	5	1	F	9	2	D

The Carry Flag (CY) is ON, so the result is subtracted from 0000 0000 to obtain the actual number.

Subtraction at 2



Final Subtraction Result



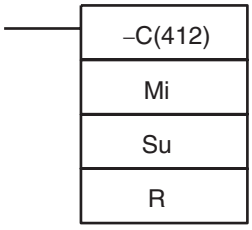
The Carry Flag (CY) is turned ON, so the actual number is –97AE06D3. Because the content of D00101 and D00100 is negative, CY is used to turn ON CIO 002100 to indicate this.

3-11-11 SIGNED BINARY SUBTRACT WITH CARRY: –C(412)

Purpose

Subtracts 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–C(412)
	Executed Once for Upward Differentiation	@–C(412)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

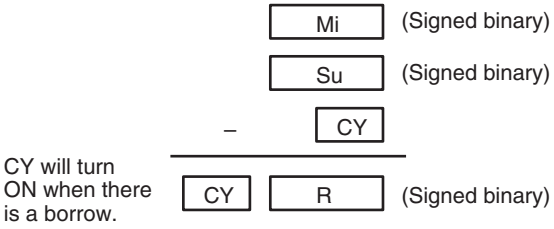
Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959

Area	Mi	Su	R
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–C(412) subtracts the binary values in Su and CY from Mi, and outputs the result to R. When the result is negative, it is output to R as a 2's complement.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the subtraction result is 0. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.
Overflow Flag	OF	ON when the result of subtracting a negative number and CY from a positive number is in the range 8000 to FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of subtracting a positive number and CY from a negative number is in the range 0000 to 7FFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When –C(412) is executed, the Error Flag will turn OFF.

If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If the subtraction results in a borrow, the Carry Flag will turn ON.

If the result of subtracting a negative number and CY from a positive number is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

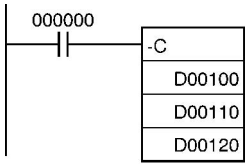
If the result of subtracting a positive number and CY from a negative number is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

If as a result of the subtraction, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 000000 is ON in the following example, D00110 and CY will be subtracted from D00100 as 4-digit signed binary values and the result will be output to D00120.

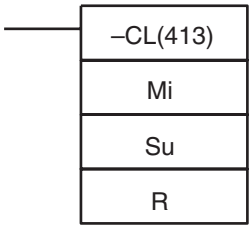


3-11-12 DOUBLE SIGNED BINARY SUBTRACT WITH CARRY: –CL(413)

Purpose

Subtracts 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–CL(413)
	Executed Once for Upward Differentiation	@–CL(413)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

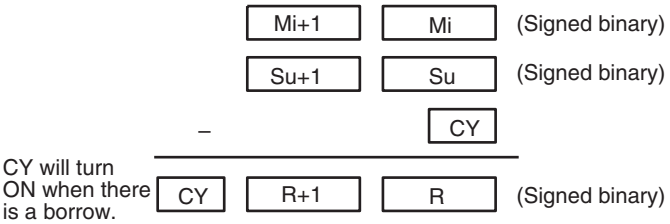
Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		

Area	Mi	Su	R
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–CL(413) subtracts the binary values in Su and Su+1 and CY from Mi and Mi+1, and outputs the result to R, R+1. When the result is negative, it is output to R, R+1 as a 2's complement.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the results in a borrow. OFF in all other cases.
Overflow Flag	OF	ON when the result of subtracting a negative number and CY from a positive number is in the range 80000000 to FFFFFFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of subtracting a positive number and CY from a negative number is in the range 00000000 to 7FFFFFFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When –CL(413) is executed, the Error Flag will turn OFF.

If as a result of the subtraction, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If the subtraction results in a borrow, the Carry Flag will turn ON.

If the result of subtracting a negative number and CY from a positive number is negative (in the range 80000000 to FFFFFFFF hex), the Overflow Flag will turn ON.

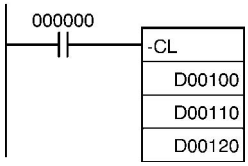
If the result of subtracting a positive number and CY from a negative number is positive (in the range 00000000 to 7FFFFFFF hex), the Underflow Flag will turn ON.

If as a result of the subtraction, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 000000 is ON in the following example, D00111, D00110 and CY will be subtracted from D00101 and D00100 as 8-digit signed binary values, and the result will be output to D00121 and D00120.



If the result of the subtraction is a negative number (Mi<Su or Mi+1, Mi <Su+1, Su), the result is output as a 2's complement. The Carry Flag (CY) will turn ON. To convert the 2's complement to the true number, a program which subtracts the result from 0 is necessary, as an input condition of the Carry Flag (CY). The Carry Flag turning ON thus indicates that the result of the subtraction is negative.

Note 2's Complement

A 2's complement is the value obtained by subtracting each binary digit from 1 and adding one to the result.

Example: The 2's complement for the binary number 1101 is as follows:

1111 (F hex) – 1101 (D hex) + 1 (1 hex) = 0011 (3 hex).

Example: The 2's complement for the 4-digit hexadecimal number 3039 is as follows:

FFFF hex – 3039 hex + 0001 hex = CFC7 hex.

Accordingly, the 2's complement for the 4-digit hexadecimal value "a" is as follows:

FFFF hex – a hex + 0001 hex = b hex.

And to obtain the true number "a" hex from the 2's complement "b" hex:

a hex + 10000 hex – b hex.

Example: To obtain the true number from the 2's complement CFC& hex:

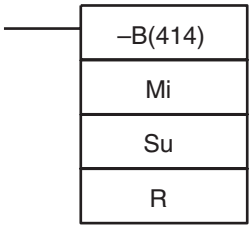
10000 hex – CFC7 hex = 3039 hex.

3-11-13 BCD SUBTRACT WITHOUT CARRY: –B(414)

Purpose

Subtracts 4-digit (single-word) BCD data and/or constants.

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	–B(414)
	Executed Once for Upward Differentiation	@–B(414)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

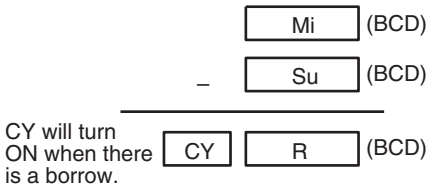
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	0000 to 9999 (BCD)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–B(414) subtracts the BCD values in Su from Mi and outputs the result to R. If the result of the subtraction is negative, the result is output as a 10's complement.



Flags

Name	Label	Operation
Error Flag	ER	ON when Mi is not BCD. ON when Su is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

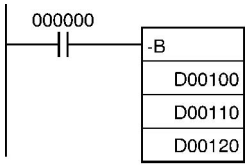
If Mi and/or Su are not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If an addition results in a borrow, the Carry Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00110 is subtracted from D00100 as 4-digit BCD values, and the result will be output to D00120.

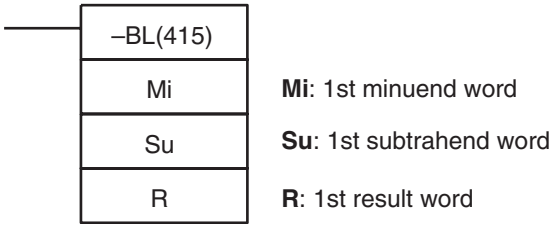


3-11-14 DOUBLE BCD SUBTRACT WITHOUT CARRY: –BL(415)

Purpose

Subtracts 8-digit (double-word) BCD data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	–BL(415)
	Executed Once for Upward Differentiation	@–BL(415)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

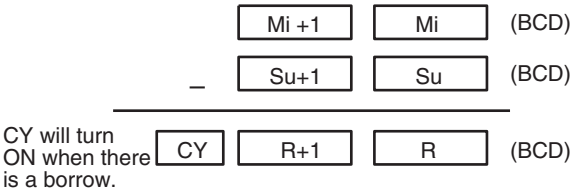
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #99999999 (BCD)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–BL(415) subtracts the BCD values in Su and Su+1 from Mi and Mi+1 and outputs the result to R, R+1. If the result is negative, it is output to R, R+1 as a 10's complement.



Flags

Name	Label	Operation
Error Flag	ER	ON when Mi and/or Mi +1 are not BCD. ON when Su and/or Su +1 are not BCD. OFF in all other cases.

Name	Label	Operation
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

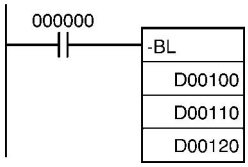
If Mi, Mi +1 and/or Su, Su +1 are not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the subtraction, the content of R, R +1 is 00000000 hex, the Equals Flag will turn ON.

If an addition results in a borrow, the Carry Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00111 and D00110 will be subtracted from D00101 and D00100 as 8-digit BCD values, and the result will be output to D00121 and D00120.



If the result of the subtraction is a negative number (Mi<Su or Mi+1, Mi <Su+1, Su), the result is output as a 10's complement. The Carry Flag (CY) will turn ON. To convert the 10's complement to the true number, a program which subtracts the result from 0 is necessary, as an input condition of the Carry Flag (CY). The Carry Flag turning ON thus indicates that the result of the subtraction is negative.

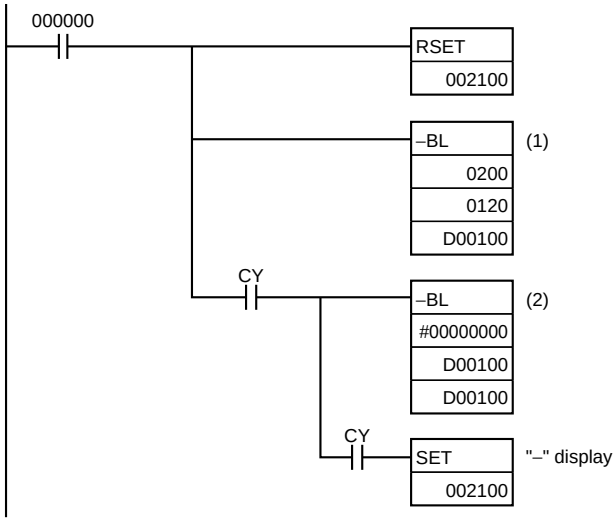
Note 10's Complement

A 10's complement is the value obtained by subtracting each digit from 9 and adding one to the result. For example, the 10's complement for 7556 is calculated as follows: 9999 – 7556 + 1 = 2444. For a four digit number, the 10's complement of A is 9999 – A + 1 = B. To obtain the true number from the 10's complement B: A = 10000 – B. For example, to obtain the true number from the 10's complement 2444: 10000 – 2444 = 7556.

Program Example

9,583,960 – 17,072,641 = –7,488,681.

In this example, the eight-digit BCD content of CIO 0121 and CIO 0120 is subtracted from the content of CIO 0201 and CIO 0200, and the result is output in eight-digit BCD to D00101 and D00100. The result is negative, so the instruction at (2) will be executed, and the true value will then be output to D00101 and D00100.



Subtraction at 1

Mi+1: CIO 0201 Mi: CIO 0200

0	9	5	8

3	9	6	0

Su+1: CIO 0121 Su: CIO 0120

– | 1 | 7 | 0 | 7 |

2	6	4	1

09583960 + (100000000 – 17072641)

CY R+1: D00101 R+1: D00100

1

9	2	5	1

1	3	1	9

The Carry Flag (CY) is ON, so the result is subtracted from 0000 0000.

Subtraction at 2

0	0	0	0

0	0	0	0

Su+1: D00101 Su: D00100

– | 9 | 2 | 5 | 1 |

1	3	1	9

00000000 + (100000000 – 92511319)

CY R+1: D00101 R+1: D00100

1

0	7	4	8

8	6	8	1

Final Subtraction Result

Mi+1: CIO 0201 Mi: CIO 0200

2	0	F	5

5	A	1	0

Su+1: D00101 Su: D00100

– | 6 | 8 | 5 | 1 |

F	9	2	D

CY R+1: D00101 R+1: D00100

1

0	7	4	8

8	6	8	1

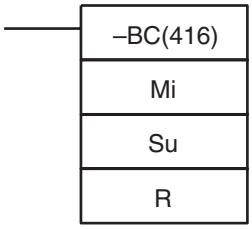
The Carry Flag (CY) will be turned ON, so the actual number is –7,488,681. Because the content of D00101 and D00100 is negative, CY is used to turn ON CIO 002100 to indicate this.

3-11-15 BCD SUBTRACT WITH CARRY: –BC(416)

Purpose

Subtracts 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	–BC(416)
	Executed Once for Upward Differentiation	@–BC(416)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

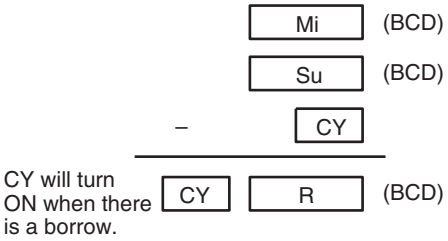
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to D32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #9999 (BCD)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–BC(416) subtracts BCD values in Su and CY from Mi and outputs the result to R. If the result is negative, it is output to R as a 2's complement.



Flags

Name	Label	Operation
Error Flag	ER	ON when Mi is not BCD. ON when Su is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

If Mi and/or Su are not BCD, an error is generated and the Error Flag will turn ON.

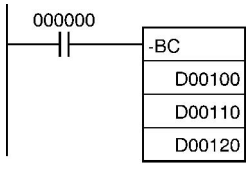
If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If an addition results in a borrow, the Carry Flag will turn ON.

Note To clear the Carry Flay (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 000000 is ON in the following example, D00110 and CY will be subtracted from D00100 as 4-digit BCD values, and the result will be output to D00120.

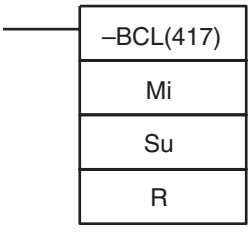


3-11-16 DOUBLE BCD SUBTRACT WITH CARRY: –BCL(417)

Purpose

Subtracts 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY).

Ladder Symbol



Mi: 1st minuend word

Su: 1st subtrahend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–BCL(417)
	Executed Once for Upward Differentiation	@–BCL(417)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

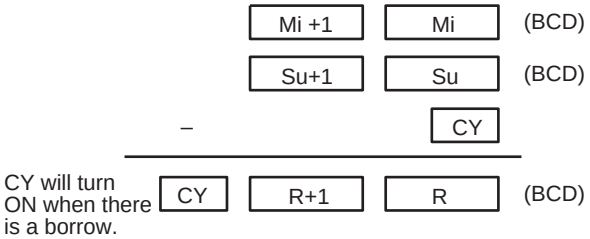
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #99999999 (BCD)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–BCL(417)subtracts the BCD values in Su, Su+1, and CY from Mi and Mi+1 and outputs the result to R, R+1. If the result is negative, it is output to R, R+1 as a 10's complement.



Flags

Name	Label	Operation
Error Flag	ER	ON when Mi and/or Mi +1 are not BCD. ON when Su and/or Su +1 are not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

If Mi, Mi +1 and/or Su, Su +1 are not BCD, an error is generated and the Error Flag will turn ON.

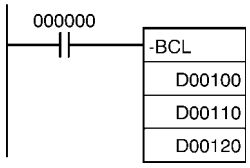
If as a result of the subtraction, the content of R, R +1 is 00000000 hex, the Equals Flag will turn ON.

If an subtraction results in a borrow, the Carry Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 000000 is ON in the following example, D00111, D00110, and CY will be subtracted from D00101 and D00100 as 8-digit BCD values, and the result will be output to D00121 and D00120.



If the result of the subtraction is a negative number (Mi<Su or Mi+1, Mi <Su+1, Su), the result is output as a 10's complement. The Carry Flag (CY) will turn ON. To convert the 10's complement to the true number, a program which subtracts the result from 0 is necessary, as an input condition of the Carry Flag (CY). The Carry Flag turning ON thus indicates that the result of the subtraction is negative.

Note 10's Complement

A 10's complement is the value obtained by subtracting each digit from 9 and adding one to the result. For example, the 10's complement for 7556 is calculated as follows: 9999 – 7556 + 1 = 2444. For a four digit number, the 10's complement of A is 9999 – A + 1 = B. To obtain the true number from the 10's complement B: A = 10000 – B. For example, to obtain the true number from the 10's complement 2444: 10000 – 2444 = 7556.

3-11-17 SIGNED BINARY MULTIPLY: *(420)

Purpose

Multiplies 4-digit signed hexadecimal data and/or constants.

Ladder Symbol

Md: Multiplicand word

Mr: Multiplier word

R: Result word

Variations

Variations	Executed Each Cycle for ON Condition	*(420)
	Executed Once for Upward Differentiation	@*(420)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

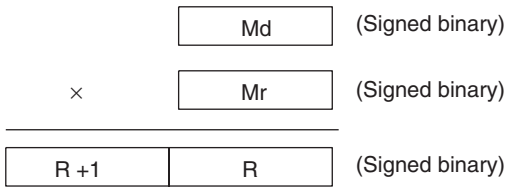
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W511		W000 to W510
Holding Bit Area	H000 to H511		H000 to H510
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D00000 to D32767		D00000 to D32766
EM Area without bank	E00000 to E32767		E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*(420) multiplies the signed binary values in Md and Mr and outputs the result to R, R+1.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

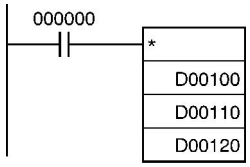
When *(420) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the multiplication, the content of the leftmost bit of R+1 and R is 1, the Negative Flag will turn ON.

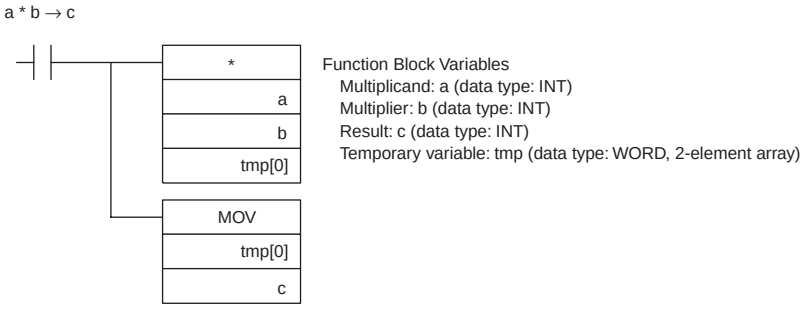
Examples

When CIO 000000 is ON in the following example, D00100 and D00110 will be multiplied as 4-digit signed hexadecimal values and the result will be output to D00120.



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.

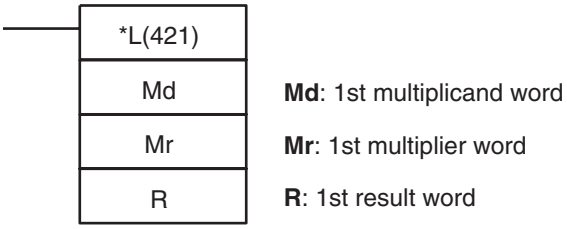


3-11-18 DOUBLE SIGNED BINARY MULTIPLY: *L(421)

Purpose

Multiplies 8-digit signed hexadecimal data and/or constants.

Ladder Symbol



Md: 1st multiplicand word

Mr: 1st multiplier word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*L(421)
	Executed Once for Upward Differentiation	@*L(421)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W510		W000 to W508
Holding Bit Area	H000 to H510		H000 to H508
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D00000 to D32766		D00000 to D32764
EM Area without bank	E00000 to E32766		E00000 to E32764
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		En_00000 to En_32764 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*L(421) multiplies the signed binary values in Md and Md+1 and Mr and Mr+1 and outputs the result to R, R+1, R+2, and R+3.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

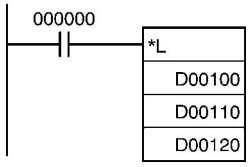
When *L(421) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R, R+1, R+2, R+3 is 0000 hex, the Equals Flag will turn ON.

If as a result of the multiplication, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00100, D00110, D00111, and D00110 will be multiplied as 8-digit signed hexadecimal values and the result will be output to D00121 and D00120.

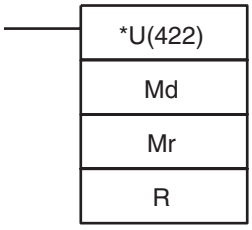


3-11-19 UNSIGNED BINARY MULTIPLY: *U(422)

Purpose

Multiplies 4-digit unsigned hexadecimal data and/or constants.

Ladder Symbol



Md: Multiplicand word

Mr: Multiplier word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*U(422)
	Executed Once for Upward Differentiation	@*U(422)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W511		W000 to W510
Holding Bit Area	H000 to H511		H000 to H510
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094

Area	Md	Mr	R
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D00000 to D32767		D00000 to D32766
EM Area without bank	E00000 to E32767		E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_ 32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*U(420) multiplies the binary values in Md and Mr and outputs the result to R, R+1.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

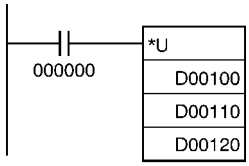
When *U(422) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R, R+1 is 0000 hex, the Equals Flag will turn ON.

If as a result of the multiplication, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

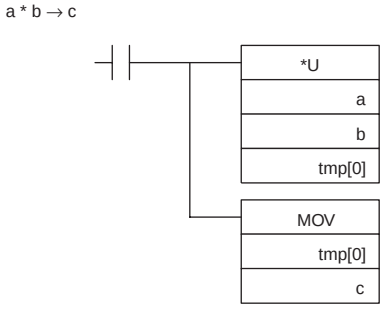
Examples

When CIO 000000 is ON in the following example, D00100 and D00110 will be multiplied as 4-digit unsigned binary values and the result will be output to D00121 and D00120.



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.



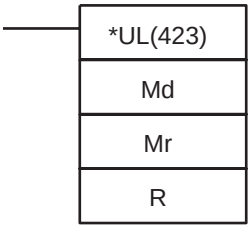
Function Block Variables
- Multiplicand: a (data type: UINT)
- Multiplier: b (data type: UINT)
- Result: c (data type: UINT)
- Temporary variable: tmp (data type: WORD, 2-element array)

3-11-20 DOUBLE UNSIGNED BINARY MULTIPLY: *UL(423)

Purpose

Multiplies 8-digit unsigned hexadecimal data and/or constants.

Ladder Symbol



Md: 1st multiplicand word

Mr: 1st multiplier word

R: 1st result word

Variations

Variations	**Executed Each Cycle for ON Condition**	*UL(423)
	Executed Once for Upward Differentiation	@*UL(423)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W510		W000 to W508
Holding Bit Area	H000 to H510		H000 to H508
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D00000 to D32766		D00000 to D32764

Area	Md	Mr	R
EM Area without bank	E00000 to E32766		E00000 to E32764
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		En_00000 to En_32764 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*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.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

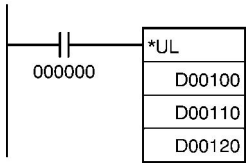
When *UL(423) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R, R+1, R+2, R+3 is 0000 hex, the Equals Flag will turn ON.

If as a result of the multiplication, the content of the leftmost bit of R+3 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00100, D00110, D00111, and D00110 will be multiplied as 8-digit unsigned binary values and the result will be output to D00123, D00122, D00121, and D00120.

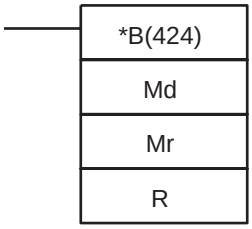


3-11-21 BCD MULTIPLY: *B(424)

Purpose

Multiplies 4-digit (single-word) BCD data and/or constants.

Ladder Symbol



Md: Multiplicand word

Mr: Multiplier word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*B(424)
	Executed Once for Upward Differentiation	@*B(424)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W511		W000 to W510
Holding Bit Area	H000 to H511		H000 to H510
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D00000 to D32767		D00000 to D32766
EM Area without bank	E00000 to E32767		E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		

Area	Md	Mr	R
Constants	#0000 to #9999 (BCD)		---
Data Registers	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*B(424) multiplies the BCD content of Md and Mr and outputs the result to R, R+1.

Flags

Name	Label	Operation
Error Flag	ER	ON when Md is not BCD. ON when Mr is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.

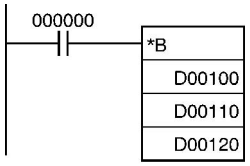
Precautions

If Md and/or Mr are not BCD, an error will be generated and the Error Flag will turn ON.

If as a result of the multiplication, the content of R, R+1 is 0000 hex, the Equals Flag will turn ON.

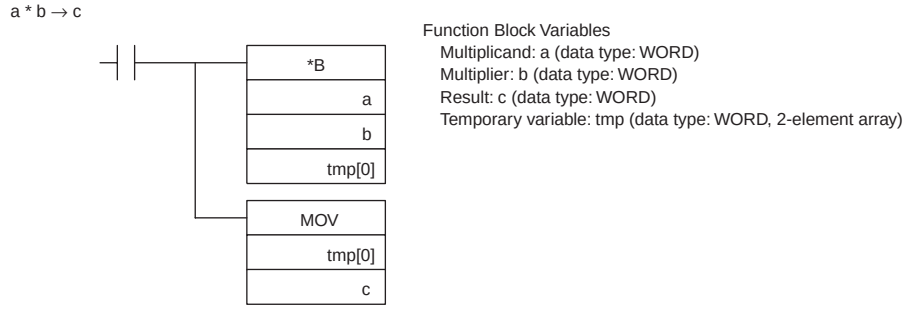
Examples

When CIO 000000 is ON in the following example, D00100 and D00110 will be multiplied as 4-digit BCD values and the result will be output to D00121 and D00120.



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.



3-11-22 DOUBLE BCD MULTIPLY: *BL(425)

Purpose

Multiplies 8-digit (double-word) BCD data and/or constants.

Ladder Symbol

Md: 1st multiplicand word

Mr: 1st multiplier word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*BL(425)
	Executed Once for Upward Differentiation	@*BL(425)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

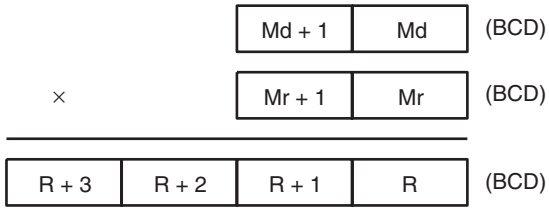
Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W510		W000 to W508
Holding Bit Area	H000 to H510		H000 to H508
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D00000 to D32766		D00000 to D32764
EM Area without bank	E00000 to E32766		E00000 to E32764
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		En_00000 to En_32764 (n = 0 to C)

Area	Md	Mr	R
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #99999999 (BCD)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*BL(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.



Flags

Name	Label	Operation
Error Flag	ER	ON when Md and/or Md+1 are not BCD. ON when Mr and/or Mr +1 are not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.

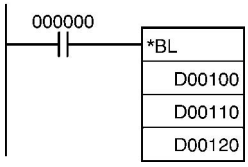
Precautions

If Md, Md+1 and/or Mr, Mr+1 are not BCD, an error will be generated and the Error Flag will turn ON.

If as a result of the multiplication, the content of R, R+1, R+2, R+3 is 00000000 hex, the Equals Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00101, D00100, D00111, and D00110 will be multiplied as 8-digit unsigned BCD values and the result will be output to D00123, D00122, D00121 and D00120.

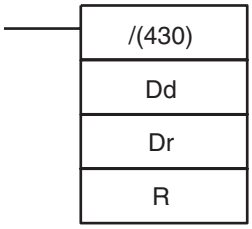


3-11-23 SIGNED BINARY DIVIDE: /(430)

Purpose

Divides 4-digit (single-word) signed hexadecimal data and/or constants.

Ladder Symbol



Dd: Dividend word

Dr: Divisor word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/(430)
	Executed Once for Upward Differentiation	@/(430)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

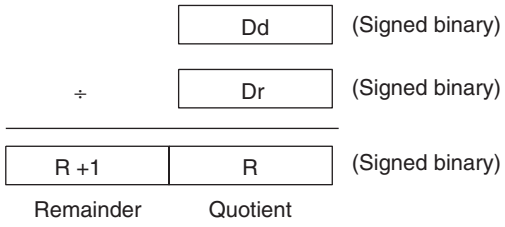
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W511		W000 to W510
Holding Bit Area	H000 to H511		H000 to H510
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D00000 to D32767		D00000 to D32766
EM Area without bank	E00000 to E32767		E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	#0001 to #FFFF (binary)	---
Data Registers	DR0 to DR15		---

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

Description

/(430) divides the signed binary (16 bit) values in Dd by those in Dr and outputs the result to R, R+1. The quotient is placed in R and the remainder in R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when the result is 0. OFF in all other cases.
Equals Flag	=	ON when as a result of the division, R is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R is 1. OFF in all other cases.

Precautions

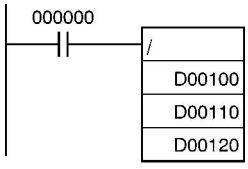
When the content of Dr is 0, an error will be generated and the Error Flag will turn ON.

If as a result of the division, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the division, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

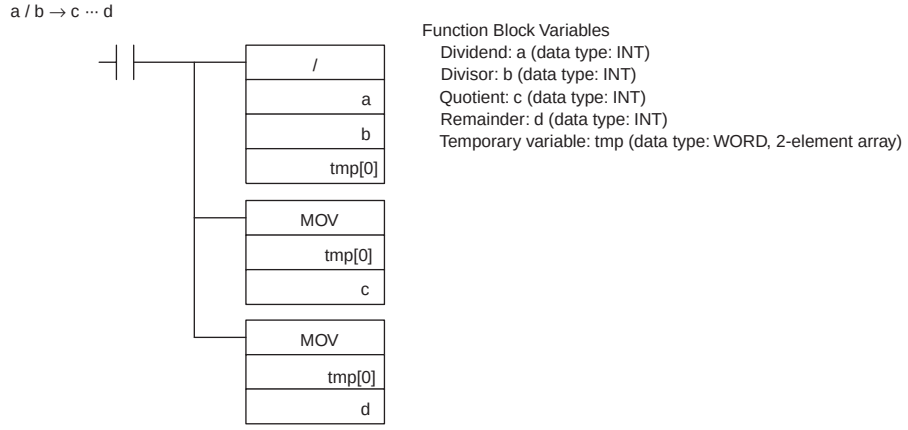
Examples

When CIO 000000 is ON in the following example, D00100 will be divided by D00110 as 4-digit signed binary values and the quotient will be output to D00120 and the remainder to D00121.



Example in Function Block Definition

In the following example, an array variable is used to get the quotient and remainder from the function block.

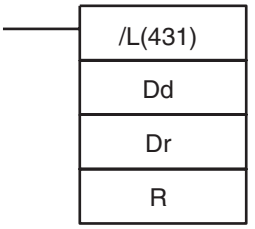


3-11-24 DOUBLE SIGNED BINARY DIVIDE: /L(431)

Purpose

Divides 8-digit (double-word) signed hexadecimal data and/or constants.

Ladder Symbol



Dd: 1st dividend word

Dr: 1st divisor word

R: 1st result word

Variations

Variations	**Executed Each Cycle for ON Condition**	/L(431)
	Executed Once for Upward Differentiation	@/L(431)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

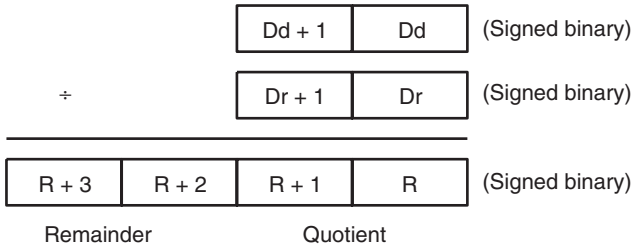
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W510		W000 to W508
Holding Bit Area	H000 to H510		H000 to H508
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D00000 to D32766		D00000 to D32764
EM Area without bank	E00000 to E32766		E00000 to E32764

Area	Dd	Dr	R
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		En_00000 to En_32764 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)	#00000001 to #FFFFFFFF (binary)	---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

/L(431) divides the signed binary values in Dd and Dd+1 by those in Dr and Dr+1 and outputs the result to R, R+1, R+2, and R+3. The quotient is output to R and R+1 and the remainder is output to R+2 and R+3.



Flags

Name	Label	Operation
Error Flag	ER	ON when the result is 0. OFF in all other cases.
Equals Flag	=	ON when as a result of the division, R+1, R is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R+1, R is 1. OFF in all other cases.

Precautions

When the remainder of the result, R+3, R+2 is 0,the Error Flag will turn ON.

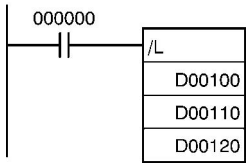
If as a result of the division, the content of R+1, R is 00000000 hex, the Equals Flag will turn ON.

If as a result of the division, the content of the leftmost bit of R+1, R is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00101 and D00100 are divided by D00111 and D00110 as 8-digit signed hexadecimal values and the

quotient will be output to D00121 and D00120 and the remainder to D00123 and D00122.

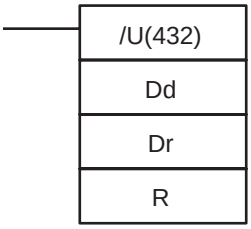


3-11-25 UNSIGNED BINARY DIVIDE: /U(432)

Purpose

Divides 4-digit (single-word) unsigned hexadecimal data and/or constants.

Ladder Symbol



Dd: Dividend word

Dr: Divisor word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/U(432)
	Executed Once for Upward Differentiation	@/U(432)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

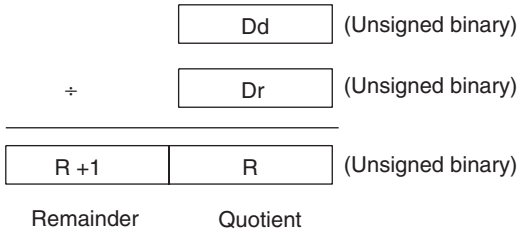
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W511		W000 to W510
Holding Bit Area	H000 to H511		H000 to H510
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D00000 to D32767		D00000 to D32766
EM Area without bank	E00000 to E32767		E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		

Area	Dd	Dr	R
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	#0001 to #FFFF (binary)	---
Data Registers	DR0 to 15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

/U(432) divides the unsigned binary values in Dd by those in Dr and outputs the quotient to R and the remainder to R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when the result is 0. OFF in all other cases.
Equals Flag	=	ON when as a result of the division, R is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R is 1. OFF in all other cases.

Precautions

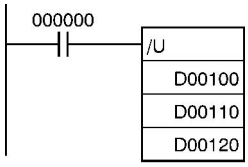
If as a result of the division, the content of R+1 is 0, the Error Flag will turn ON.

If as a result of the division, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the division, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

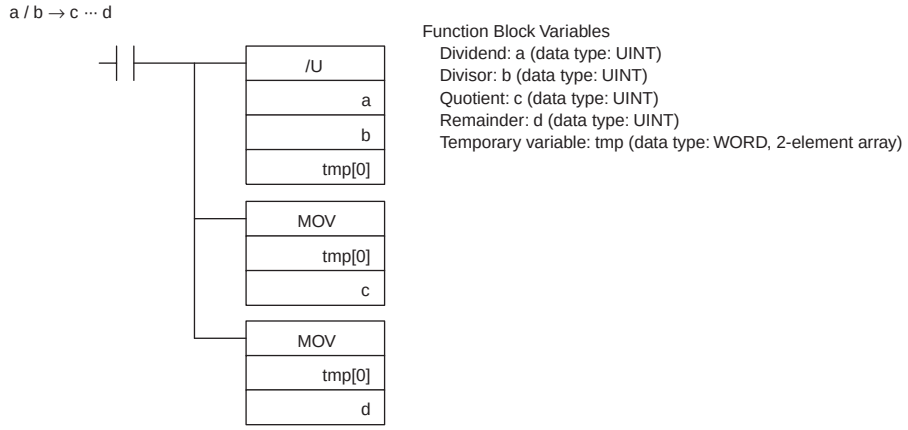
Examples

When CIO 000000 is ON in the following example, D00100 will be divided by D00110 as 4-digit unsigned binary values and the quotient will be output to D00120 and the remainder will be output to D00121.



Example in Function Block Definition

In the following example, an array variable is used to get the quotient and remainder from the function block.

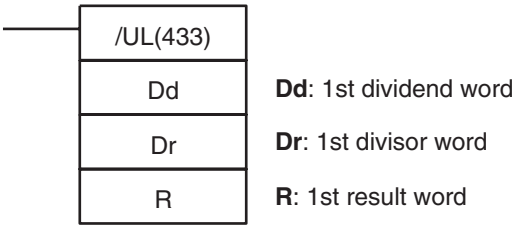


3-11-26 DOUBLE UNSIGNED BINARY DIVIDE: /UL(433)

Purpose

Divides 8-digit (double-word) unsigned hexadecimal data and/or constants.

Ladder Symbol



Variations

Variations	Executed Each Cycle for ON Condition	/UL(433)
	Executed Once for Upward Differentiation	@/UL(433)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

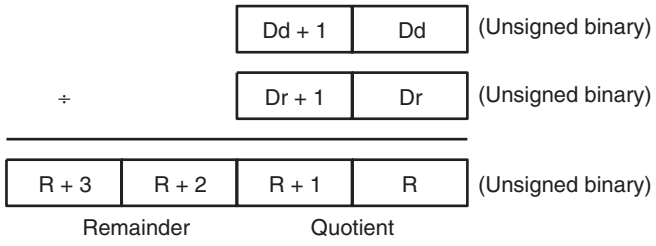
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W510		W000 to W508
Holding Bit Area	H000 to H510		H000 to H508
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D00000 to D32766		D00000 to D32764
EM Area without bank	E00000 to E32766		E00000 to E32764

Area	Dd	Dr	R
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		En_00000 to En_32764 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)	#00000001 to #FFFFFFFF (binary)	---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

/UL(433) divides the unsigned binary values in Dd and Dd+1 by those in Dr and Dr+1 and outputs the quotient to R, R+1 and the remainder to R+2, and R+3.



Flags

Name	Label	Operation
Error Flag	ER	ON when the result is 0. OFF in all other cases.
Equals Flag	=	ON when as a result of the division R+1, R is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R+1, R is 1. OFF in all other cases.

Precautions

When the content of Dr, Dr+1 is 0, the Error Flag will turn ON.

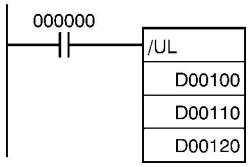
If as a result of the division, the content of R, R+1, is 0000 hex, the Equals Flag will turn ON.

If as a result of the division, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00100 and D00101 will be divided by D00111 and D00110 as 8-digit unsigned hexadecimal values

and the quotient will be output to D00121 and D00120 and the remainder to D00123 and D00122.

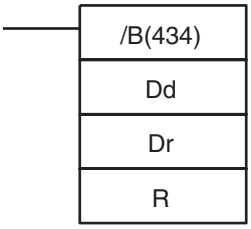


3-11-27 BCD DIVIDE: /B(434)

Purpose

Divides 4-digit (single-word) BCD data and/or constants.

Ladder Symbol



Dd: Dividend word

Dr: Divisor word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/B(434)
	Executed Once for Upward Differentiation	@/B(434)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

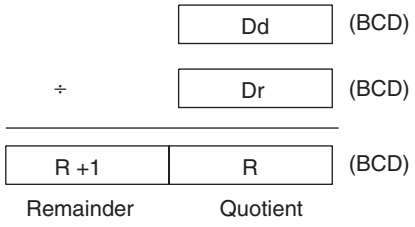
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W511		W000 to W510
Holding Bit Area	H000 to H511		H000 to H510
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D00000 to D32767		D00000 to D32766
EM Area without bank	E00000 to E32767		E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		

Area	Dd	Dr	R
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #9999 (BCD)	#0001 to #9999 (BCD)	---
Data Registers	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

/B(434) divides the BCD content of Dd by those of Dr and outputs the quotient to R and the remainder to R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when Dd is not BCD. ON when Dr is not BCD. ON when the remainder is 0. OFF in all other cases.
Equals Flag	=	ON when R is 0. OFF in all other cases.

Precautions

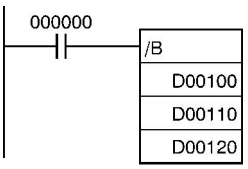
If Dd or Dr are not BCD or if the remainder (R+1) is 0, an error will be generated and the Error Flag will turn ON.

If as a result of the division, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the division, the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

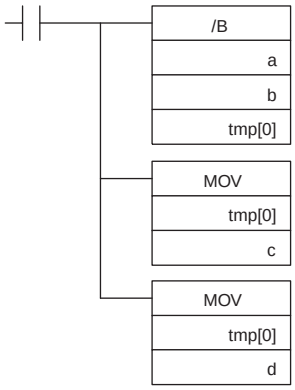
When CIO 000000 is ON in the following example, D00100 will be divided by D00110 as 4-digit BCD values and the quotient will be output to D00120 and the remainder to D00120.



Example in Function Block Definition

In the following example, an array variable is used to get the quotient and remainder from the function block.

a / b → c ··· d



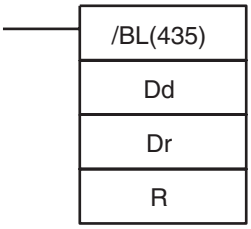
Function Block Variables
Dividend: a (data type: WORD)
Divisor: b (data type: WORD)
Quotient: c (data type: WORD)
Remainder: d (data type: WORD)
Temporary variable: tmp (data type: WORD, 2-element array)

3-11-28 DOUBLE BCD DIVIDE: /BL(435)

Purpose

Divides 8-digit (double-word) BCD data and/or constants.

Ladder Symbol



Dd: 1st dividend word

Dr: 1st divisor word

R: 1st result word

Variations

Variations	**Executed Each Cycle for ON Condition**	/BL(435)
	Executed Once for Upward Differentiation	@/BL(435)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

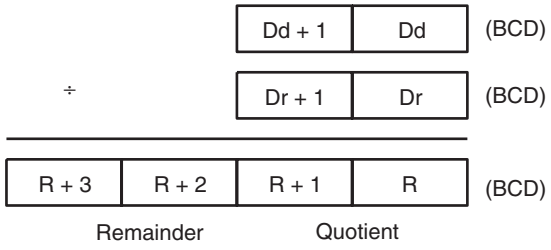
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W510		W000 to W508
Holding Bit Area	H000 to H510		H000 to H508
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T4094		T0000 to T4092
Counter Area	C0000 to C4094		C0000 to C4092
DM Area	D00000 to D32766		D00000 to D32764
EM Area without bank	E00000 to E32766		E00000 to E32764
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		En_00000 to En_32764 (n = 0 to C)

Area	Dd	Dr	R
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #99999999 (BCD)	#00000001 to #99999999 (BCD)	---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

/BL(435) divides BCD values in Dd and Dd+1 by those in Dr and Dr+1 and outputs the quotient to R, R+1 and the remainder to R+2, R+3.



Flags

Name	Label	Operation
Error Flag	ER	ON when Dd, Dd+1 is not BCD. ON when Dr, Dr +1 is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0. OFF in all other cases.

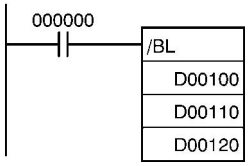
Precautions

If Dd, Dd+1 and/or Dr, Dr+1 are not BCD or the content of Dr, Dr+1 is 0, an error will be generated and the Error Flag will turn ON.

If as a result of the division, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, D00101 and D00100 will be divided by D00111 and D00110 as 8-digit BCD values and the quotient will be output to D00121 and D00120 and the remainder to D00123 and D00122.



3-12 Conversion Instructions

This section describes instructions used for data conversion.

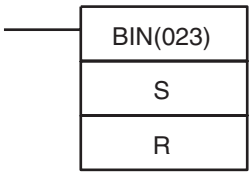
Instruction	Mnemonic	Function code	Page
BCD TO BINARY	BIN	023	483
DOUBLE BCD TO DOUBLE BINARY	BINL	058	485
BINARY TO BCD	BCD	024	487
DOUBLE BINARY TO DOUBLE BCD	BCDL	059	489
2'S COMPLEMENT	NEG	160	491
DOUBLE 2'S COMPLEMENT	NEGL	161	493
16-BIT TO 32-BIT SIGNED BINARY	SIGN	600	494
DATA DECODER	MLPX	076	496
DATA ENCODER	DMPX	077	500
ASCII CONVERT	ASC	086	504
ASCII TO HEX	HEX	162	508
COLUMN TO LINE	LINE	063	512
LINE TO COLUMN	COLM	064	514
SIGNED BCD TO BINARY	BINS	470	517
DOUBLE SIGNED BCD TO BINARY	BISL	472	520
SIGNED BINARY TO BCD	BCDS	471	523
DOUBLE SIGNED BINARY TO BCD	BDSL	473	525
GRAY CODE CONVERSION	GRY	474	529
FOUR-DIGIT NUMBER TO ASCII	STR4	601	534
EIGHT-DIGIT NUMBER TO ASCII	STR8	602	537
SIXTEEN-DIGIT NUMBER TO ASCII	STR16	603	539
ASCII TO FOUR-DIGIT NUMBER	NUM4	604	541
ASCII TO EIGHT-DIGIT NUMBER	NUM8	605	544
ASCII TO SIXTEEN-DIGIT NUMBER	NUM16	606	545

3-12-1 BCD TO BINARY: BIN(023)

Purpose

Converts BCD data to binary data.

Ladder Symbol



S: Source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BIN(023)
	Executed Once for Upward Differentiation	@BIN(023)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

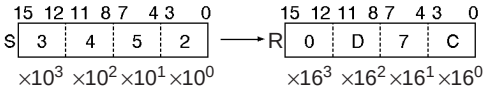
BIN(023) converts the BCD data in S to binary data and writes the result to R.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of S is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	OFF

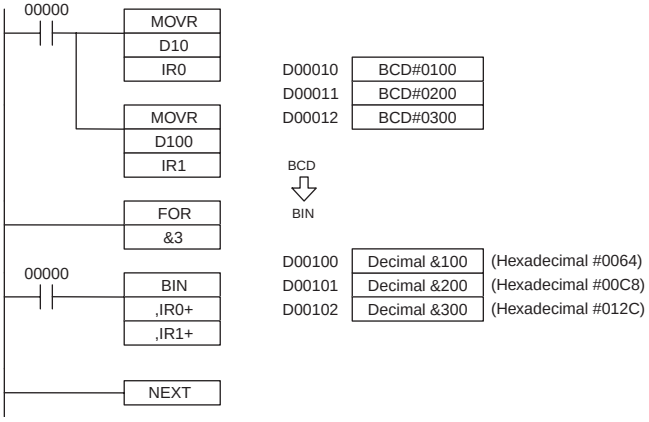
Example

The following diagram shows an example BCD-to-binary conversion.



In this example, N words of BCD data is converted to binary data.

If N = 3, the three words of BCD starting from D00010 will be converted to binary data one word at a time when CIO 00000 turns ON. The resulting binary data will be stored starting from D00100.

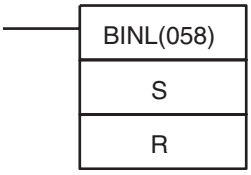


3-12-2 DOUBLE BCD TO DOUBLE BINARY: BINL(058)

Purpose

Converts 8-digit BCD data to 8-digit hexadecimal (32-bit binary) data.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BINL(058)
	Executed Once for Upward Differentiation	@BINL(058)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	

Area	S	R
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

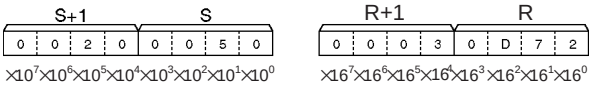
BINL(058) converts the 8-digit BCD data in S and S+1 to 8-digit hexadecimal (32-bit binary) data and writes the result to R and R+1.

Flags

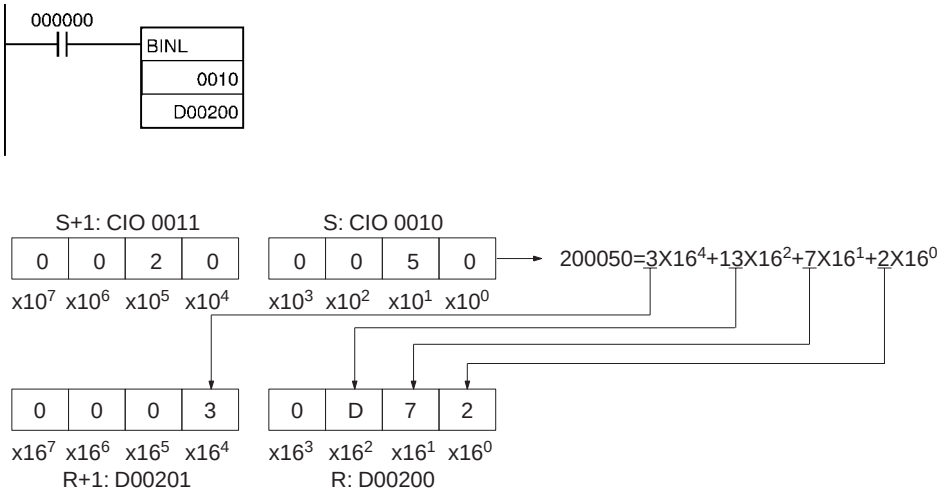
Name	Label	Operation
Error Flag	ER	ON if the contents of S+1, S are not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Negative Flag	N	OFF

Examples

The following diagram shows an example of 8-digit BCD-to-binary conversion.



When CIO 000000 is ON in the following example, the 8-digit BCD value in CIO 0010 and CIO 0011 is converted to hexadecimal and stored in D00200 and D00201.

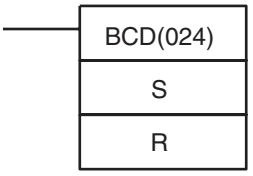


3-12-3 BINARY TO BCD: BCD(024)

Purpose

Converts a word of binary data to a word of BCD data.

Ladder Symbol



S: Source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BCD(024)
	Executed Once for Upward Differentiation	@BCD(024)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: Source Word

S must be between 0000 and 270F hexadecimal (0000 and 9999 decimal).

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	

Area	S	R
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

BCD(024) converts the binary data in S to BCD data and writes the result to R.

Flags

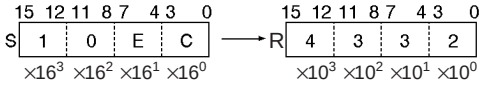
Name	Label	Operation
Error Flag	ER	ON if the content of S exceeds 270F (9999 decimal). OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

Precautions

The content of S must not exceed 270F (9999 decimal).

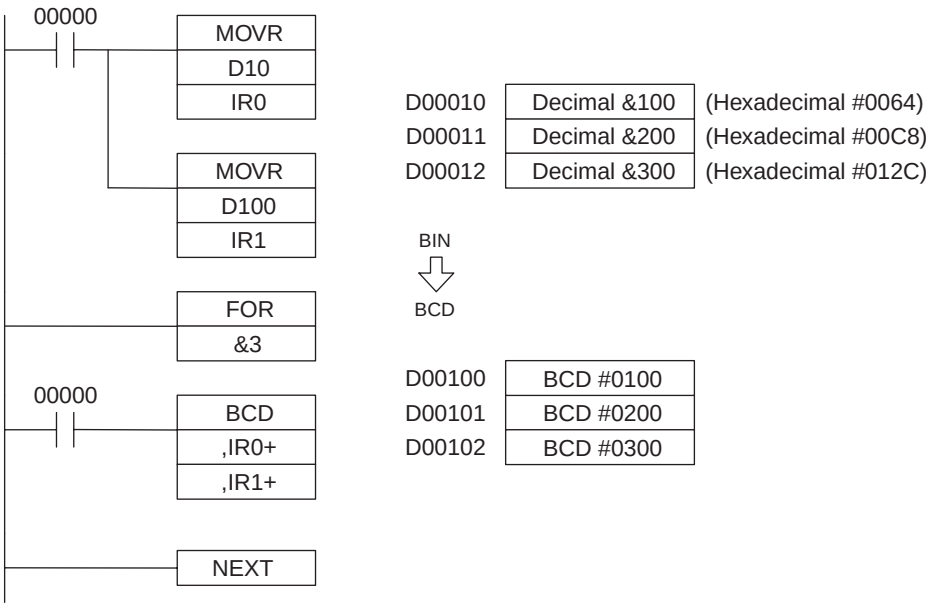
Example

The following diagram shows an example BCD-to-binary conversion.



In this example, N words of binary data is converted to BCD data.

If N = 3, the three words of binary starting from D00010 will be converted to binary data one word at a time when CIO 00000 turns ON. The resulting BCD data will be stored starting from D00100.

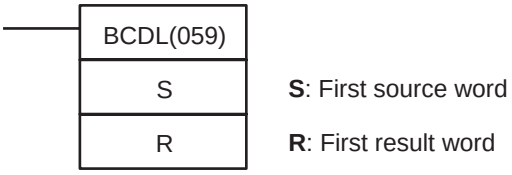


3-12-4 DOUBLE BINARY TO DOUBLE BCD: BCDL(059)

Purpose

Converts 8-digit hexadecimal (32-bit binary) data to 8-digit BCD data.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	BCDL(059)
	Executed Once for Upward Differentiation	@BCDL(059)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: First Source Word

The content of S+1 and S must be between 0000 0000 and 05F5 E0FF hexadecimal (0000 0000 and 9999 9999 decimal).

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	

Area	S	R
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

BCDL(059) converts the 8-digit hexadecimal (32-bit binary) data in S and S+1 to 8-digit BCD data and writes the result to R and R+1.

Flags

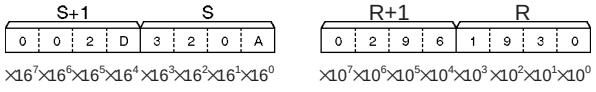
Name	Label	Operation
Error Flag	ER	ON if the contents of S and S+1 exceed 05F5 E0FF (9999 9999 decimal). OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.

Precautions

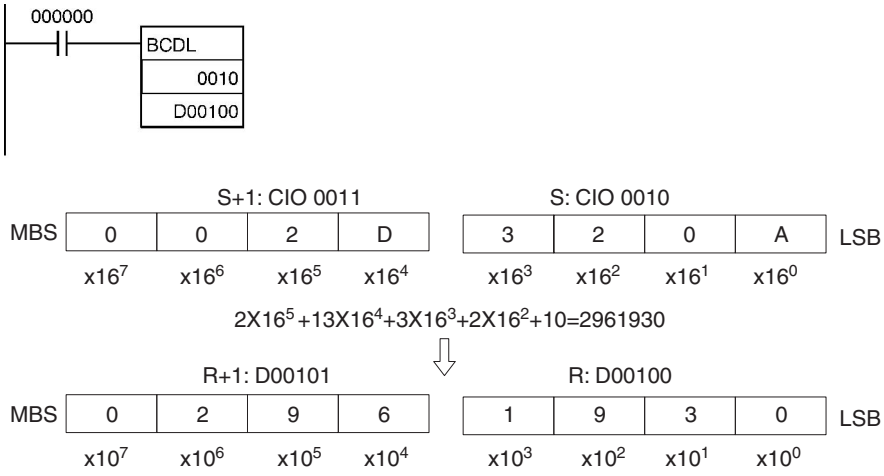
The content of S+1 and S must not exceed 05F5 E0FF (9999 9999 decimal).

Examples

The following diagram shows an example of 8-digit BCD-to-binary conversion.



When CIO 000000 is ON in the following example, the hexadecimal value in CIO 0011 and CIO 0010 is converted to a BCD value and stored in D00200 and D00201.

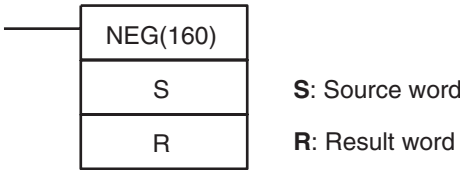


3-12-5 2'S COMPLEMENT: NEG(160)

Purpose

Calculates the 2's complement of a word of hexadecimal data.

Ladder Symbol



S: Source word

R: Result word

Variations

Variations	**Executed Each Cycle for ON Condition**	NEG(160)
	Executed Once for Upward Differentiation	@NEG(160)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

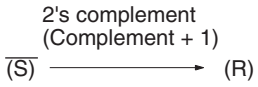
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	

Area	S	R
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

NEG(160) calculates the 2's complement of S and writes the result to R. The 2's complement calculation basically reverses the status of the bits in S and adds 1.



Note This operation (reversing the status of the bits and adding 1) is equivalent to subtracting the content of S from 0000.

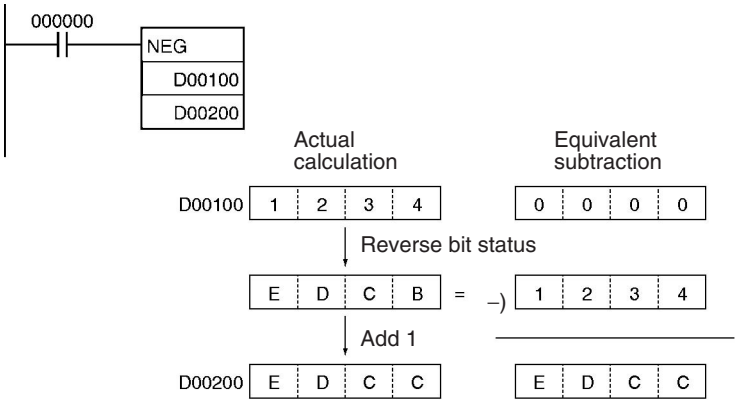
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of the result is ON. OFF in all other cases.

Note The result for 8000 hex will be 8000 hex.

Example

When CIO 000000 is ON in the following example, NEG(160) calculates the 2's complement of the content of D00100 and writes the result to D00200.

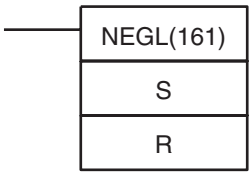


3-12-6 DOUBLE 2'S COMPLEMENT: NEGL(161)

Purpose

Calculates the 2's complement of two words of hexadecimal data.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	NEGL(161)
	Executed Once for Upward Differentiation	@NEGL(161)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

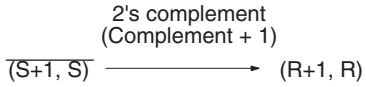
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Note R and R+1 must be in the same data area.

Description

NEGL(161) calculates the 2's complement of S+1 and S and writes the result to R+1 and R. The 2's complement calculation basically reverses the status of the bits in S+1 and S and adds 1.



Note This operation (reversing the status of the bits and adding 1) is equivalent to subtracting the content of S+1 and S from 0000 0000.

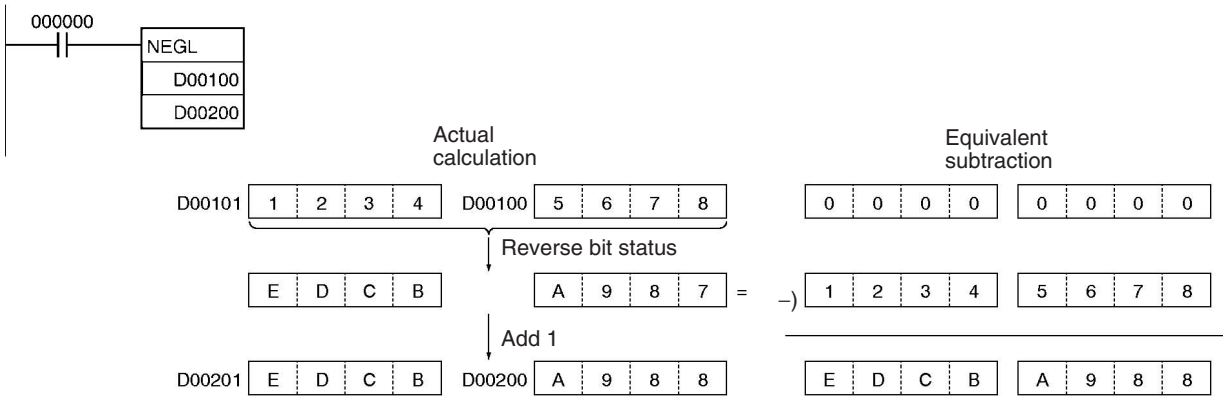
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of R+1 is ON. OFF in all other cases.

Note The result for 8000 hex will be 8000 hex.

Example

When CIO 000000 is ON in the following example, NEGL(161) calculates the 2's complement of the content of D00101 and D00100 and writes the result to D00201 and D00200.

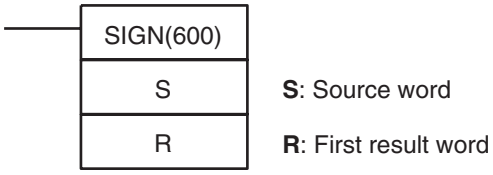


3-12-7 16-BIT TO 32-BIT SIGNED BINARY: SIGN(600)

Purpose

Expands a 16-bit signed binary value to its 32-bit equivalent.

Ladder Symbol



S: Source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	SIGN(600)
	Executed Once for Upward Differentiation	@SIGN(600)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications

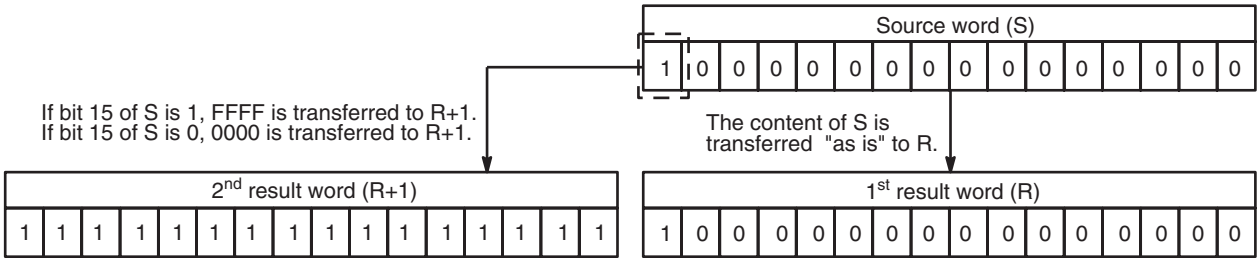
Area	S	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142
Work Area	W000 to W511	W000 to W510
Holding Bit Area	H000 to H511	H000 to H510
Auxiliary Bit Area	A000 to A959	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094
Counter Area	C0000 to C4095	C0000 to C4094
DM Area	D00000 to D32767	D00000 to D32766
EM Area without bank	E00000 to E32767	E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Note R and R+1 must be in the same data area.

Description

SIGN(600) converts the 16-bit signed binary number in S to its 32-bit signed binary equivalent and writes the result in R+1 and R.

The conversion is accomplished by copying the content of S to R and writing FFFF to R+1 if bit 15 of S is 1 or writing 0000 to R+1 if bit 15 of S is 0.

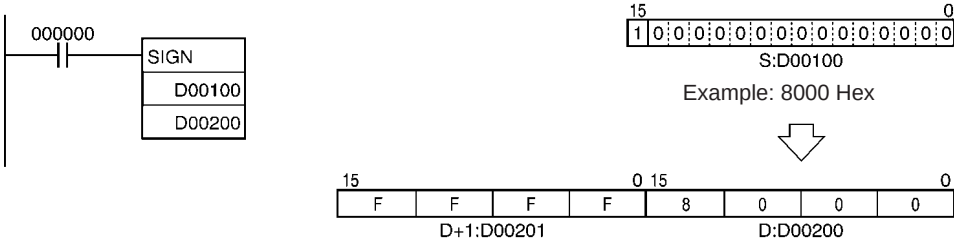


Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of R+1 is ON. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, SIGN(600) converts the 16-bit signed binary content of D00100 (#8000 = –32,768 decimal) to its 32-bit equivalent (#FFFF 8000 = –32,768 decimal) and writes that result to D00201 and D00200.

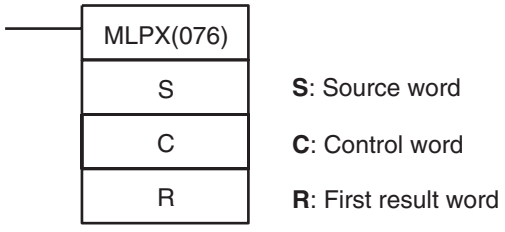


3-12-8 DATA DECODER: MLPX(076)

Purpose

Reads the numerical value in the specified digit (or byte) in the source word, turns ON the corresponding bit in the result word (or 16-word range), and turns OFF all other bits in the result word (or 16-word range).

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	MLPX(076)
	Executed Once for Upward Differentiation	@MLPX(076)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

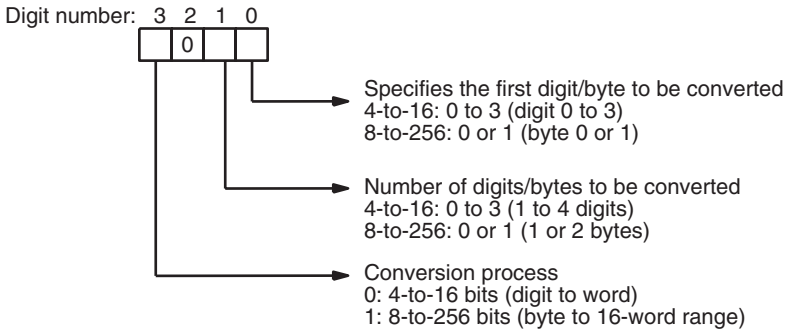
Operands

S: Source Word

The data in the source word indicates the location of the bit(s) that will be turned ON.

C: Control Word

The control word specifies whether MLPX(076) will perform a 4-to-16 bit conversion or an 8-to-256 bit conversion, the number of digits or bytes to be converted, and the starting digit or byte.



R: First result word

There can be anywhere from 1 to 32 result words, depending upon the type of conversion process and number of digits/bytes being converted. The result words must be in the same data area.

Operand Specifications

Area	S	C	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	---
Data Registers	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

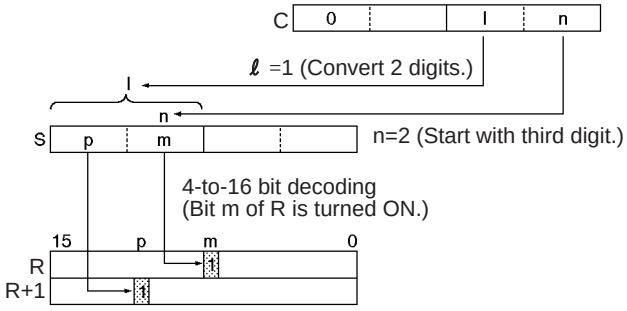
Description

MLPX(076) can perform 4-to-16 bit or 8-to-256 bit conversions. Set the leftmost digit of C to 0 to specify 4-to-16 bit conversion and set it to 1 to specify 8-to-256 bit conversion.

4-to-16 bit Conversion

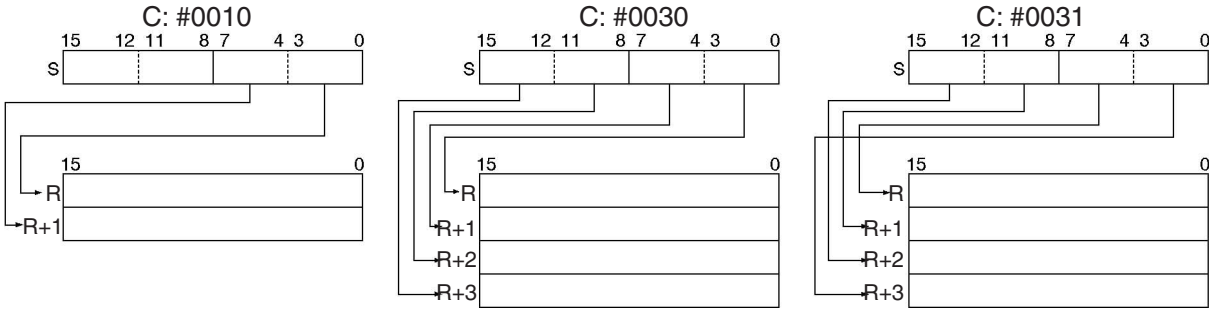
When the leftmost digit of C is 0, MLPX(076) takes the value of the specified digit in S (0 to F) and turns ON the corresponding bit in the result word. All

other bits in the result word will be turned OFF. Up to four digits can be converted.



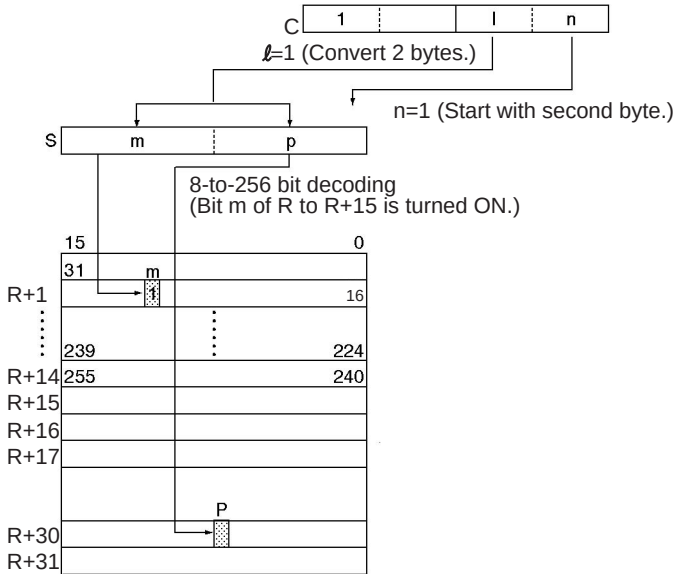
When two or more digits are being converted, MLPX(076) will read the digits in S from right to left and will wrap around to the rightmost digit after the leftmost digit, if necessary.

The following diagram shows some example values for C and the 4-to-16 bit conversions that they produce.



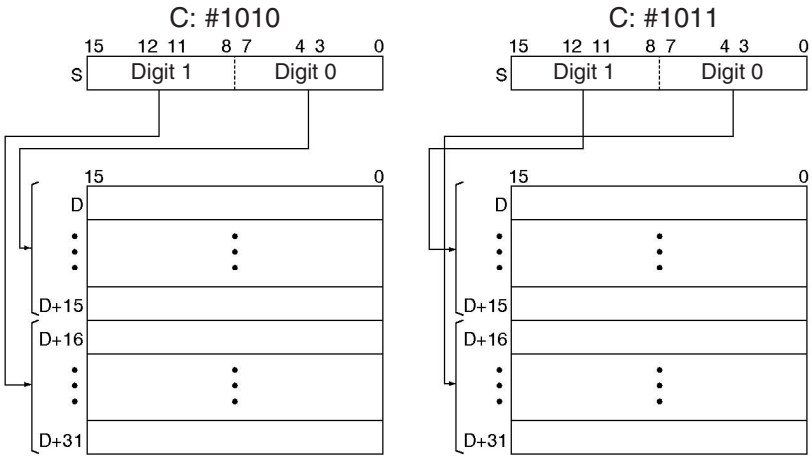
8-to-256 bit Conversion

When the leftmost digit of C is 1, MLPX(076) takes the value of the specified byte in S (00 to FF) and turns ON the corresponding bit in the range of 16 result words. All other bits in the result words will be turned OFF. Up to two bytes can be converted.



When two bytes are being converted, MLPX(076) will read the bytes in S from right to left and will wrap around to the rightmost byte if the leftmost byte (byte 1) has been specified as the starting byte.

The following diagram shows some example values for C and the 8-to-256 bit conversions that they produce.



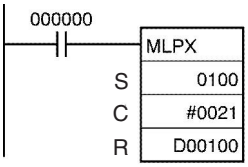
Flags

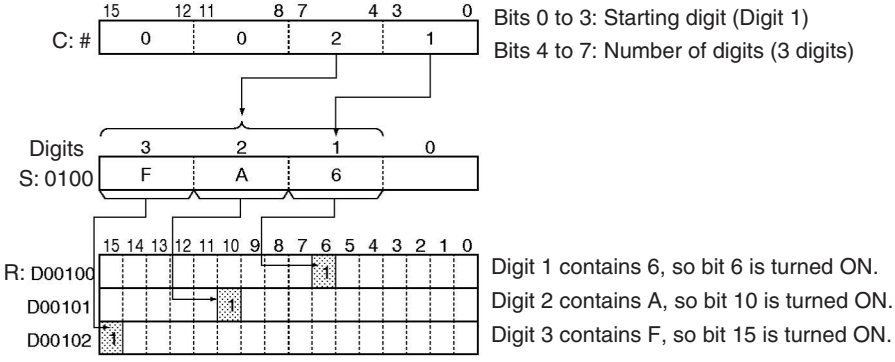
Name	Label	Operation
Error Flag	ER	ON if C is not within the specified ranges. OFF in all other cases.

Examples

4-to-16 bit Conversion

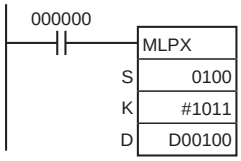
When CIO 000000 is ON in the following example, MLPX(076) will convert 3 digits in S beginning with digit 1 (the second digit), as indicated by C (#0021). The corresponding bits in D00100, D00101, and D00102 will be turned ON.

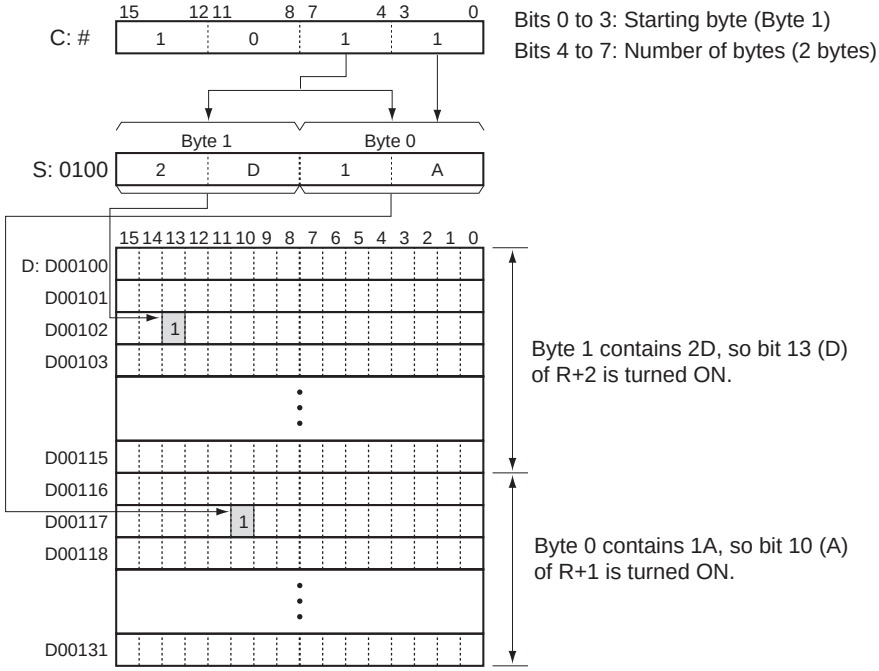




8-to-256 bit Conversion

When CIO 000000 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 D00100 to D00115 and D00116 to D00131 will be turned ON.



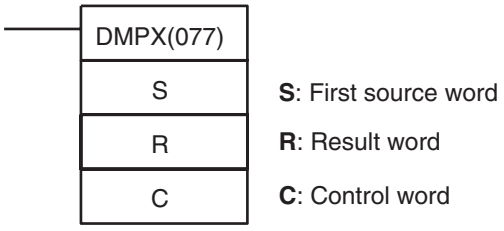


3-12-9 DATA ENCODER: DMPX(077)

Purpose

FInds the location of the first or last ON bit within the source word (or 16-word range), and writes that value to the specified digit (or byte) in the result word.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	DMPX(077)
	Executed Once for Upward Differentiation	@DMPX(077)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: First Source Word

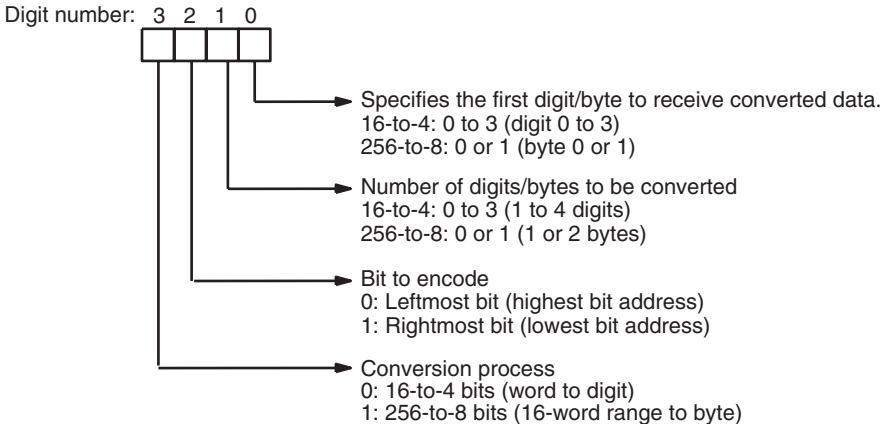
There can be anywhere from 1 to 32 source words, depending upon the type of conversion process and number of digits/bytes being converted. The source words must be in the same data area.

R: Result Word

The locations of the bits that were ON in the source word(s) are written to the digits/bytes in R starting with the specified first digit/byte.

C: Control Word

The control word specifies whether DMPX(077) will perform a 16-to-4 bit conversion or an 256-to-8 bit conversion, whether the leftmost or rightmost ON bit will be encoded, the number of digits or bytes that will be converted, and the starting digit or byte where the results will be written.



Operand Specifications

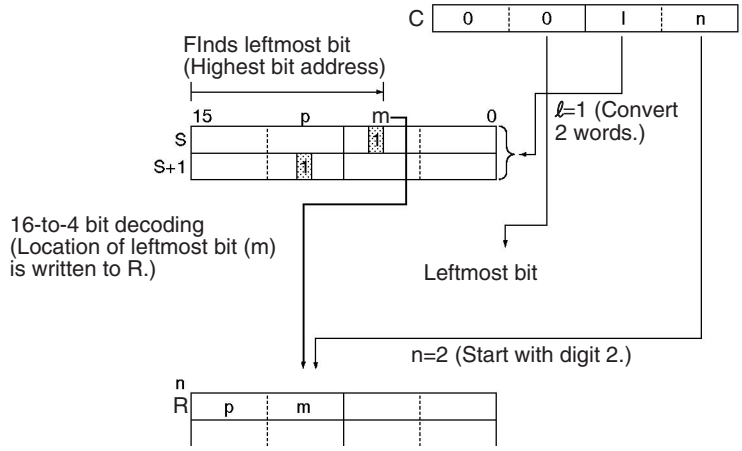
Area	S	R	C
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959	A448 to A959	A000 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	---	Specified values only
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

DMPX(077) can perform 16-to-4 bit or 256-to-8 bit conversions. Set the leftmost digit of C to 0 to specify 16-to-4 bit conversion and set it to 1 to specify 256-to-8 bit conversion.

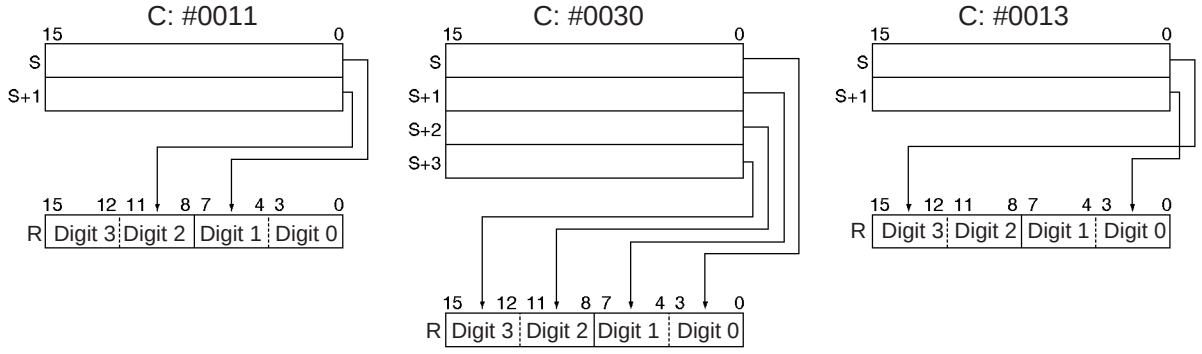
16-to-4 bit Conversion

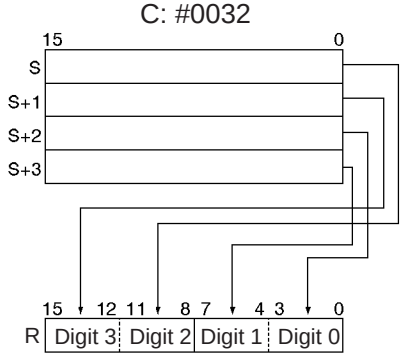
When the fourth (leftmost) digit of C is 0, DMPX(077) finds the locations of the leftmost or rightmost ON bits in up to 4 source words and writes these locations to R beginning with the specified digit. (Set the third digit of C to 0 to find the leftmost ON bits or 1 to find the rightmost ON bits.)



When two or more digits are being converted, DMPX(077) will write the values to the digits in R from right to left and will wrap around to the rightmost digit after the leftmost digit, if necessary.

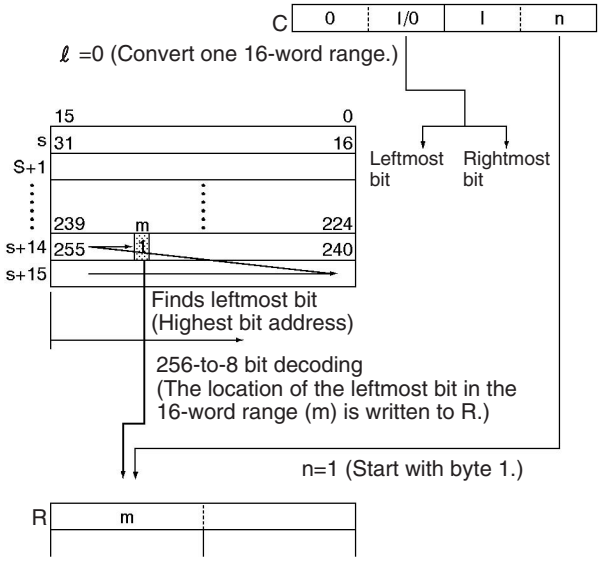
The following diagram shows some example values for C and the 16-to-4 bit conversions that they produce.





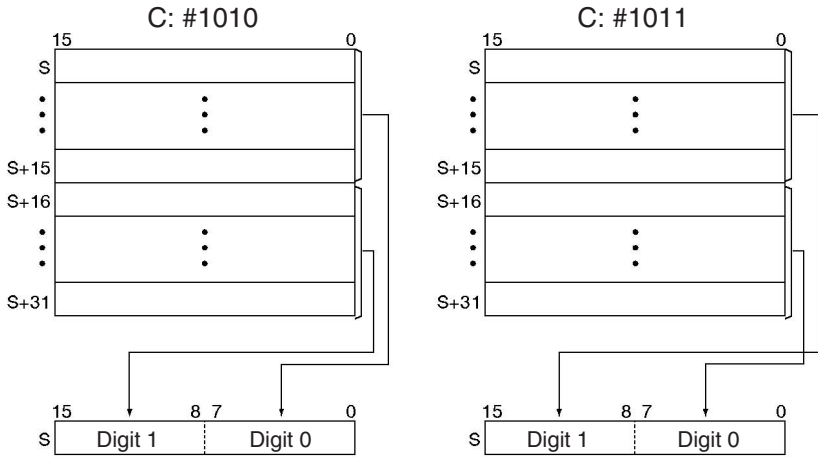
256-to-8 bit Conversion

When the fourth (leftmost) digit of C is 1, DMPX(077) finds the locations of the leftmost (highest bit address) or rightmost (lowest bit address) ON bits in one or two 16-word ranges of source words. The locations of these bits are written to R beginning with the specified byte. (Set the third digit of C to 0 to find the leftmost ON bits or 1 to find the rightmost ON bits.)



When two bytes are being converted, DMPX(077) will write the values to the bytes in R from right to left and will wrap around to the rightmost byte if the leftmost byte (byte 1) has been specified as the starting byte.

The following diagram shows some example values for C and the 256-to-8 bit conversions that they produce.



Flags

Name	Label	Operation
Error Flag	ER	ON if any of the source words contains 0000 hex (i.e., no bit to encode). ON if C is not within the specified ranges. OFF in all other cases.

Precautions

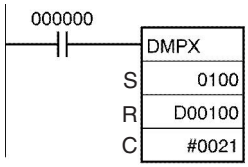
If the conversion data contains 0000 hex, but other data is to be encoded, separate the conversion by using more than one DMPX(077) instructions.

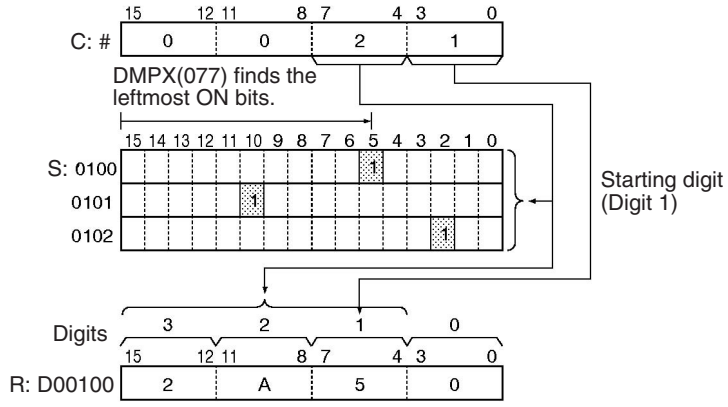
DMPX(077) D0000 D0100 #0300

```
DMPX(077)   D0000  D0100  #0000
DMPX(077)   D0001  D0100  #0001
DMPX(077)   D0002  D0100  #0002
DMPX(077)   D0003  D0100  #0003
```

Examples

When CIO 000000 is ON in the following example, DMPX(077) will find the leftmost ON bits in CIO 0100, CIO 0101, and CIO 0102 and write those locations to 3 digits in R beginning with digit 1 (the second digit), as indicated by C (#0021).



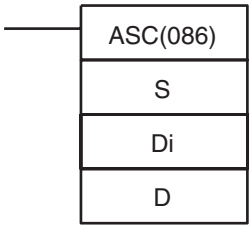


3-12-10 ASCII CONVERT: ASC(086)

Purpose

Converts 4-bit hexadecimal digits in the source word into their 8-bit ASCII equivalents.

Ladder Symbol



S: Source word

Di: Digit designator

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASC(086)
	Executed Once for Upward Differentiation	@ASC(086)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

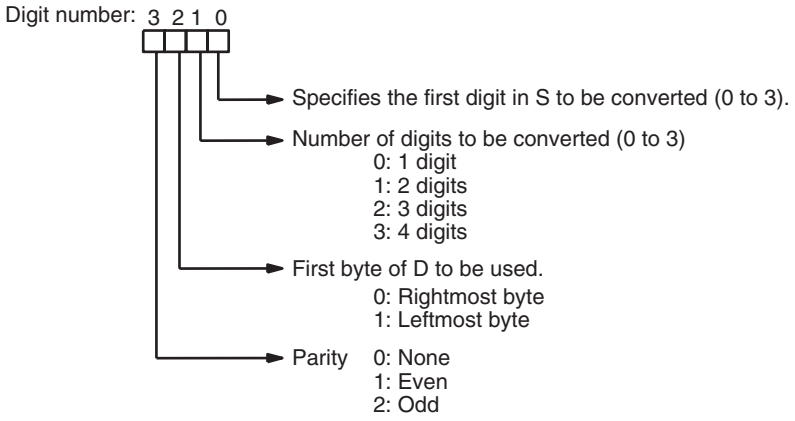
Operands

S: Source Word

Up to four digits in the source word can be converted. The digits are numbered 0 to 3, right to left.

Di: Digit Designator

The digit designator specifies various parameters for the conversion, as shown in the following diagram.



D: First destination word

The converted ASCII data is written to the destination word(s) beginning with the specified byte in D. Three destination words (D to D+3) will be required if 4 digits are being converted and the leftmost byte is selected as the first byte in D. The destination words must be in the same data area.

Any bytes in the destination word(s) that are not overwritten with ASCII data will be left unchanged.

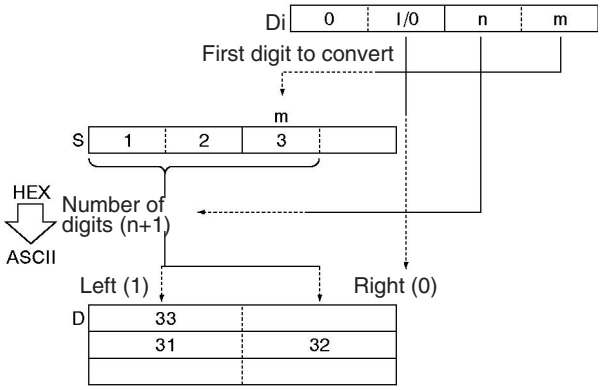
Operand Specifications

Area	S	Di	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	---
Data Registers	DR0 to DR15		---

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

Description

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.



Note Refer to *Appendix A* in the *CS/CJ-series Programming Consoles Operation Manual* (W341) for a table of extended ASCII characters.

Parity

It is possible to specify the parity of the ASCII data for use in error control during data transmissions. The leftmost bit of each ASCII character will be automatically adjusted for even, odd, or no parity.

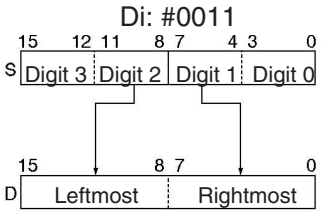
When no parity (0) is designated, the leftmost bit will always be zero. When even parity (1) is designated, the leftmost bit will be adjusted so that the total number of ON bits is even. When odd parity (2) is designated, the leftmost bit of each ASCII character will be adjusted so that there is an odd number of ON bits. The status of the parity bit does not affect the meaning of the ASCII code.

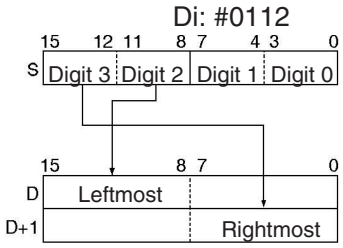
Examples of even parity:
When adjusted for even parity, ASCII “31” (00110001) will be “B1” (10110001: parity bit turned ON to create an even number of ON bits); ASCII “36” (00110110) will be “36” (00110110: parity bit remains OFF because the number of ON bits is already even).

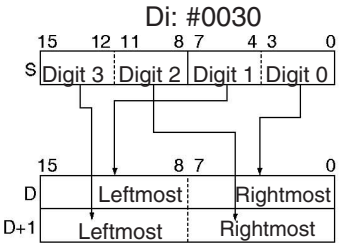
Examples of odd parity:
When adjusted for odd parity, ASCII “36” (00110110) will be “B6” (10110110: parity bit turned ON to create an odd number of ON bits); ASCII “46” (01000110) will be “46” (01000110: parity bit remains OFF because the number of ON bits is already odd).

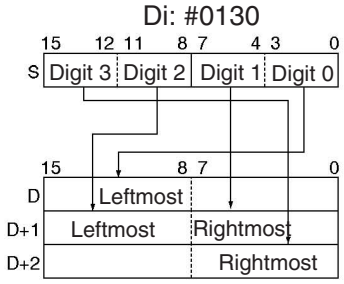
Examples of Di

When two or more digits are being converted, ASC(086) will read the bytes in S from right to left and will wrap around to the rightmost byte if necessary. The following diagram shows some example values for Di and the conversions that they produce.







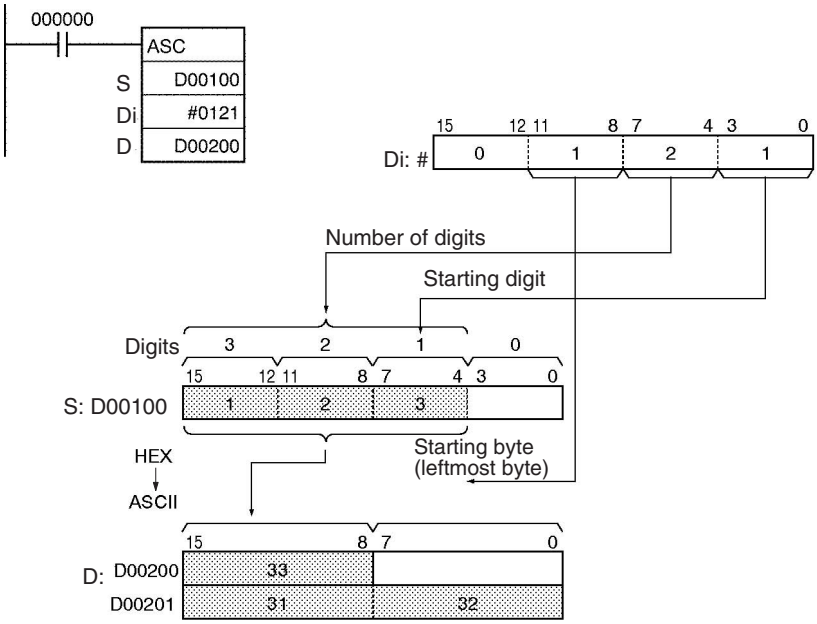


Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Di is not within the specified ranges. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, ASC(086) converts three hexadecimal digits in D00100 (beginning with digit 1) into their ASCII equivalents and writes this data to D00200 and D00201 beginning with the leftmost byte in D00200. In this case, a digit designator of #0121 specifies no parity, the starting byte (when writing) = leftmost byte, the number of digits to read = 3, and the starting digit (when reading) = digit 1.



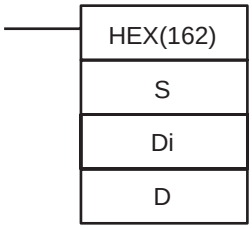
With CPU Units with unit version 4.0 of later, there are instructions to convert 4, 8, and 16 digits of numeric data to ASCII (STR4(524), STR8(527), and STR16(528)).

3-12-11 ASCII TO HEX: HEX(162)

Purpose

Converts up to 4 bytes of ASCII data in the source word to their hexadecimal equivalents and writes these digits in the specified destination word.

Ladder Symbol



S: First source word

Di: Digit designator

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	HEX(162)
	Executed Once for Upward Differentiation	@HEX(162)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

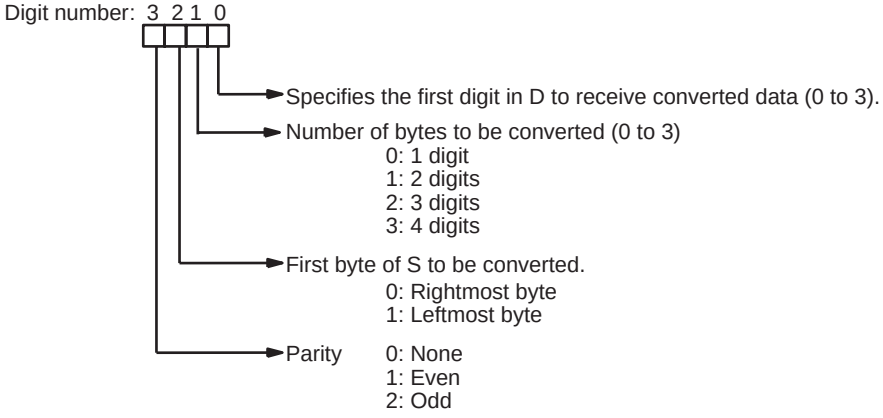
Operands

S: First Source Word

The contents of the source words are treated as ASCII data. Up to three source words can be used. (Three source words will be required if 4 bytes are being converted and the leftmost byte is selected as the first byte in S.) The source words must be in the same data area.

Di: Digit Designator

The digit designator specifies various parameters for the conversion, as shown in the following diagram.



D: Destination word

The converted hexadecimal digits are written into D from right to left, beginning with the specified first digit. Any digits in the destination word that are not overwritten with the converted data will be left unchanged.

Operand Specifications

Area	S	Di	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	---
Data Registers	---	DR0 to DR15	---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

HEX(162) treats the contents of the source word(s) as ASCII data representing hexadecimal digits (0 to 9 and A to F), converts the specified number of bytes to hexadecimal, and writes the hexadecimal data to the destination word beginning at the specified digit.

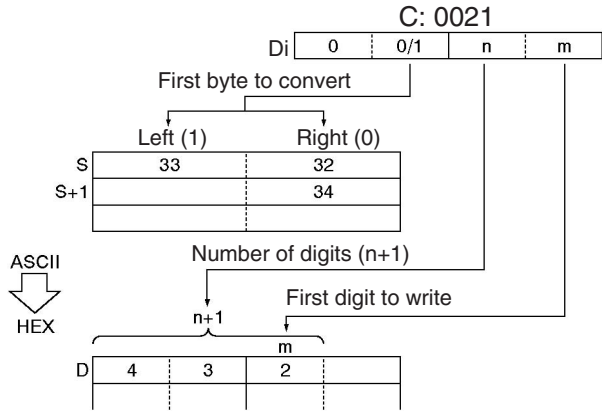
An error will occur if the source words contain data which is not an ASCII equivalent of hexadecimal digits. The following table shows hexadecimal digits and their ASCII equivalents (excluding parity bits).

Flags

Hexadecimal digits (4 bits)	ASCII equivalent (2 hexadecimal digits)
0 to 9	30 to 39
A to F	41 to 46

Note Refer to *Appendix A* in the *CS/CJ-series Programming Consoles Operation Manual* (W341) for a table of extended ASCII characters.

The following diagram shows the basic operation of HEX(162) with Di=0021.



Parity

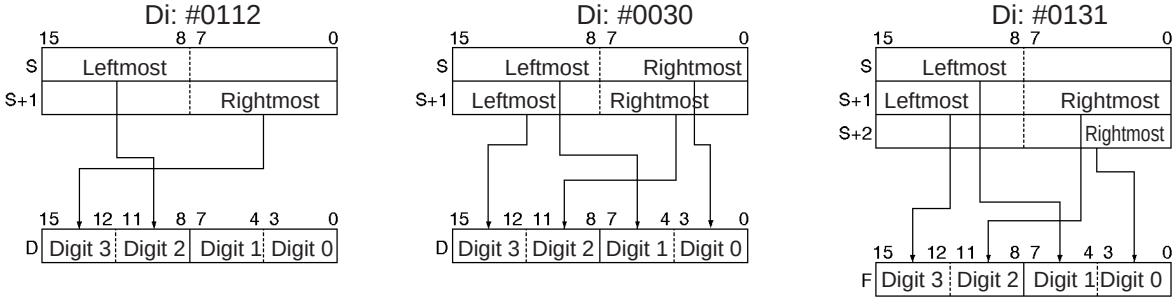
It is possible to specify the parity of the ASCII data for use in error control during data transmissions. The leftmost bit in each byte is the parity bit. With no parity the parity bit should always be zero, with even parity the status of the parity bit should result in an even number of ON bits, and with odd parity the status of the parity bit should result in an odd number of ON bits.

The following table shows the operation of HEX(162) for each parity setting.

Parity setting (leftmost digit of Di)	Operation of HEX(162)
No parity (0)	HEX(162) will be executed only when the parity bit in each byte is 0. An error will occur if a parity bit is non-zero.
Even parity (1)	HEX(162) will be executed only when there is an even number of ON bits in each byte. An error will occur if a byte has an odd number of ON bits.
Odd parity (2)	HEX(162) will be executed only when there is an odd number of ON bits in each byte. An error will occur if a byte has an even number of ON bits.

Examples of Di

When two or more bytes are being converted, HEX(162) will write the converted digits to the destination word from right to left and will wrap around to the rightmost digit if necessary. The following diagram shows some example values for Di and the conversions that they produce.



Flags

Name	Label	Operation
Error Flag	ER	ON if there is a parity error in the ASCII data. ON if the ASCII data in the source words is not equivalent to hexadecimal digits ON if the content of Di is not within the specified ranges. OFF in all other cases.

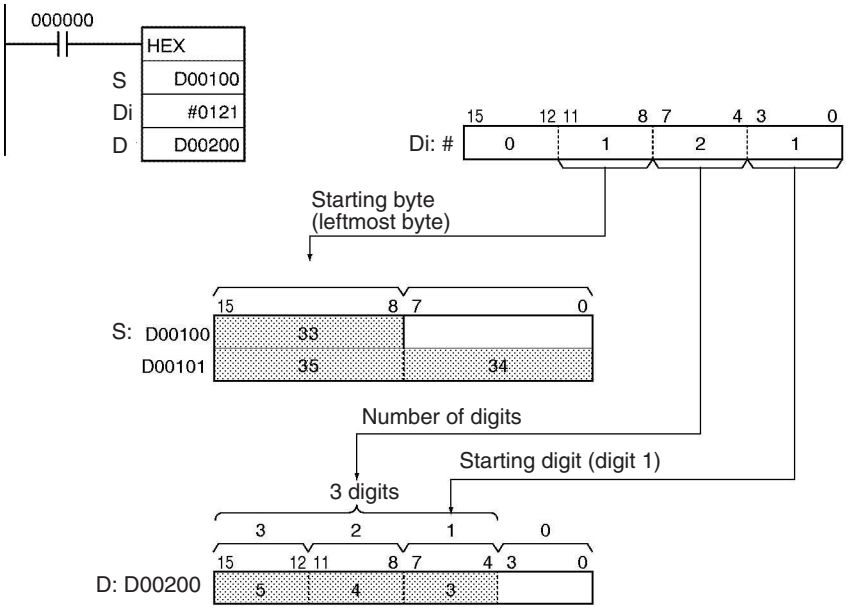
Precautions

An error will occur and the Error Flag will be turned ON if there is a parity error in the ASCII data, the ASCII data in the source words is not equivalent to hexadecimal digits, or the content of Di is not within the specified ranges.

Examples

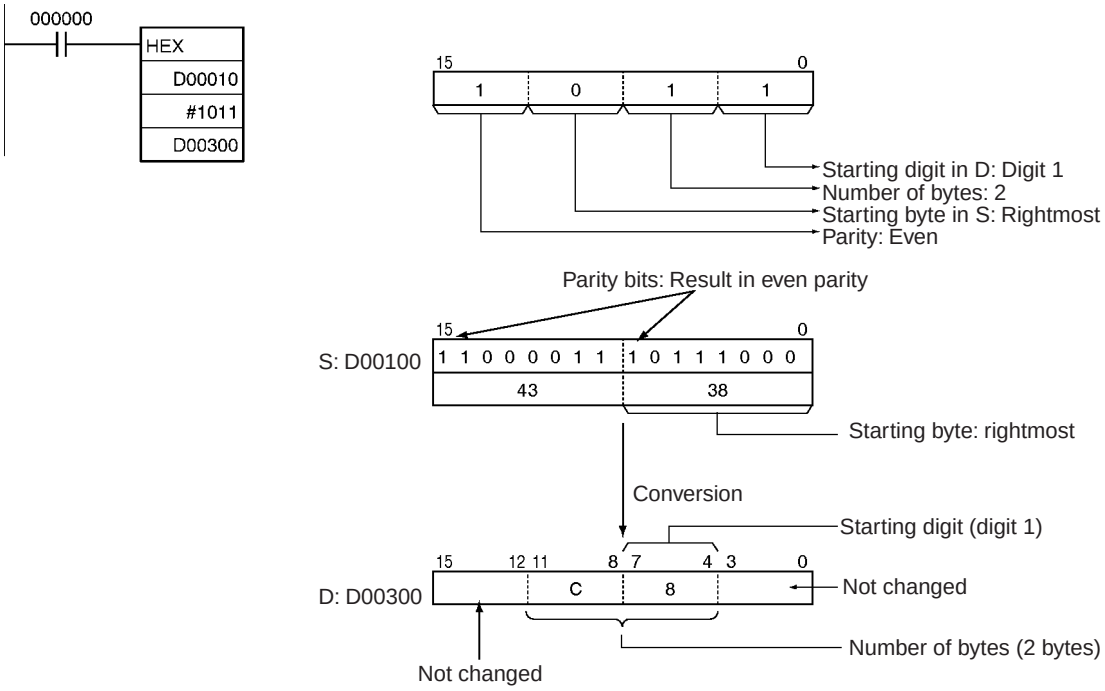
When CIO 000000 is ON in the following example, HEX(162) converts the ASCII data in D00100 and D00101 according to the settings of the digit designator. (Di=#0121 specifies no parity, the starting byte (when reading) = leftmost byte, the number of bytes to read = 3, and the starting digit (when writing) = digit 1.)

HEX(162) converts three bytes of ASCII data (3 characters) beginning with the leftmost byte of D00100 into their hexadecimal equivalents and writes this data to D00200 beginning with digit 1.



When CIO 000000 is ON in the following example, HEX(162) converts the ASCII data in D00010 beginning with the rightmost byte and writes the hexadecimal equivalents in D00300 beginning with digit 1.

The digit designator setting of #1011 specifies even parity, the starting byte (when reading) = rightmost byte, the number of bytes to read = 2, and the starting digit (when writing) = digit 1.)



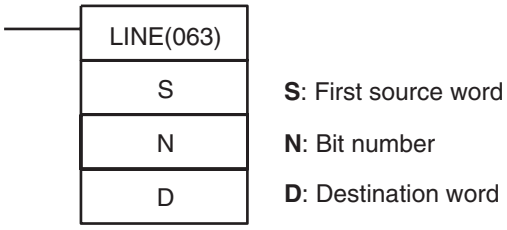
With CPU Units with unit version 4.0 of later, there are instructions to convert ASCII to 4, 8, and 16 digits of numeric data (NUM4(517), NUM8(520), and NUM16(522)).

3-12-12 COLUMN TO LINE: LINE(063)

Purpose

Converts a column of bits from a 16-word range (the same bit number in 16 consecutive words) to the 16 bits of the destination word.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	LINE(063)
	Executed Once for Upward Differentiation	@LINE(063)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

S: First Source Word

Specifies the first source word. S and S+15 must be in the same data area.

N: Bit Number

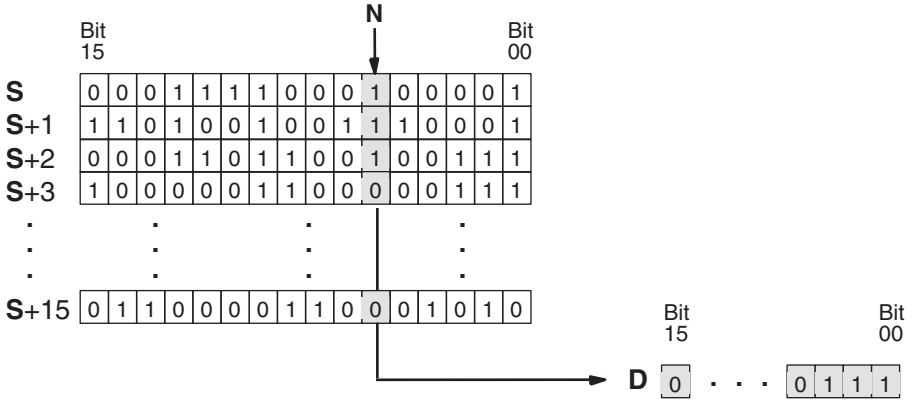
Specifies the bit number (0000 to 000F or &0 to &15) to be copied from the source words.

Operand Specifications

Area	S	N	D
CIO Area	CIO 0000 to CIO 6128	CIO 0000 to CIO 6143	
Work Area	W000 to W496	W000 to W511	
Holding Bit Area	H000 to H496	H000 to H511	
Auxiliary Bit Area	A000 to A944	A000 to A959	A448 to A959
Timer Area	T0000 to T4080	T0000 to T4095	
Counter Area	C0000 to C4080	C0000 to C4095	
DM Area	D00000 to D32752	D00000 to D32767	
EM Area without bank	E00000 to E32752	E00000 to E32767	
EM Area with bank	En_00000 to En_32752 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0000 to 000F (binary) or &0 to &15	---
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

LINE(063) copies the 16 bits with bit number N from the 16-word range S to S+15 to the destination word D. Bit N of S+m is copied to bit m of D, i.e., bit N of S is copied to bit 00 of D and bit N of S+15 is copied to bit 15 of D.

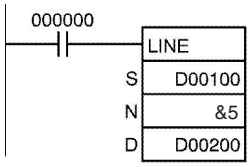


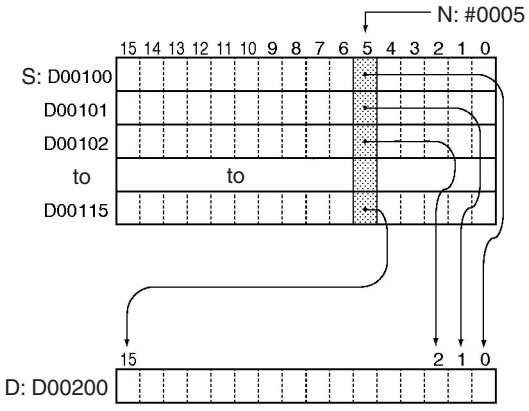
Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0000 to 000F. OFF in all other cases.
Equals Flag	=	ON if D is 0000 after execution. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, LINE(063) copies bit 5 from D00100 to D00115 to the 16 bits in D00200.



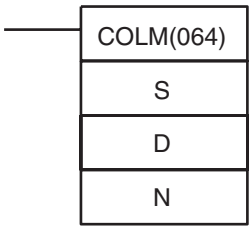


3-12-13 LINE TO COLUMN: COLM(064)

Purpose

Converts the 16 bits of the source word to a column of bits in a 16-word range of destination words (the same bit number in 16 consecutive words).

Ladder Symbol



S: Source word

D: First destination word

N: Bit number

Variations

Variations		
	Executed Each Cycle for ON Condition	COLM(064)
	Executed Once for Upward Differentiation	@COLM(064)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

D: First Destination Word

Specifies the first destination word. D and D+15 must be in the same data area.

N: Bit Number

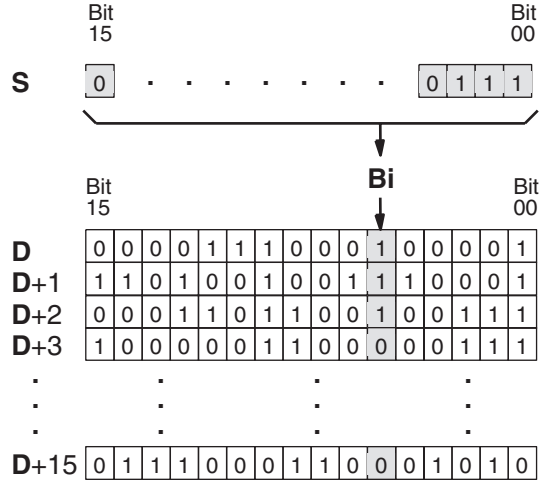
Specifies the bit number (0000 to 000F or &0 to &15) to be overwritten by the source word.

Operand Specifications

Area	S	D	N
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6128	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W496	W000 to W511
Holding Bit Area	H000 to H511	H000 to H496	H000 to H511
Auxiliary Bit Area	A000 to A959	A448 to A944	A000 to A959
Timer Area	T0000 to T4095	T0000 to T4080	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4080	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32752	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32752	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32752 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	#0000 to #000F (binary) or &0 to &15
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

COLM(064) copies the 16 bits from S to the 16 bits with bit number N in the 16-word range D to D+15. Bit m of S is copied to bit N of D+m, i.e., bit 00 of S is copied to bit N of D and bit 15 of S is copied to bit N of D+15.

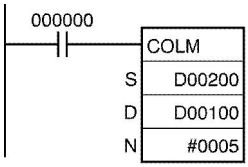


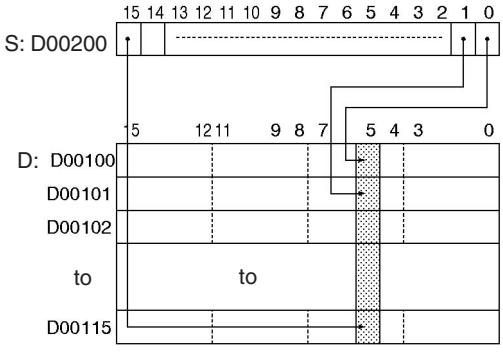
Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0000 to 000F. OFF in all other cases.
Equals Flag	=	ON if bit N is 0 in all 16 words D to D+15 after execution. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, COLM(064) copies the 16 bits in D00200 (bits 00 through 15) to bit 5 in D00100 through D00115.



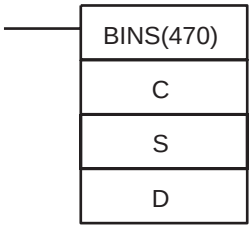


3-12-14 SIGNED BCD TO BINARY: BINS(470)

Purpose

Converts one word of signed BCD data to one word of signed binary data.

Ladder Symbol



C: Control word

S: Source word

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	BINS(470)
	Executed Once for Upward Differentiation	@BINS(470)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

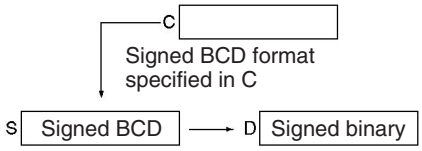
Operand Specifications

Area	C	S	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #0003 (binary)	---	
Data Registers	DR0 to DR15		

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

Description

BINS(470) converts signed BCD data to signed binary data. First the signed BCD data format and range in word S are checked against the setting in the control word (C). If the source data is correct, the signed BCD data in S is converted to signed binary and output to D. If the source data is incorrect, the Error Flag will be turned ON and the instruction will not be executed.



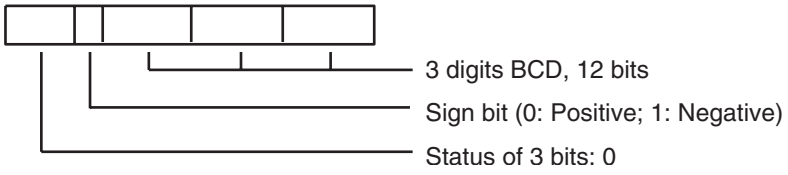
When the converted data is negative, it will be output as the 2's complement and the Negative Flag be will turned ON. NEG(160) can be used to determine the absolute value of a negative signed binary number. Refer to *3-12-52'S COMPLEMENT: NEG(160)* for details.

A value of –0 in the source data will be treated as 0 and will not cause an error. Also, the status of bits 13 to 15 of S is not checked when C=0000.

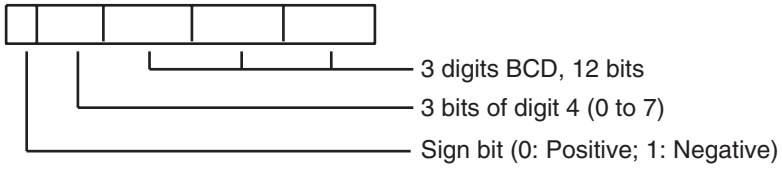
Note Some Special I/O Units output signed BCD data. Calculations using this data will normally be easier if it is first converted to signed binary data with BINS(470).

The control word specifies the signed BCD format as shown below.

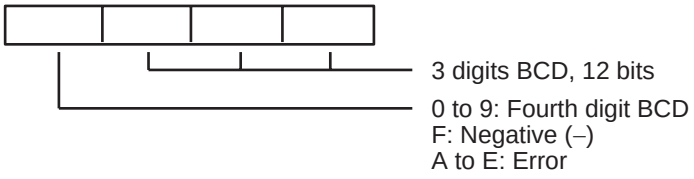
C = 0000 (Input Data Range: –999 to 999 BCD)



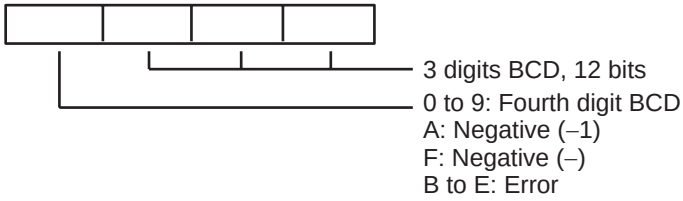
C = 0001 (Input Data Range: –7999 to 7999 BCD)



C = 0002 (Input Data Range: –999 to 9999 BCD)



C = 0003 (Input Data Range: –1999 to 9999 BCD)



The following table shows the possible BCD values for each signed BCD format and the corresponding signed binary values.

Setting	Signed BCD values	Signed binary values
C=0000	–999 to –1 and 0 to 999	FC19 to FFFF and 0000 to 03E7
C=0001	–7999 to –1 and 0 to 7999	E0C1 to FFFF and 0000 to 1F3F
C=0002	–999 to –1 and 0 to 9999	FC19 to FFFF and 0000 to 270F
C=0003	–1999 to –1 and 0 to 9999	F831 to FFFF and 0000 to 270F

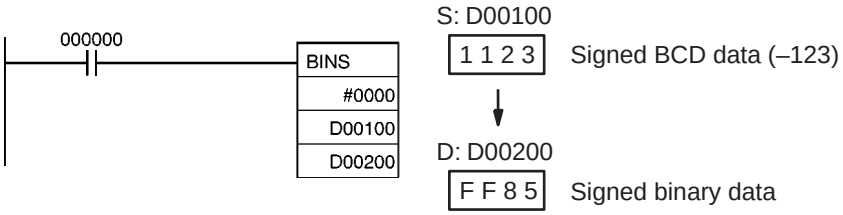
Flags

Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 0003. ON if C=0002 and the leftmost digit of S is A to E. ON if C=0003 and the leftmost digit of S is B to E. ON if the content of S is not BCD. OFF in all other cases.
Equals Flag	=	ON if D is 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of D is ON after execution. OFF in all other cases.

Examples

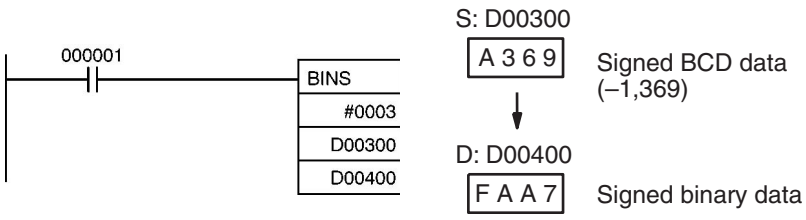
BCD Format 0 (C=#0000)

When CIO 000000 is ON in the following example, the signed BCD data format and range in D00100 are checked against the format specified in the control word (0000). The source data is correct, so the signed BCD data in D00100 is converted to signed binary and output to D00200.



BCD Format 0 (C=#0003)

When CIO 000001 is ON in the following example, the signed BCD data format and range in D00100 are checked against the format specified in the control word (0003). The source data is correct, so the signed BCD data in D00300 is converted to signed binary and output to D00400.

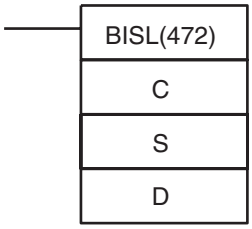


3-12-15 DOUBLE SIGNED BCD TO BINARY: BISL(472)

Purpose

Converts double signed BCD data to double signed binary data.

Ladder Symbol



C: Control word

S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	BISL(472)
	Executed Once for Upward Differentiation	@BISL(472)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

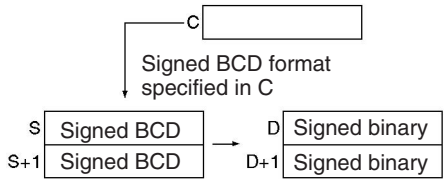
Operand Specifications

Area	C	S	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	
Work Area	W000 to W511	W000 to W510	
Holding Bit Area	H000 to H511	H000 to H510	
Auxiliary Bit Area	A000 to A959	A000 to A958	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094	
Counter Area	C0000 to C4095	C0000 to C4094	
DM Area	D00000 to D32767	D00000 to D32766	
EM Area without bank	E00000 to E32767	E00000 to E32766	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #0003 (binary)	---	
Data Registers	DR0 to DR15	---	

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

Description

BISL(472) converts the double signed BCD data in S+1 and S to double signed binary data and writes the result in D+1 and D. First the signed BCD data format and range in words S+1 and S are checked against the setting in the control word (C). If the source data is correct, the signed BCD data S+1 and S is converted to signed binary and output to D+1 and D. If the source data is incorrect, the Error Flag will be turned ON and the instruction will not be executed.



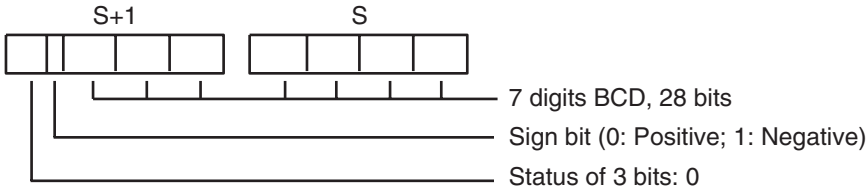
When the converted data is negative, it will be output as the 2's complement and the Negative Flag be will turned ON. NEGL(161) can be used to determine the absolute value of a negative double signed binary number. Refer to *3-12-6 DOUBLE 2'S COMPLEMENT: NEGL(161)* for details.

Values of –0 in the source data will be treated as 0 and will not cause an error. Also, the status of bits 13 to 15 of S+1 is not checked when C=0000.

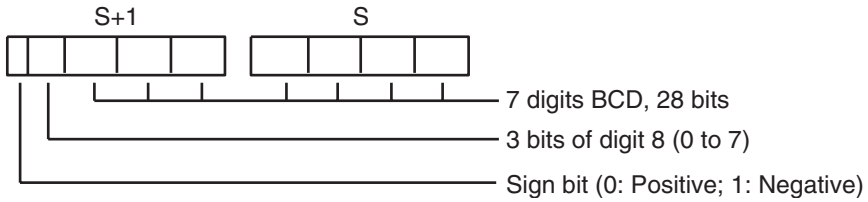
Note Some Special I/O Units output signed BCD data. Calculations using this data will normally be easier if it is first converted to signed binary data with BISL(472).

The control word specifies the signed BCD format as shown below.

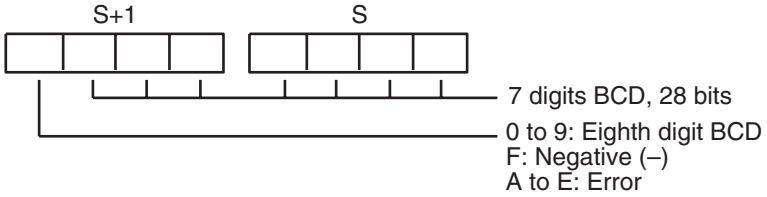
C = 0000 (Input Data Range: –999 9999 to 999 9999 BCD)



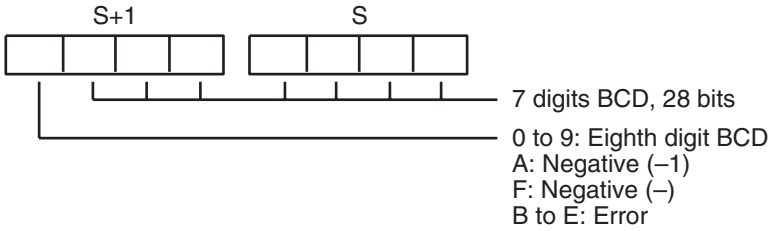
C = 0001 (Input Data Range: –7999 9999 to 7999 9999 BCD)



C = 0002 (Input Data Range: –999 9999 to 9999 9999 BCD)



C = 0003 (Input Data Range: –1999 9999 to 9999 9999 BCD)



The following table shows the possible BCD values for each signed BCD format and the corresponding signed binary values.

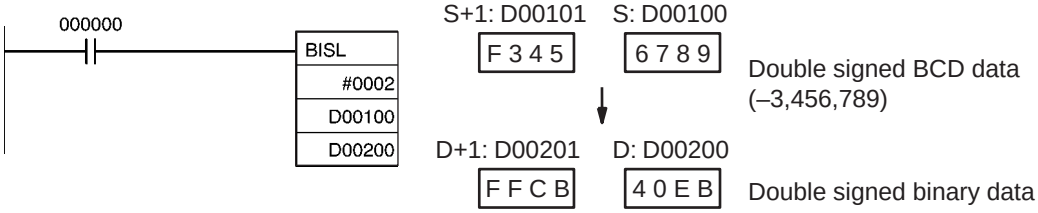
Setting	Signed BCD values	Signed binary values
C=0000	–999 9999 to –1	FF67 6981 to FFFF FFFF
	0 to 999 9999	0000 0000 to 0098 967F
C=0001	–7999 9999 to –1	FB3B 4C01 to FFFF FFFF
	0 to 7999 9999	0000 0000 to 04C4 B3FF
C=0002	–999 9999 to –1	FF67 6981 to FFFF FFFF
	0 to 9999 9999	0000 0000 to 05F5 E0FF
C=0003	–1999 9999 to –1	FECE D301 to FFFF FFFF
	0 to 9999 9999	0000 0000 to 05F5 E0FF

Flags

Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 0003. ON if C=0002 and the leftmost digit of S+1 is A to E. ON if C=0003 and the leftmost digit of S+1 is B to E. ON if the content of S+1 and S is not BCD. OFF in all other cases.
Equals Flag	=	ON if D+1 contains 0000 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of D+1 is ON after execution. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, the double signed BCD data format and range in D00101 and D00100 are checked against the format specified in the control word (0002). The source data is correct, so the double signed BCD data in D00101 and D00100 is converted to double signed binary and output to D00201 and D00200.

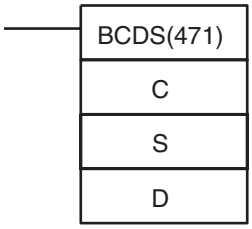


3-12-16 SIGNED BINARY TO BCD: BCDS(471)

Purpose

Converts one word of signed binary data to one word of signed BCD data.

Ladder Symbol



C: Control word

S: Source word

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	BCDS(471)
	Executed Once for Upward Differentiation	@BCDS(471)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand

C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

S: Source Word

Contains the signed binary data to be converted. The content of S must be within the valid range of the BCD format specified in C.

Setting	Allowed values for S
C=0000	FC19 to FFFF or 0000 to 03E7
C=0001	E0C1 to FFFF or 0000 to 1F3F
C=0002	FC19 to FFFF or 0000 to 270F
C=0003	F831 to FFFF or 0000 to 270F

D: Destination word

Contains the converted signed BCD data. See the description section below for an explanation of the BCD formats.

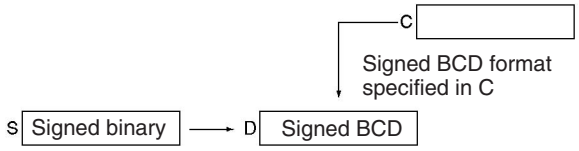
Operand Specifications

Area	C	S	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		

Area	C	S	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #0003 (binary)	---	
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to 1–2048 to +2047 ,IR5 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

BCDS(471) converts signed binary data to signed BCD data. First the signed binary data in word S is checked to verify that it is within the valid range for the signed BCD format specified in the control word (C). If the source data is correct, the signed binary data in S is converted to signed BCD and output to D. If the source data is incorrect, the Error Flag will be turned ON and the instruction will not be executed.

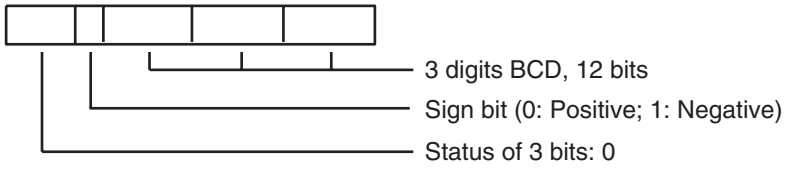


Note

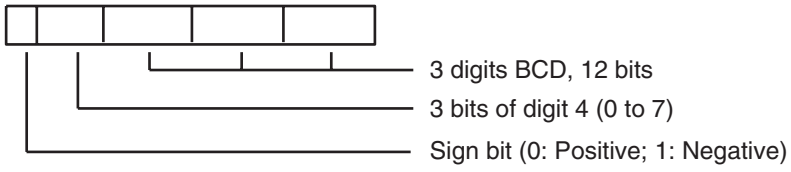
1. Values of –0 in the source data will be treated as 0 and will not cause an error.
2. Some Special I/O Units require signed BCD data inputs. BCDS(471) can be used to convert signed binary data for output to these Units.

The control word specifies the signed BCD format that will be used for the result, as shown below.

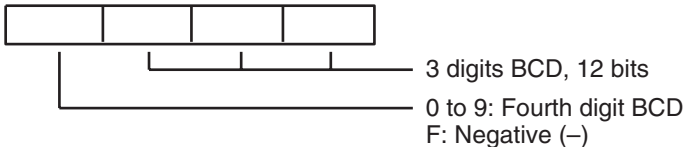
C = 0000 (Output Data Range: –999 to 999 BCD)



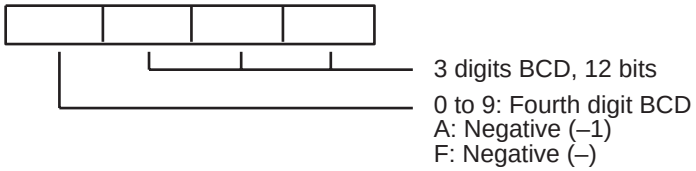
C = 0001 (Output Data Range: –7999 to 7999 BCD)



C = 0002 (Output Data Range: –999 to 9999 BCD)



C = 0003 (Output Data Range: –1999 to 9999 BCD)



The following table shows the possible signed binary values for each signed BCD format. An error will occur if the source data is not within the allowed range for the specified signed BCD format.

Setting	Signed binary values	Signed BCD values
C=0000	FC19 to FFFF and 0000 to 03E7	–999 to –1 and 0 to 999
C=0001	E0C1 to FFFF and 0000 to 1F3F	–7999 to –1 and 0 to 7999
C=0002	FC19 to FFFF and 0000 to 270F	–999 to –1 and 0 to 9999
C=0003	F831 to FFFF and 0000 to 270F	–1999 to –1 and 0 to 9999

Flags

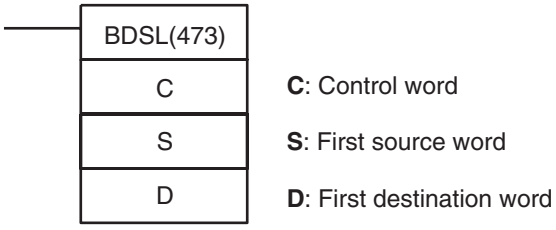
Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 0003. ON if C=0000 and the source data is not within the allowed ranges (FC19 to FFFF or 0000 to 03E7). ON if C=0001 and the source data is not within the allowed ranges (E0C1 to FFFF or 0000 to 1F3F). ON if C=0002 and the source data is not within the allowed ranges (FC19 to FFFF or 0000 to 270F). ON if C=0003 and the source data is not within the allowed ranges (F831 to FFFF or 0000 to 270F). OFF in all other cases.
Equals Flag	=	ON if D is 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if C=0000 or 0001 and the result's sign bit is ON after execution. ON if C=0002 and the leftmost digit of the result is F. ON if C=0003 and the leftmost digit of the result is A or F. OFF in all other cases.

3-12-17 DOUBLE SIGNED BINARY TO BCD: BDSL(473)

Purpose

Converts double signed binary data to double signed BCD data.

Ladder Symbol



Variations

Variations	Executed Each Cycle for ON Condition	BDSL(473)
	Executed Once for Upward Differentiation	@BDSL(473)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Operands

C: Control Word

Specifies the signed BCD format. C must be 0000 to 0003.

S: First Source Word

Source words S+1 and S contain the double signed binary data to be converted. Their content must be within the valid range of the BCD format specified in C.

Setting	Allowed values for S+1 and S
C=0000	FF67 6981 to FFFF FFFF or 0000 0000 to 0098 967F
C=0001	FB3B 4C01 to FFFF FFFF or 0000 0000 to 04C4 B3FF
C=0002	FF67 6981 to FFFF FFFF or 0000 0000 to 05F5 E0FF
C=0003	FECE D301 to FFFF FFFF or 0000 0000 to 05F5 E0FF

D: First destination word

Destination words D+1 and D contain the converted double signed BCD data. See the description section below for an explanation of the BCD formats.

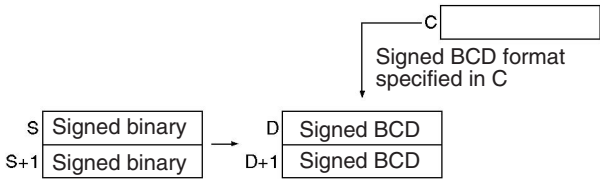
Operand Specifications

Area	C	S	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	
Work Area	W000 to W511	W000 to W510	
Holding Bit Area	H000 to H511	H000 to H510	
Auxiliary Bit Area	A000 to A959	A000 to A958	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094	
Counter Area	C0000 to C4095	C0000 to C4094	
DM Area	D00000 to D32767	D00000 to D32766	
EM Area without bank	E00000 to E32767	E00000 to E32766	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #0003 (binary)	---	
Data Registers	DR0 to DR15	---	

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

Description

BDSL(473) converts double signed binary data to double signed BCD data. First the double signed binary data in S+1 and S is checked to verify that it is within the valid range for the signed BCD format specified in the control word (C). If the source data is correct, the double signed binary data in S+1 and S is converted to double signed BCD and output to D+1 and D. If the source data is incorrect, the Error Flag will be turned ON and the instruction will not be executed.

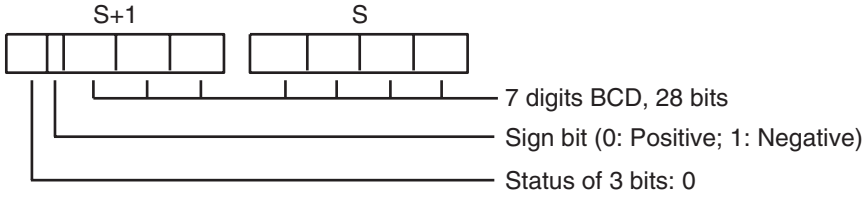


Note

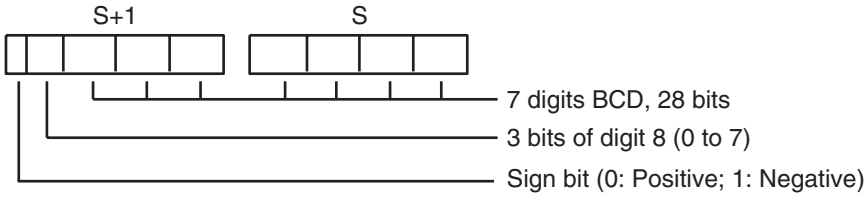
1. Values of –0 in the source data will be treated as 0 and will not cause an error.
2. Some Special I/O Units require signed BCD data inputs. BDSL(473) can be used to convert double signed binary data for output to these Units.

The control word specifies the signed BCD format that will be used for the result, as shown below.

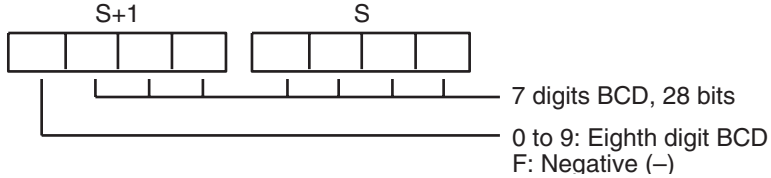
C = 0000 (Output Data Range: –999 9999 to 999 9999 BCD)



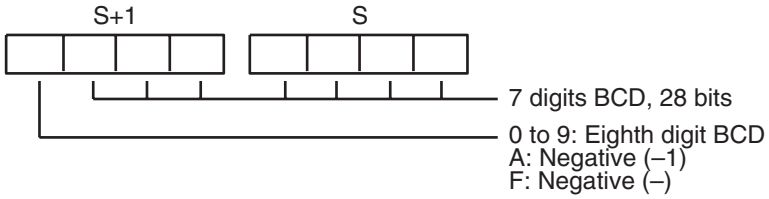
C = 0001 (Output Data Range: –7999 9999 to 7999 9999 BCD)



C = 0002 (Output Data Range: –999 9999 to 9999 9999 BCD)



C = 0003 (Output Data Range: –1999 9999 to 9999 9999 BCD)



The following table shows the possible double signed binary values for each signed BCD format. An error will occur if the source data is not within the allowed range for the specified signed BCD format.

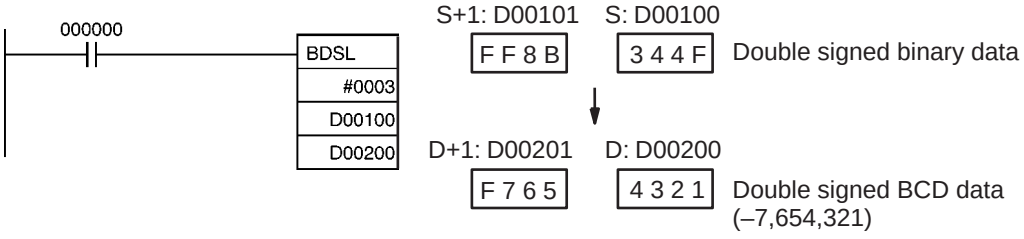
Setting	Signed binary values	Signed BCD values
C=0000	FF67 6981 to FFFF FFFF	–999 9999 to –1
	0000 0000 to 0098 967F	0 to 999 9999
C=0001	FB3B 4C01 to FFFF FFFF	–7999 9999 to –1
	0000 0000 to 04C4 B3FF	0 to 7999 9999
C=0002	FF67 6981 to FFFF FFFF	–999 9999 to –1
	0000 0000 to 05F5 E0FF	0 to 9999 9999
C=0003	FECE D301 to FFFF FFFF	–1999 9999 to –1
	0000 0000 to 05F5 E0FF	0 to 9999 9999

Flags

Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 0003. ON if C=0000 and the source data is not within the range: FF67 6981 to FFFF FFFF or 0000 0000 to 0098 967F. ON if C=0001 and the source data is not within the range: FB3B 4C01 to FFFF FFFF or 0000 0000 to 04C4 B3FF. ON if C=0002 and the source data is not within the range: FF67 6981 to FFFF FFFF or 0000 0000 to 05F5 E0FF. ON if C=0003 and the source data is not within the range: FECE D301 to FFFF FFFF or 0000 0000 to 05F5 E0FF. OFF in all other cases.
Equals Flag	=	ON if D is 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if C=0000 or 0001 and the result's sign bit is ON after execution. ON if C=0002 and the leftmost digit of the result is F. ON if C=0003 and the leftmost digit of the result is A or F. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, the double signed binary data in D00101 and D00100 are checked against the format specified in the control word (0003). The source data is correct, so the double signed binary data in D00101 and D00100 is converted to double signed BCD and output to D00201 and D00200.



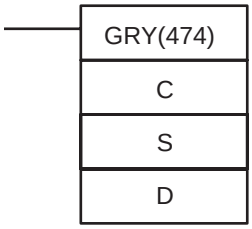
3-12-18 GRAY CODE CONVERT: GRY(474)

Purpose

Converts the gray binary code in a specified word to standard binary data, BCD data, or an angle at the specified resolution.

This instruction is supported by only CS/CJ-series CPU Unit Ver. 2.0 or later (including CS1-H, CJ1-H, and CJ1M CPU Units from lot number 030201 or later).

Ladder Symbol



C: First control word

S: Source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	GRY(474)
	Executed Once for Upward Differentiation	@GRY(474)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

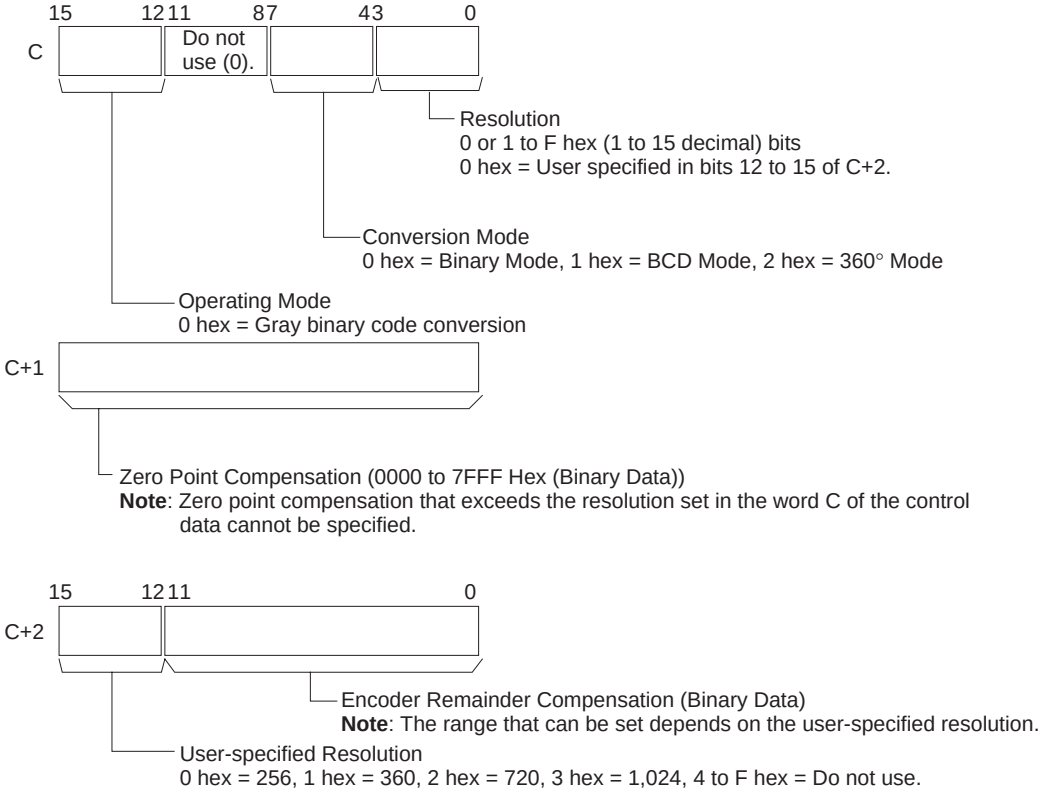
Applicable Program Areas

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

Operands

C: Control Word

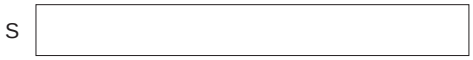
Specifies the parameters for the conversion as shown below.



Note: The above setting is valid when the resolution is set to 0 hex in bits 00 to 03 of C.

S: Source Word

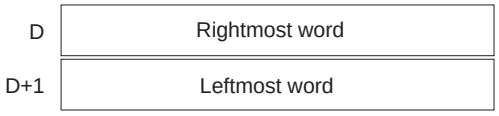
Contains the gray binary code to be converted. The range must be within the number of bits determined by the resolution specified in bits 00 to 03 of C. All bits outside of the number of bits for the specified resolution will be ignored. For example, if the specified resolution is 08 hex and S contains FFFF hex, the gray binary code will be taken as 00FF hex.



D: First destination word

Destination words D+1 and D contain the results of converting the gray binary code at the resolution specified in bits 00 to 03 of the control data word C and the conversion mode specified in bits 04 to 07 of the control data word C. The leftmost word is output to D+1 and the rightmost word is output to D. The ranges of data that are output are as follows:

Binary Mode: 0000 0000 to 0000 7FFF hex
BCD Mode: 0000 0000 to 0003 2767
360° Mode: 0000 0000 to 0000 3599
(0.0° to 359.9° in 0.1° increments, BCD)



Operand Specifications

Area	C	S	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142
Work Area	W000 to W510	W000 to W511	W000 to W510
Holding Bit Area	H000 to H510	H000 to H511	H000 to H510
Auxiliary Bit Area	A000 to A958	A000 to A959	A448 to A958
Timer Area	T0000 to T4094	T0000 to T4095	T0000 to T4094
Counter Area	C0000 to C4094	C0000 to C4095	C0000 to C4094
DM Area	D00000 to D32766	D00000 to D32767	D00000 to D32766
EM Area without bank	E00000 to E32766	E00000 to E32767	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0000 to #FFFF (binary)	---
Data Registers	---	DR0 to DR15	---

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

Description

GRY(474) converts the gray binary code in the word specified in S at the resolution specified in C using one of the following conversion modes (binary, BCD, or 360°), also specified in C, and places the results in D and D+1.

Conversion mode	Function
Binary Mode	Gray binary code is converted to binary data between 0000 0000 and 0000 7FFF hex. Zero point offset and remainder compensation is applied and then the result is output to D and D+1.
BCD Mode	Gray binary code is converted to BCD data. Zero point offset and remainder compensation is applied, the data is converted to BCD between 0000 0000 and 0003 2767, and then the result is output to D and D+1.
360° Mode	Gray binary code is converted to BCD data. Zero point offset and remainder compensation is applied, the data is converted to an angle between 0000 0000 and 0000 3599 (0.0° to 359.9° in 0.1° increments), and then the result is output to D and D+1.

Note

1. GRY(474) is normally used when inputting, through a DC Input Unit, a parallel signal (2^n) from an absolute encoder that outputs a gray binary code.
2. If the word specified for S is allocated to an Input Unit, the input data converted by GRY(474) will be for the gray binary code from the previous CPU Unit cycle, i.e., it will be one cycle time old.

Restrictions

The following restrictions apply to GRY(474).

■ **Restrictions on the CPU Unit**

GRY(474) can be used only for the following models of CPU Unit and only for CPU Units manufactured on or after 1 February 2003 (lot number 030201 or later, including CPU Unit Ver. 2.0 or later).

- CJ1H-CPU□□H-R
- CJ1M-CPU□□
- CJ1G-CPU□□H
- CJ1H-CPU□□H
- CS1G-CPU□□H
- CS1H-CPU□□H
- CS1D-CPU□□S

The manufacturing date can be confirmed using the lot number given on the side or bottom of the CPU Unit. Lot numbers indicate the manufacturing date as follows:

YYMMDD nnnn

YY = Rightmost two digits of the year, MM = Month as a numeric value, DD = Day of month, nnnn = Serial number

Note If GRY(474) is transferred to a CPU Unit that does not support it and the program is read from a Programming Console, "?" will be displayed for GRY(474) to indicate an illegal instruction. If GRY(474) is executed with an ON input

condition in a CPU Unit that does not support it, an error will occur and program execution will stop.

■ **Restrictions on the CX-Programmer**

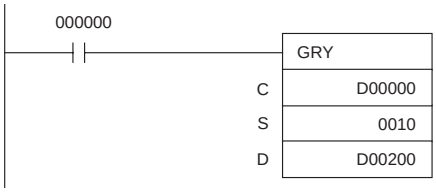
GRY(474) can be used only with CX-Programmer version 3.2 or later.

Flags

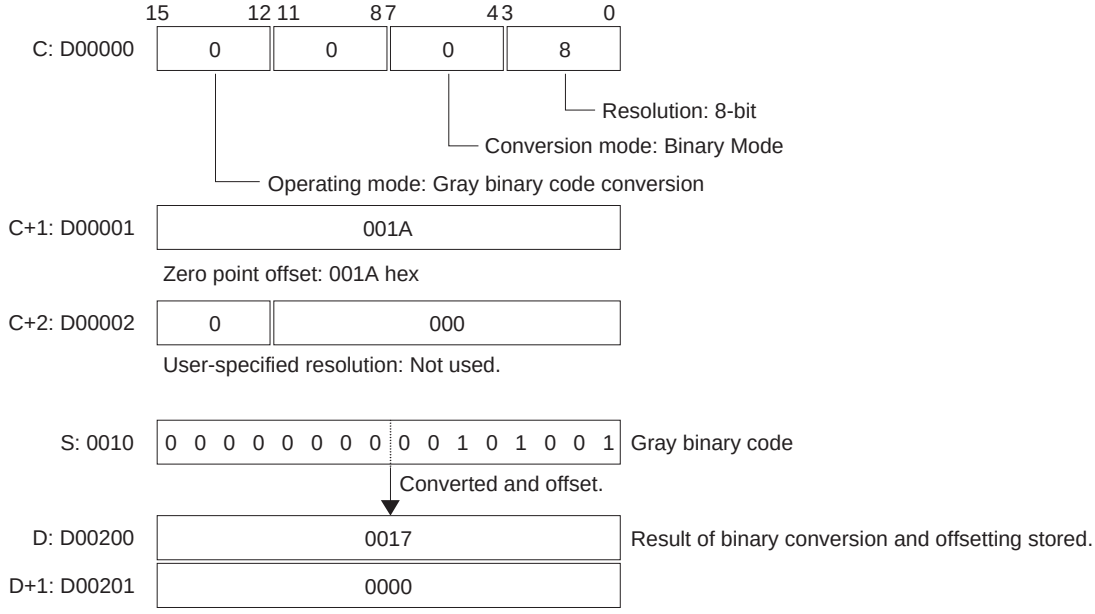
Name	Label	Operation
Error Flag	ER	ON if bits 12 to 15 of C are not 0 hex (operating mode = gray binary code conversion). ON if the zero point offset in C+1 is not within the specified resolution (including user-specified resolutions). ON if bits 04 to 07 of C are not 0 hex (= Binary Mode), 1 hex (= BCD Mode), or 2 hex (= 360° Mode). ON if the specified encoder remainder compensation exceeds the set user-specified resolution when bits 00 to 03 of C are 0 hex (= user-specified resolution). ON if the converted binary value is less than the encoder remainder compensation when bits 00 to 03 of C are 0 hex (= user-specified resolution). ON if the converted binary value is less than the resolution when bits 00 to 03 of C are 0 hex (= user-specified resolution). OFF in all other cases.
Equals Flag	=	OFF in all cases.
Negative Flag	N	OFF in all cases.

Examples

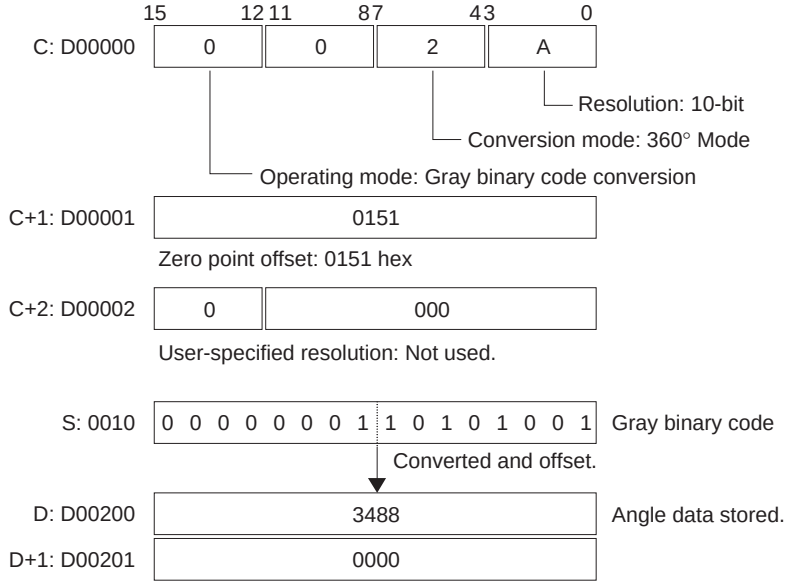
When CIO 000000 is ON in the following example, the gray binary code in CIO 0010 is converted according to the settings in the control data in D00000 to D00002 and the result is output to D00200 and D00201.



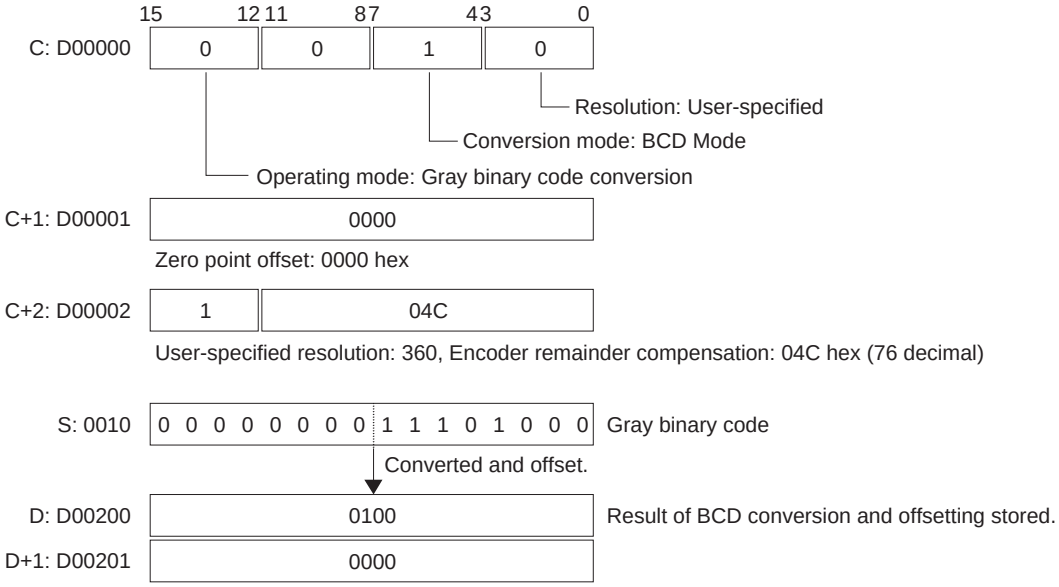
■ Example 1: Converting to Binary Data with an 8-bit Resolution and Zero Point Offset of 001A Hex



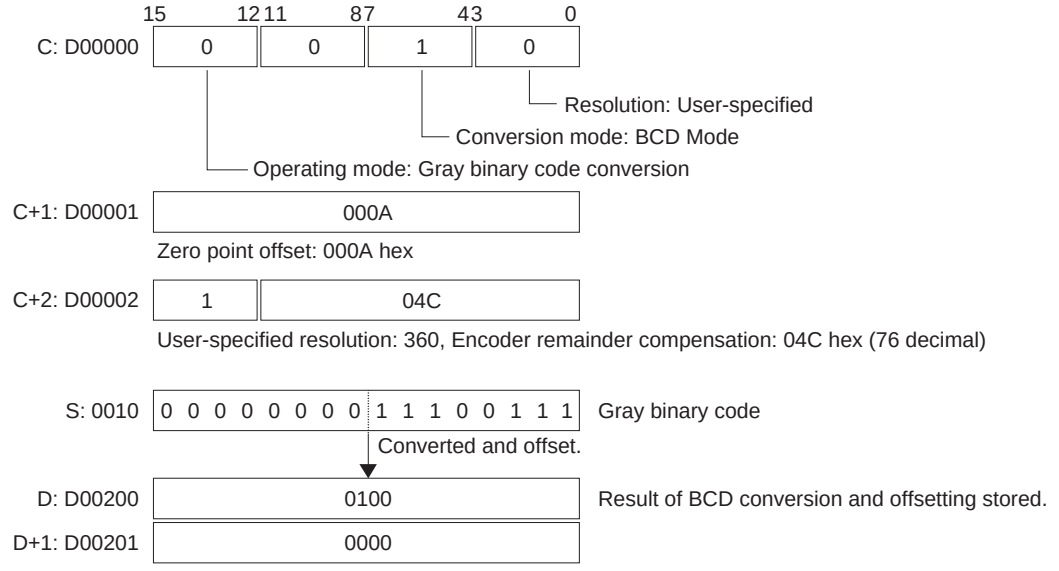
■ Example 2: Converting to Angle Data with a 10-bit Resolution and Zero Point Offset of 0151 Hex



■ Example 3: Converting to BCD Data with for an OMRON E6C2-AG5C Absolute Encoder (Resolution: 360/rotation, Encoder Remainder Compensation: 76) and Zero Point Offset of 0000 Hex



■ Example 4: Converting to BCD Data with for an OMRON E6C2-AG5C Absolute Encoder (Resolution: 360/rotation, Encoder Remainder Compensation: 76) and Zero Point Offset of 000A Hex



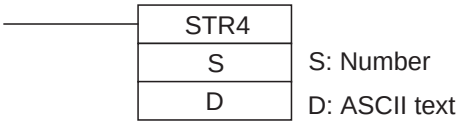
3-12-19 FOUR-DIGIT NUMBER TO ASCII: STR4(601)

Purpose

Converts a 4-digit hexadecimal number (#0000 to #FFFF) to ASCII data (4 characters).

This instruction is supported by CS/CJ-series CPU Units with unit version 4.0 or later only.

Ladder Symbol



Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	STR4(601)
	Executed Once for Upward Differentiation	@STR4(601)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

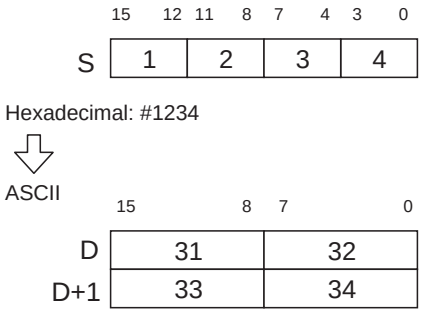
Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142
Work Area	W000 to W511	W000 to W510
Holding Bit Area	H000 to H511	H000 to H510
Auxiliary Bit Area	A000 to A959	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094
Counter Area	C0000 to C4095	C0000 to C4094
DM Area	D00000 to D32767	D00000 to D32766
EM Area without bank	E00000 to E32767	E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF	---
Data Registers	---	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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.



Note If the source data is 0, the Equals Flag will turn ON.
If the leftmost bit of the source data is 1, the Negative Flag will turn ON.

Restrictions

The following restrictions apply to STR4(601).

■ **Restrictions on the CPU Unit**

STR4(601) can be used in CPU Units with unit version 4.0 or later only.

■ **Restrictions on the CX-Programmer**

STR4(601) can be used in CX-Programmer version 7 or higher only.

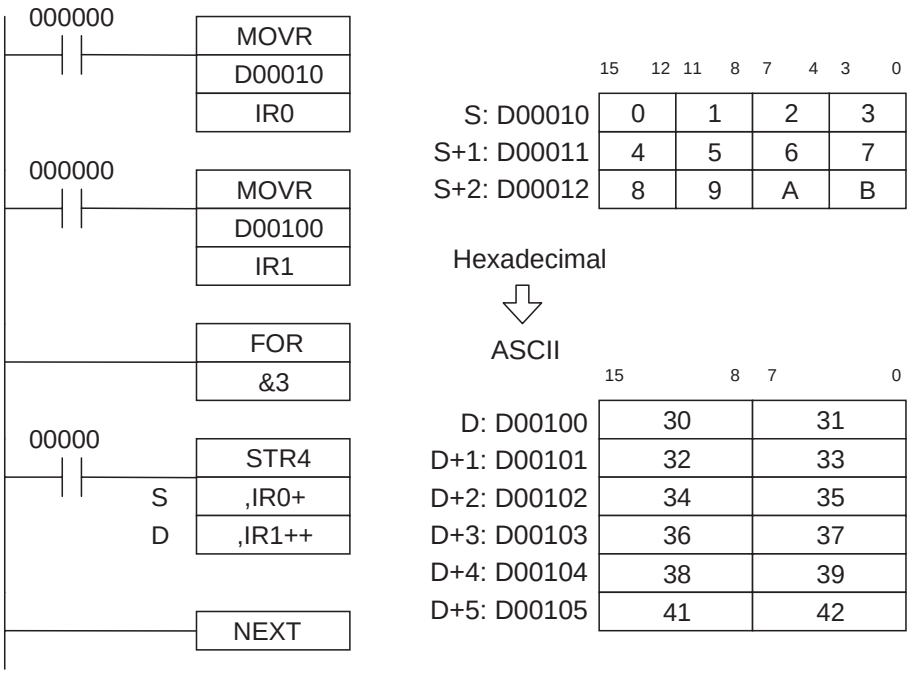
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the source data is 1. OFF in all other cases.

Examples

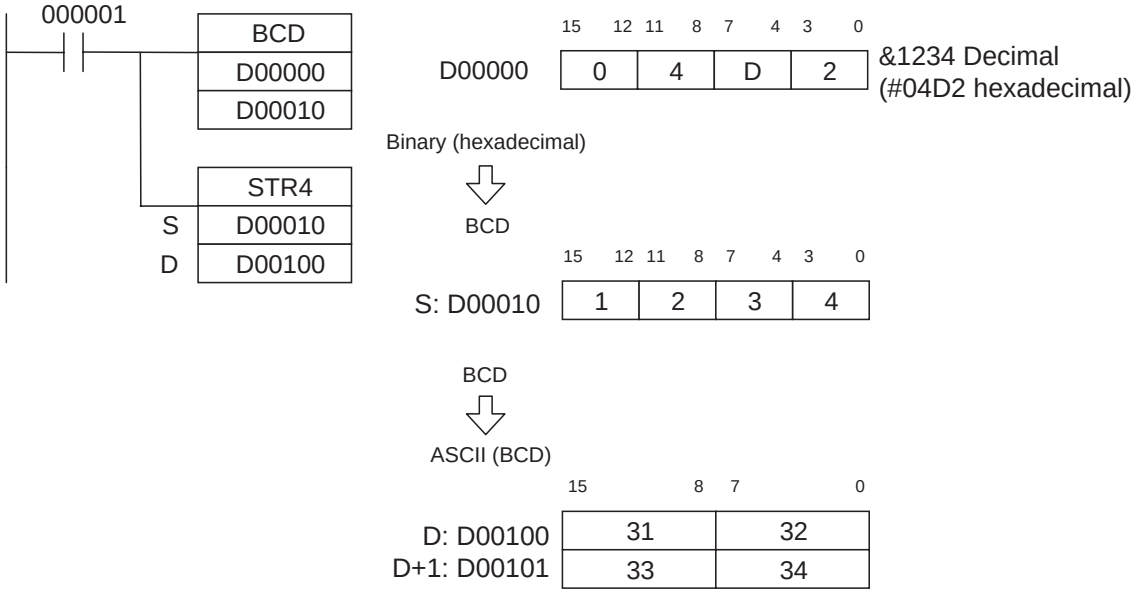
■ **Example 1: Converting 3 Words of Numerical Data to ASCII Data**

When CIO 000000 is ON in the following example, the 3 words of numerical data starting at D00010 are converted, one word at a time, to ASCII data. The converted ASCII data is stored in the DM Area starting at D00100.



■ Example 2: Converting Hexadecimal Data to ASCII Data in BCD Format

When CIO 000001 is ON in the following example, the source data in D00000 (&1234 in decimal) is converted to BCD data and the result is stored temporarily in D00010. Next, the BCD data is converted to ASCII data and the result is output to D00100 and D00101.



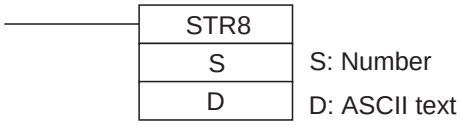
3-12-20 EIGHT-DIGIT NUMBER TO ASCII: STR8(602)

Purpose

Converts an 8-digit hexadecimal number (#0000 0000 to #FFFF FFFF) to ASCII data (8 characters).

This instruction is supported by CS/CJ-series CPU Units with unit version 4.0 or later only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	STR8(602)
	Executed Once for Upward Differentiation	@STR8(602)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

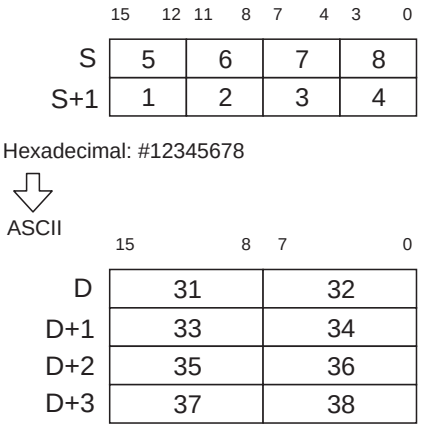
Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6140
Work Area	W000 to W510	W000 to W508
Holding Bit Area	H000 to H510	H000 to H508
Auxiliary Bit Area	A448 to A958	A448 to A956
Timer Area	T0000 to T4094	T0000 to T4092
Counter Area	C0000 to C4094	C0000 to C4092
DM Area	D00000 to D32766	D00000 to D32764
EM Area without bank	E00000 to E32766	E00000 to E32764
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32764 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 0000 to #FFFF FFFF	---
Data Registers	---	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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.



Note If the source data is 0, the Equals Flag will turn ON.
If the leftmost bit of the source data is 1, the Negative Flag will turn ON.

Restrictions

The following restrictions apply to STR8(602).

■ **Restrictions on the CPU Unit**

STR8(602) can be used in CPU Units with unit version 4.0 or later only.

■ **Restrictions on the CX-Programmer**

STR8(602) can be used in CX-Programmer version 7 or higher only.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the source data is 1. OFF in all other cases.

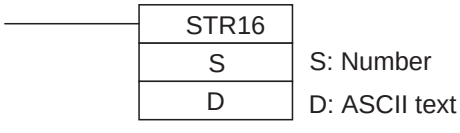
3-12-21 SIXTEEN-DIGIT NUMBER TO ASCII: STR16(603)

Purpose

Converts a 16-digit hexadecimal number (#0000 0000 0000 0000 to #FFFF FFFF FFFF FFFF) to ASCII data (16 characters).

This instruction is supported by CS/CJ-series CPU Units with unit version 4.0 or later only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	STR16(603)
	Executed Once for Upward Differentiation	@STR16(603)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	CIO 0000 to CIO 6136
Work Area	W000 to W508	W000 to W504
Holding Bit Area	H000 to H508	H000 to H504
Auxiliary Bit Area	A448 to A956	A448 to A952
Timer Area	T0000 to T4092	T0000 to T4088
Counter Area	C0000 to C4092	C0000 to C4088
DM Area	D00000 to D32764	D00000 to D32760
EM Area without bank	E00000 to E32764	E00000 to E32760
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	En_00000 to En_32760 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	---
Data Registers	---	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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

Note If the source data is 0, the Equals Flag will turn ON.
If the leftmost bit of the source data is 1, the Negative Flag will turn ON.

Restrictions

The following restrictions apply to STR16(603).

■ **Restrictions on the CPU Unit**

STR16(603) can be used in CPU Units with unit version 4.0 or later only.

■ **Restrictions on the CX-Programmer**

STR16(603) can be used in CX-Programmer version 7 or higher only.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the source data is 1. OFF in all other cases.

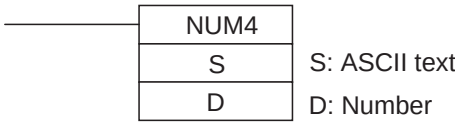
3-12-22 ASCII TO FOUR-DIGIT NUMBER: NUM4(604)

Purpose

Converts 4 characters of ASCII data to a 4-digit hexadecimal number.

This instruction is supported by CS/CJ-series CPU Units with unit version 4.0 or later only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	NUM4(604)
	Executed Once for Upward Differentiation	@NUM4(604)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operand Specifications

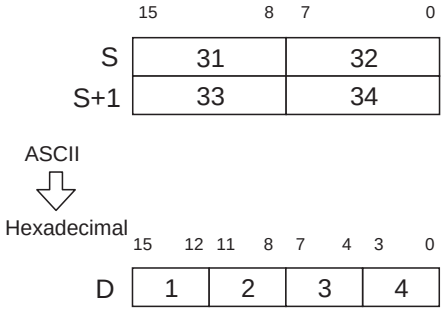
Area	S	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W510	W000 to W511
Holding Bit Area	H000 to H510	H000 to H511
Auxiliary Bit Area	A448 to A958	A000 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)

Area	S	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	---
Data Registers	---	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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.



Note If the numerical data is 0, the Equals Flag will turn ON.
If the leftmost bit of the numerical data is 1, the Negative Flag will turn ON.

Restrictions

The following restrictions apply to NUM4(604).

■ **Restrictions on the CPU Unit**

NUM4(604) can be used in CPU Units with unit version 4.0 or later only.

■ **Restrictions on the CX-Programmer**

NUM4(604) can be used in CX-Programmer version 7 or higher only.

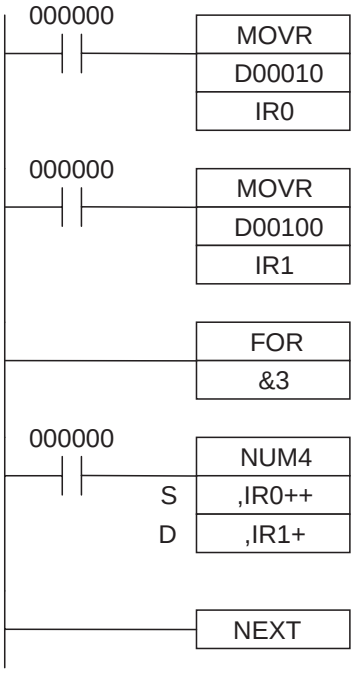
Flags

Name	Label	Operation
Error Flag	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.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the source data is 1. OFF in all other cases.

Examples

■ Example 1: Converting 3 Sets of 4 ASCII Characters to the Equivalent Hexadecimal Digits

When CIO 000000 is ON in the following example, the 6 words of ASCII data starting at D00010 are converted, two words at a time, to numerical data. The converted numerical data is stored in the DM Area starting at D00100.



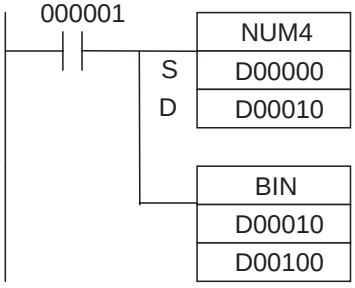
	15–8	7–0
S: D00010	31	32
S+1: D00011	41	42
S+2: D00012	38	39
S+3: D00013	45	46
S+4: D00014	30	30
S+5: D00015	30	30

ASCII ⇩ Hexadecimal

	15–12	11–8	7–4	3–0
D: D00100	1	2	A	B
D+1: D00101	8	9	E	F
D+2: D00102	0	0	0	0

■ Example 2: Converting ASCII Data in BCD Format to Hexadecimal Data

When CIO 000001 is ON in the following example, the ASCII characters in D00000 and D00001 are converted to BCD data and the result is stored temporarily in D00010. Next, the BCD data is converted to hexadecimal and the result is output to D00100.



	15–8	7–0
S: D00000	31	32
S+1: D00001	33	34

ASCII (BCD) ⇩ BCD

	15–12	11–8	7–4	3–0
D: D00010	1	2	3	4

BCD ⇩ Binary (hexadecimal)

	15–12	11–8	7–4	3–0
D00100	0	4	D	2

&1234 Decimal (#04D2 hexadecimal)

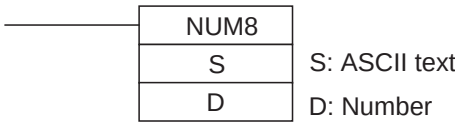
3-12-23 ASCII TO EIGHT-DIGIT NUMBER: NUM8(605)

Purpose

Converts 8 characters of ASCII data to an 8-digit hexadecimal number.

This instruction is supported by CS/CJ-series CPU Units with unit version 4.0 or later only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	NUM8(605)
	Executed Once for Upward Differentiation	@NUM8(605)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	CIO 0000 to CIO 6142
Work Area	W000 to W508	W000 to W510
Holding Bit Area	H000 to H508	H000 to H510
Auxiliary Bit Area	A448 to A956	A448 to A958
Timer Area	T0000 to T4092	T0000 to T4094
Counter Area	C0000 to C4092	C0000 to C4094
DM Area	D00000 to D32764	D00000 to D32766
EM Area without bank	E00000 to E32764	E00000 to E32766
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	---
Data Registers	---	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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.

	15 – 8	7 – 0
S	31	32
S+1	33	34
S+2	35	36
S+3	37	38

ASCII ⇩ Hexadecimal

	15 – 12	11 – 8	7 – 4	3 – 0
D	5	6	7	8
D+1	1	2	3	4

Note If the numerical data is 0, the Equals Flag will turn ON.
If the leftmost bit of the numerical data is 1, the Negative Flag will turn ON.

Restrictions

The following restrictions apply to NUM8(605).

■ **Restrictions on the CPU Unit**

NUM8(605) can be used in CPU Units with unit version 4.0 or later only.

■ **Restrictions on the CX-Programmer**

NUM8(605) can be used in CX-Programmer version 7 or higher only.

Flags

Name	Label	Operation
Error Flag	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.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the source data is 1. OFF in all other cases.

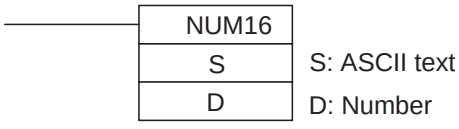
3-12-24 ASCII TO SIXTEEN-DIGIT NUMBER: NUM16(606)

Purpose

Converts 16 characters of ASCII data to an 16-digit hexadecimal number.

This instruction is supported by CS/CJ-series CPU Units with unit version 4.0 or later only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	NUM16(606)
	Executed Once for Upward Differentiation	@NUM16(606)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6136	CIO 0000 to CIO 6140
Work Area	W000 to W504	W000 to W508
Holding Bit Area	H000 to H504	H000 to H508
Auxiliary Bit Area	A448 to A952	A448 to A956
Timer Area	T0000 to T4088	T0000 to T4092
Counter Area	C0000 to C4088	C0000 to C4092
DM Area	D00000 to D32760	D00000 to D32764
EM Area without bank	E00000 to E32760	E00000 to E32764
EM Area with bank	En_00000 to En_32760 (n = 0 to C)	En_00000 to En_32764 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	---
Data Registers	---	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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 and 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.

	15 8	7 0
S	30	31
S+1	32	33
S+2	34	35
S+3	36	37
S+4	38	39
S+5	41	42
S+6	43	44
S+7	45	46

ASCII

⇩

Hexadecimal

	15 12	11 8	7 4	3 0
D	C	D	E	F
D+1	8	9	A	B
D+2	4	5	6	7
D+3	0	1	2	3

Note If the numerical data is 0, the Equals Flag will turn ON.
If the leftmost bit of the numerical data is 1, the Negative Flag will turn ON.

Restrictions

The following restrictions apply to NUM16(606).

■ **Restrictions on the CPU Unit**

NUM16(606) can be used in CPU Units with unit version 4.0 or later only.

■ **Restrictions on the CX-Programmer**

NUM16(606) can be used in CX-Programmer version 7 or higher only.

Flags

Name	Label	Operation
Error Flag	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.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the source data is 1. OFF in all other cases.

3-13 Logic Instructions

This section describes instructions which perform logic operations on word data.

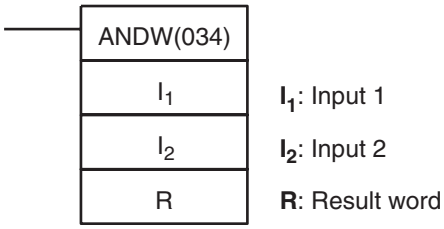
Instruction	Mnemonic	Function code	Page
LOGICAL AND	ANDW	034	548
DOUBLE LOGICAL AND	ANDL	610	550
LOGICAL OR	ORW	035	551
DOUBLE LOGICAL OR	ORWL	611	553
EXCLUSIVE OR	XORW	036	555
DOUBLE EXCLUSIVE OR	XORL	612	557
EXCLUSIVE NOR	XNRW	037	559
DOUBLE EXCLUSIVE NOR	XNRL	613	560
COMPLEMENT	COM	029	562
DOUBLE COMPLEMENT	COML	614	564

3-13-1 LOGICAL AND: ANDW(034)

Purpose

Takes the logical AND of corresponding bits in single words of word data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	ANDW(034)
	Executed Once for Upward Differentiation	@ANDW(034)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		

Area	I_1	I_2	R
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

ANDW(034) takes the logical AND of data specified in I_1 and I_2 and outputs the result to R.

- The logical AND is taken of corresponding bits in I_1 and I_2 in succession.
- When the content of corresponding bits in both I_1 and I_2 are 1 or when either is 0, a 0 will be output to the corresponding bit in R.

I_1, I_2 → R

I_1	I_2	R
1	1	1
1	0	0
0	1	0
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

When ANDW(034) is executed, the Error Flag will turn OFF.

If as a result of the AND, the content of R is 0000 hex, the Equals Flag will turn ON.

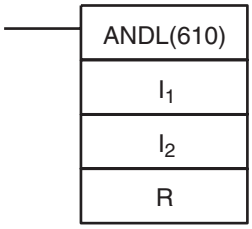
If as a result of the AND, the leftmost bit of R is 1, the Negative Flag will turn ON.

3-13-2 DOUBLE LOGICAL AND: ANDL(610)

Purpose

Takes the logical AND of corresponding bits in double words of word data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	ANDL(610)
	Executed Once for Upward Differentiation	@ANDL(610)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

ANDL(610) takes the logical AND of data specified in I_1, I_1+1 and I_2, I_2+1 and outputs the result to R, R+1.

(I_1, I_1+1), (I_2, I_2+1) → (R, R+1)

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	1
1	0	0
0	1	0
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

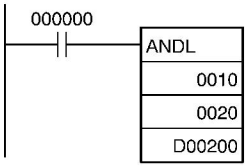
When ANDL(610) is executed, the Error Flag will turn OFF.

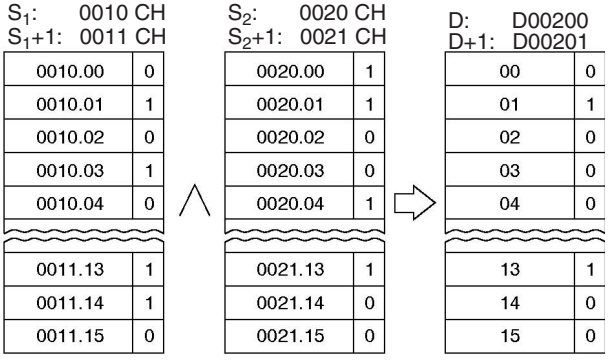
If as a result of the AND, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If as a result of the AND, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 00000000 is ON, the logical AND is taken of corresponding bits in CIO 0011, CIO 0010 and CIO 0021, CIO 0020 and the results will be output to corresponding bits in D00201 and D00200.





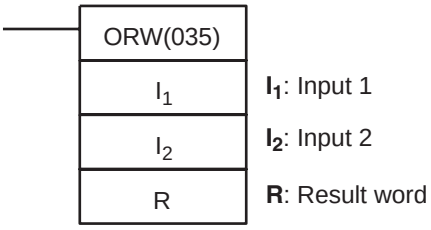
Note: The vertical arrow indicates logical AND.

3-13-3 LOGICAL OR: ORW(035)

Purpose

Takes the logical OR of corresponding bits in single words of word data and/or constants.

Ladder Symbol



I_1: Input 1

I_2: Input 2

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ORW(035)
	Executed Once for Upward Differentiation	@ORW(035)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to+2047 ,IR0 to –2048 to+2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

ORW(035) takes the logical OR of data specified in I_1 and I_2 and outputs the result to R.

- The logical OR is taken of corresponding bits in I_1 and I_2 in succession.
- When either one of the corresponding bits in I_1 and I_2 are 1 or when both of them are 0, a 0 will be output to the corresponding bit in R.

$I_1 + I_2 \rightarrow R$

I_1	I_2	R
1	1	1
1	0	1

I_1	I_2	R
0	1	1
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

When ORW(035) is executed, the Error Flag will turn OFF.

If as a result of the OR, the content of R is 0000 hex, the Equals Flag will turn ON.

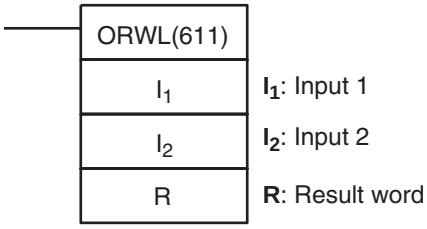
If as a result of the OR, the leftmost bit of R is 1, the Negative Flag will turn ON.

3-13-4 DOUBLE LOGICAL OR: ORWL(611)

Purpose

Takes the logical OR of corresponding bits in double words of word data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	ORWL(611)
	Executed Once for Upward Differentiation	@ORWL(611)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		

Area	I_1	I_2	R
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

ORWL(611) takes the logical OR of data specified in I_1 and I_2 as double-word data and outputs the result to R, R+1.

- When any of the corresponding bits in I_1, I_1+1, I_2, and I_2 +1are 1, a 1 will be output to the corresponding bit it R+1. When any of them are 0, a 0 will be output to the corresponding bit in R+1.

(I_1, I_1+1) + (I_2, I_2+1) → (R, R+1)

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	1
1	0	1
0	1	1
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

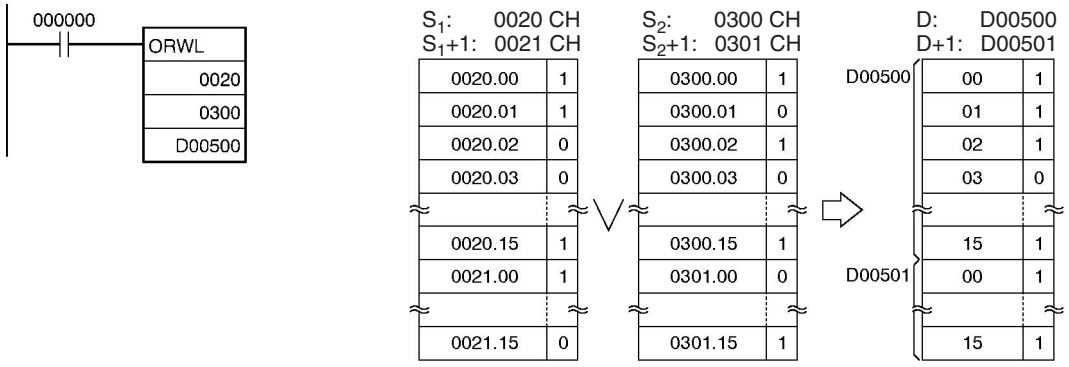
When ORWL(611) is executed, the Error Flag will turn OFF.

If as a result of the OR, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If as a result of the OR, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 00000000 is ON, the logical OR is taken of corresponding bits in CIO 0021, CIO 0020 and CIO 0301, CIO 0300 and the results will be output to corresponding bits in D00501 and D00500.



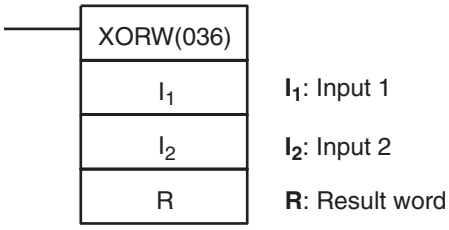
Note: The vertical arrow indicates logical OR.

3-13-5 EXCLUSIVE OR: XORW(036)

Purpose

Takes the logical exclusive OR of corresponding bits in single words of word data and/or constants.

Ladder Symbol



I1: Input 1

I2: Input 2

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	XORW(036)
	Executed Once for Upward Differentiation	@XORW(036)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		

Area	I_1	I_2	R
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

XORW(036) takes the logical exclusive OR of data specified in I_1 and I_2 and outputs the result to R.

- The logical exclusive OR is taken of corresponding bits in I_1 and I_2 in succession.
- When the content of corresponding bits of I_1 and I_2 are different, a 1 will be output to the corresponding bit of R and when there are different, 0 will be output to the corresponding bit in R.

$I_1, \overline{I_2} + \overline{I_1}, I_2 \rightarrow R$

I_1	I_2	R
1	1	0
1	0	1
0	1	1
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

When XORW(036) is executed, the Error Flag will turn OFF.

If as a result of the OR, the content of R is 0000 hex, the Equals Flag will turn ON.

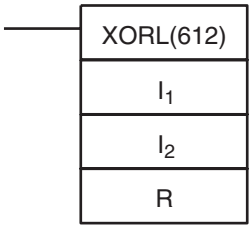
If as a result of the OR, the leftmost bit of R is 1, the Negative Flag will turn ON.

3-13-6 DOUBLE EXCLUSIVE OR: XORL(612)

Purpose

Takes the logical exclusive OR of corresponding bits in double words of word data and/or constants.

Ladder Symbol



I1: Input 1

I2: Input 2

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	XORL(612)
	Executed Once for Upward Differentiation	@XORL(612)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

XORL(612) takes the logical exclusive OR of data specified in I_1 and I_2 as double-word data and outputs the result to R, R+1.

- When the content of any of the corresponding bits in I_1, I_1+1, I_2, and I_2 +1are different, a 1 will be output to the corresponding bit it R, R+1. When any of them are the same, a 0 will be output to the corresponding bit in R, R+1.

$(I_1, I_1+1), (\overline{I_2, I_2+1}) + (\overline{I_1, I_1+1}), (I_2, I_2+1) \rightarrow (R, R+1)$

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	0
1	0	1
0	1	1
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

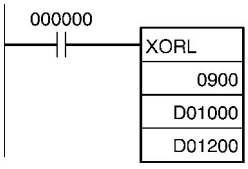
When XORL(612) is executed, the Error Flag will turn OFF.

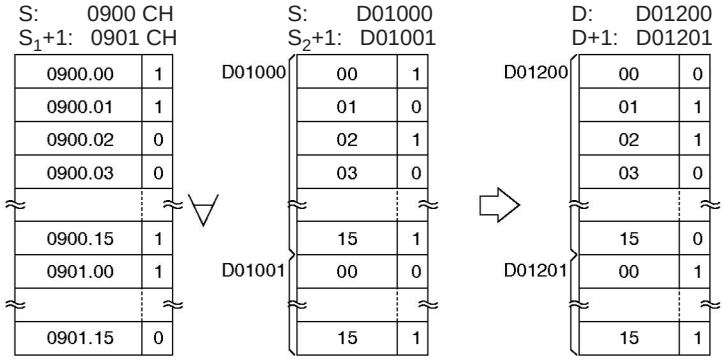
If as a result of the exclusive OR, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If as a result of the exclusive OR, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 00000000 is ON, the logical exclusive OR is taken of corresponding bits in CIO 0901, CIO 0900 and D01001, D01000 and the results will be output to corresponding bits in D01201 and D01200.





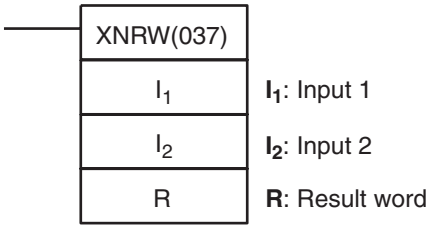
Note: The symbol indicates exclusive logical OR.

3-13-7 EXCLUSIVE NOR: XNRW(037)

Purpose

Takes the logical exclusive NOR of corresponding single words of word data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	XNRW(037)
	Executed Once for Upward Differentiation	@XNRW(037)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

XNRW(037) takes the logical exclusive NOR of data specified in I_1 and I_2 and outputs the result to R.

- The logical exclusive NOR is taken of corresponding bits in I_1 and I_2 in succession.
- When the content of corresponding bits of I_1 and I_2 are different, a 0 will be output to the corresponding bit of R and when they are different, 1 will be output to the corresponding bit in R.

$I_1, I_2 + \overline{I_1}, \overline{I_2} \rightarrow R$

I_1	I_2	R
1	1	1
1	0	0
0	1	0
0	0	1

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

When XNRW(037) is executed, the Error Flag will turn OFF.

If as a result of the NOR, the content of R is 0000 hex, the Equals Flag will turn ON.

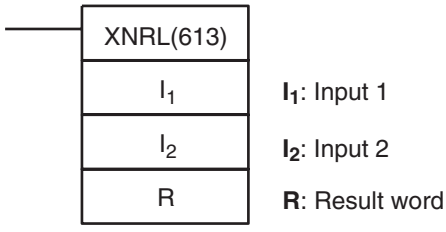
If as a result of the NOR, the leftmost bit of R is 1, the Negative Flag will turn ON.

3-13-8 DOUBLE EXCLUSIVE NOR: XNRL(613)

Purpose

Takes the logical exclusive NOR of corresponding bits in double words of word data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	XNRL(613)
	Executed Once for Upward Differentiation	@XNRL(613)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 toW 510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

XNRL(613) takes the logical exclusive NOR of data specified in I_1 and I_2 and outputs the result to R, R+1.

- When the content of any of the corresponding bits in I_1, I_1+1, I_2, and I_2 +1are different, a 0 will be output to the corresponding bit in R, R+1. When any of them are the same, a 1 will be output to the corresponding bit in R, R+1.

$(I_1, I_1+1), (I_2, I_2+1) + \overline{(I_1, I_1+1)}, \overline{(I_2, I_2+1)} \rightarrow (R, R+1)$

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	1
1	0	0
0	1	0
0	0	1

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

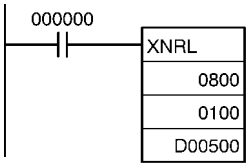
When XNRL(613) is executed, the Error Flag will turn OFF.

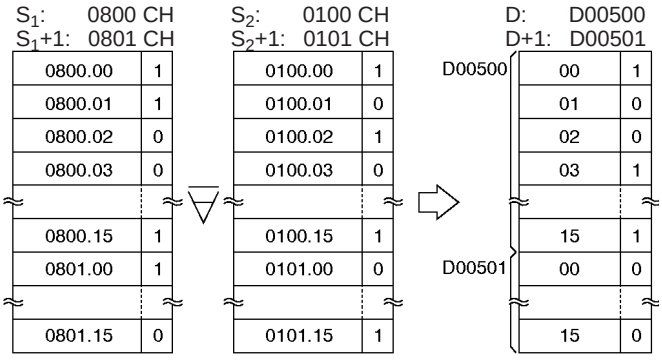
If as a result of the exclusive NOR, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If as a result of the exclusive NOR, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 00000000 is ON, the logical exclusive NOR is taken of corresponding bits in CIO 0801, CIO 0800, and CIO 0101, CIO 0100 and the results will be output to corresponding bits in D00501 and D00500.





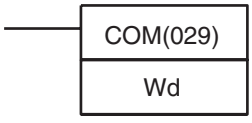
Note: The symbol indicates exclusive logical NOR.

3-13-9 COMPLEMENT: COM(029)

Purpose

Turns OFF all ON bits and turns ON all OFF bits in Wd.

Ladder Symbol



Wd: Word

Variations

Variations	**Executed Each Cycle for ON Condition**	COM(029)
	Executed Once for Upward Differentiation	@COM(029)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767

Area	Wd
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

COM(029) reverses the status of every specified bit in Wd.
$\overline{Wd} \rightarrow Wd$: 1 → 0 and 0 → 1

Note When using the COM instruction, be aware that the status of each bit will change each cycle in which the execution condition is ON.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

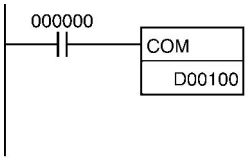
When COM(029) is executed, the Error Flag will turn OFF.

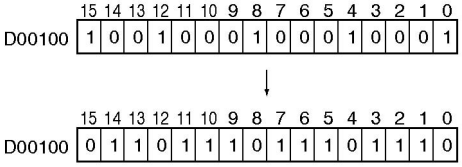
If as a result of COM, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of COM, the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, the status of each bit will be D00100 is reversed.



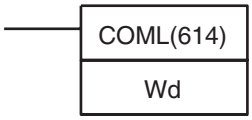


3-13-10 DOUBLE COMPLEMENT: COML(614)

Purpose

Turns OFF all ON bits and turns ON all OFF bits in Wd and Wd+1.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	COML(614)
	Executed Once for Upward Differentiation	@COML(614)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W510
Holding Bit Area	H000 to H510
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T4094
Counter Area	C0000 to C4094
DM Area	D00000 to D32766
EM Area without bank	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

COML(614) reverses the status of every specified bit in Wd and Wd+1.
$\overline{(Wd+1, Wd)} \rightarrow (Wd+1, Wd)$

Note When using the COM instruction, be aware that the status of each bit will change each cycle in which the execution condition is ON.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

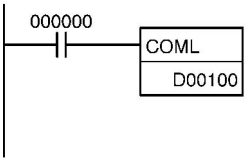
When COML(614) is executed, the Error Flag will turn OFF.

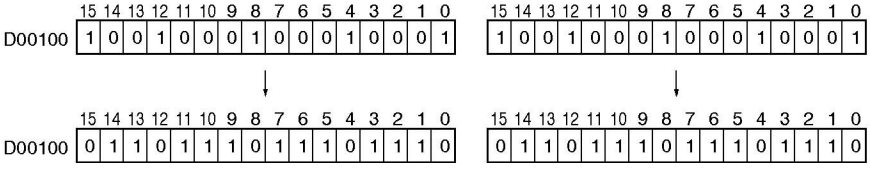
If as a result of COML, the content of R, R+1 is 00000000 hex, the Equals Flag will turn ON.

If as a result of COML, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON in the following example, the status of each bit in D00100 and D00101 will be reversed.





3-14 Special Math Instructions

This section describes instructions used for special math calculations.

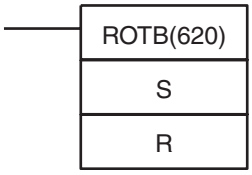
Instruction	Mnemonic	Function code	Page
BINARY ROOT	ROTB	620	565
BCD SQUARE ROOT	ROOT	072	567
ARITHMETIC PROCESS	APR	069	571
FLOATING POINT DIVIDE	FDIV	079	583
BIT COUNTER	BCNT	067	587

3-14-1 BINARY ROOT: ROTB(620)

Purpose

Computes the square root of the 32-bit signed binary contents (positive value) of the specified words and outputs the integer portion of the result to the specified result word.

Ladder Symbol



S: First source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ROTB(620)
	Executed Once for Upward Differentiation	@ROTB(620)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

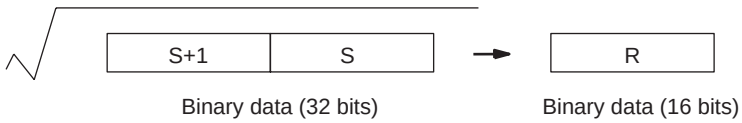
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W510	W000 to W511
Holding Bit Area	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to A958	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

ROTB(620) computes the square root of the 32-bit binary number in S+1 and S and outputs the integer portion of the result to R. The non-integer remainder is eliminated.



The range of data that can be specified for words S+1 and S is 0000 0000 to 3FFF FFFF. If a number from 4000 0000 to 7FFF FFFF is specified, it will be treated as 3FFF FFFF for the square root computation. An error will occur if the content of the source words is greater than 7FFF FFFF, i.e., if bit 15 of S+1 is 1.

Flags

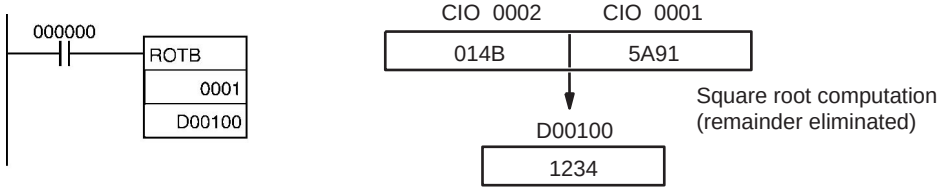
Name	Label	Operation
Error Flag	ER	ON if bit 15 of S+1 is 1 (ON). OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Overflow Flag	OF	ON if the content of S+1 and S is 4000 0000 to 7FFF FFFF. OFF in all other cases.
Underflow Flag	UF	OFF
Negative Flag	N	OFF

Precautions

The content of S+1 and S must be less than 8000 0000.
The operands of this instruction (S+1, S, and R) are all treated as binary values. If the input data is BCD, use the ROOT(072) instruction.

Example

When CIO 000000 is ON in the following example, ROTB(620) calculates the square root of the data in CIO 0002 and CIO 0001, and writes the integer portion of the result in D00100.

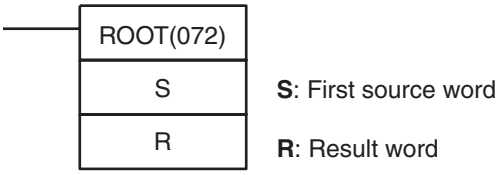


3-14-2 BCD SQUARE ROOT: ROOT(072)

Purpose

Computes the square root of an 8-digit BCD number and outputs the integer portion of the result to the specified result word.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	ROOT(072)
	Executed Once for Upward Differentiation	@ROOT(072)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

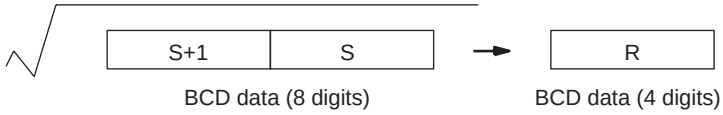
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W510	W000 to W511
Holding Bit Area	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to A958	A448 to A959

Area	S	R
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #99999999 (BCD)	---
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

ROOT(072) computes the square root of the 8-digit BCD number in S+1 and S and outputs the integer portion of the result to R. The non-integer remainder is eliminated.



Flags

Name	Label	Operation
Error Flag	ER	ON if the data in S+1 and S is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

Precautions

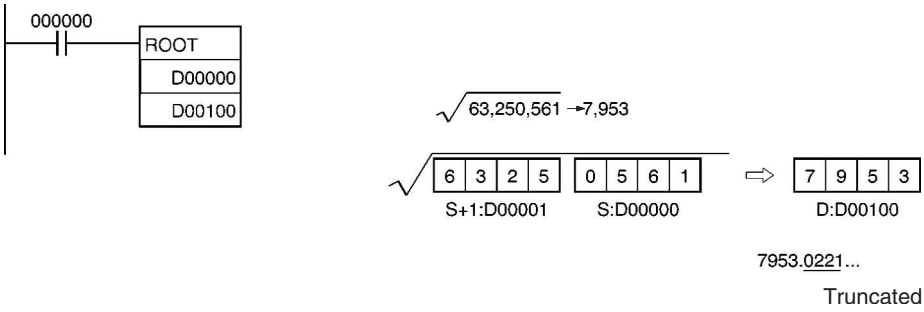
The operands of this instruction (S+1, S, and R) are all treated as BCD values. If the input data is binary, use the ROTB(620) instruction.

Examples

Square Root of 8-digit Number

When CIO 000000 is ON in the following example, ROOT(072) calculates the square root of the data in D00001 and D00000, and writes the integer portion of the result in D00100.

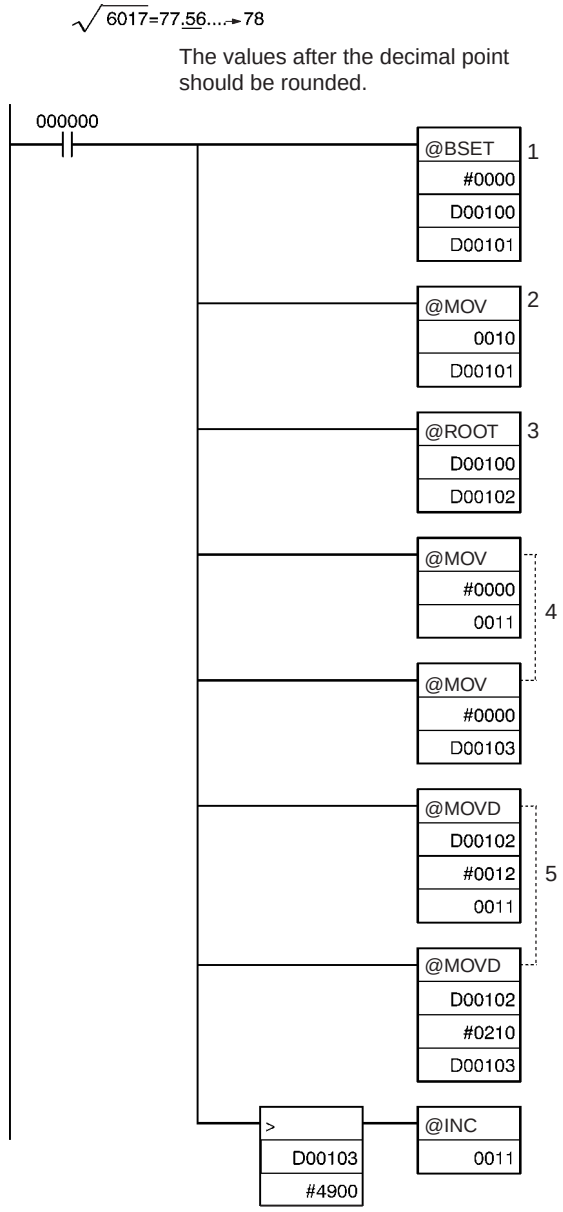
Note Figures after the decimal point are truncated for 8-digit numbers.



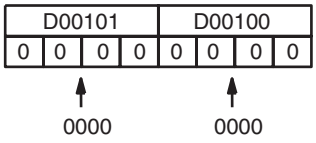
Square Root of a 4-digit Number

The following example shows how to take the square root of a 4-digit number and round off the result. This program example calculates the square root of the 4-digit number in CIO 0010, rounds off the result, and writes it to CIO 0011. (Basically, the 4-digit number is multiplied by 10,000 (100^2) and the result is divided by 100, increasing the precision of the calculation by a factor of 100.)

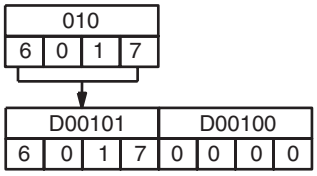
Note Figures after the decimal point are rounded for 4-digit numbers.



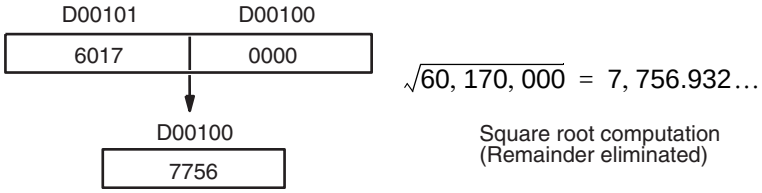
1,2,3... 1. The source words (D00101 and D00100) to be are cleared to 0000 0000.



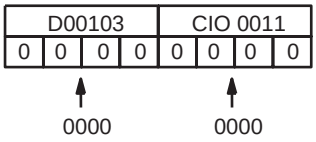
2. The 4-digit number is moved to D00101.



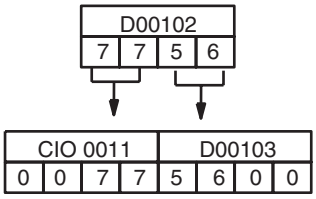
3. ROOT(072) calculates the square root of D00101 and D00100 and writes the result to D00102.



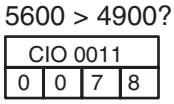
4. D00103 and the result word, CIO 0011, are cleared to 0000 0000.



5. The result of the square root calculation is divided by 100, with the integer portion written to CIO 0011 and the remainder going to D00103.



6. If the content of D00103 is greater than 4900, CIO 0011 is incremented by 1. In this case, the result is 78.

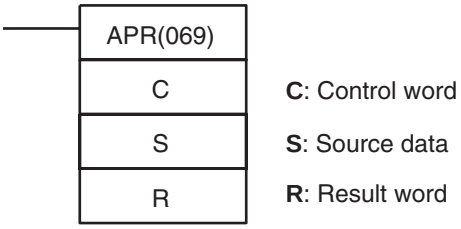


3-14-3 ARITHMETIC PROCESS: APR(069)

Purpose

Calculates the sine, cosine, or a linear extrapolation of the source data.
The linear extrapolation function allows any relationship between X and Y to be approximated with line segments.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	APR(069)
	Executed Once for Upward Differentiation	@APR(069)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

Sine Function (C = 0000 Hex)

Operand	Value	Data range
C	0000 hex	---
S	0000 to 0900 (BCD)	0° to 90°
D	0000 to 9999 (BCD)	0.0000 to 0.9999
	9999 (BCD)	1.0000

Cosine Function (C = 0001 Hex)

Operand	Value	Data range
C	0001 hex	---
S	0000 to 0900 (BCD)	0° to 90°
D	0000 to 9999 (BCD)	0.0000 to 0.9999
	9999 (BCD)	1.0000

Linear Extrapolation Function (C = Data area address)

Operand	Value	Data range
C	Data area address	---
S	16-bit unsigned BCD data	0000 to 9999
	16-bit unsigned binary data	0 to 65,535
	16-bit signed binary data[1]	–32,768 to 32,767
	32-bit signed binary data[1]	–2,147,483,648 to 2,147,483,647
	Floating-point data[1]	$-\infty$, -3.402823×10^{38} to $-1.175494 \times 10^{-38}$, 1.175494×10^{-38} to 3.402823×10^{38}, $+\infty$
D	16-bit unsigned BCD data	0000 to 9999
	16-bit unsigned binary data	0 to 65,535
	16-bit signed binary data[1]	–32,768 to 32,767
	32-bit signed binary data[1]	–2,147,483,648 to 2,147,483,647
	Floating-point data[1]	$-\infty$, -3.402823×10^{38} to $-1.175494 \times 10^{-38}$, 1.175494×10^{-38} to 3.402823×10^{38}, $+\infty$

Note

1. Signed binary data and floating-point data are supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.
2. If C is a word address, APR(069) extrapolates the Y value for the X value in S based on coordinates (forming line segments) entered in advance in a table beginning at C. Refer to the *Description* section below for details.

Operand Specifications

Area	C	S	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		

Area	C	S	R
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	Specified values only		---
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

The operation of APR(069) depends on the control word C. If C is 0000 or 0001, APR(069) computes the sine or cosine of S with S in units of tenths of degrees.

If C is a word address, APR(069) extrapolates the Y value for the X value in S based on coordinates (forming line segments) entered in advance in a table beginning at C.

Sine Function (C=0000)

When C is 0000, APR(069) calculates the SIN(S) and writes the result to R. The range for S is 0000 to 0900 BCD (0.0° to 90.0°) and the range for R is 0000 to 9999 BCD (0.0000 to 0.9999). The remainder of the result beyond the fourth decimal place is eliminated.

Cosine Function (C=0001)

When C is 0001, APR(069) calculates the COS(S) and writes the result to R. The range for S is 0000 to 0900 BCD (0.0° to 90.0°) and the range for R is 0000 to 9999 BCD (0.0000 to 0.9999). The remainder of the result beyond the fourth decimal place is eliminated.

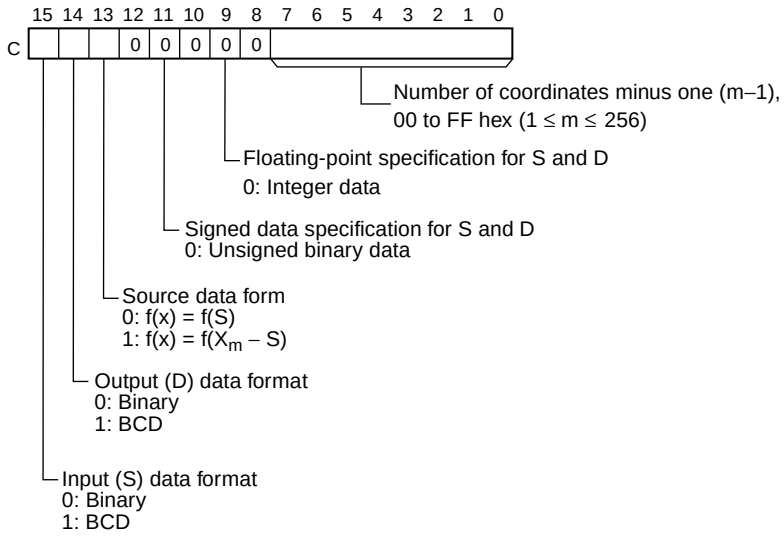
Linear Extrapolation

APR(069) linear extrapolation is specified when C is a word address.

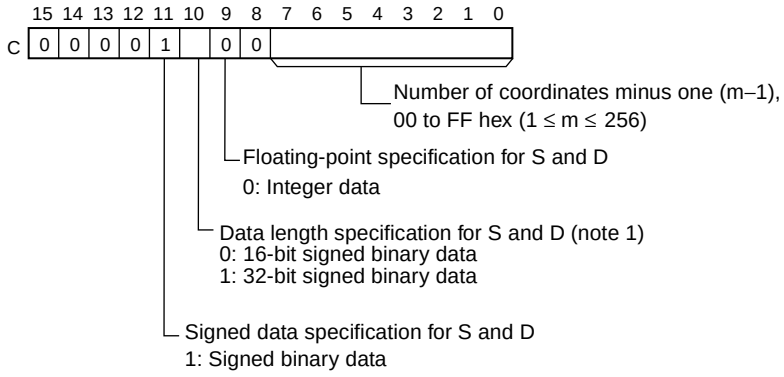
The content of word C specifies the number of coordinates in a data table starting at C+2, the form of the source data, and whether data is BCD or

binary. In CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, the source data can also be signed binary data or floating-point data.

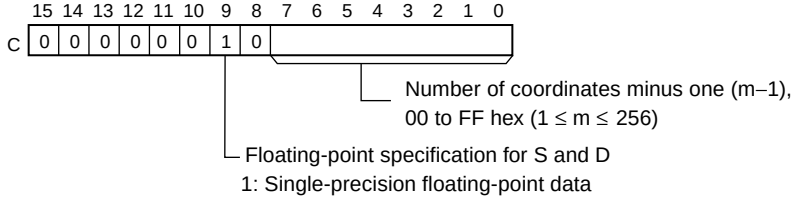
Unsigned Integer Data (Binary or BCD)



Signed Integer Data (Binary)



Single-precision Floating-point Data



If 16-bit binary or BCD data is being used, the line-segment data is contained in words C+ 1 through C+2m+2. If 32-bit binary or floating point data is being used (CS1-H, CJ1-H, and CJ1M CPU Units only), the line-segment data is contained in words C+ 1 through C+4m+4.

Bits 00 to 07 contain the number (binary) of line coordinates less 1, m–1. Bits 08 to 12 are not used. Bit 13 specifies either f(x)=f(S) or f(x)=f(X_m–S): OFF specifies f(x)=f(S) and ON specifies f(x)=f(X_m–S). Bit 14 determines whether the output is BCD or binary: OFF specifies binary and ON specifies BCD. Bit

15 determines whether the input is BCD or binary: OFF specifies binary and ON specifies BCD.

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 X_m (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)

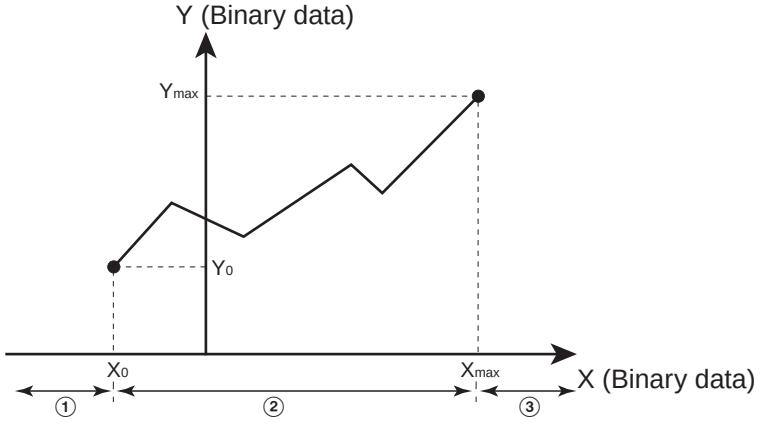
Floating-point data

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)

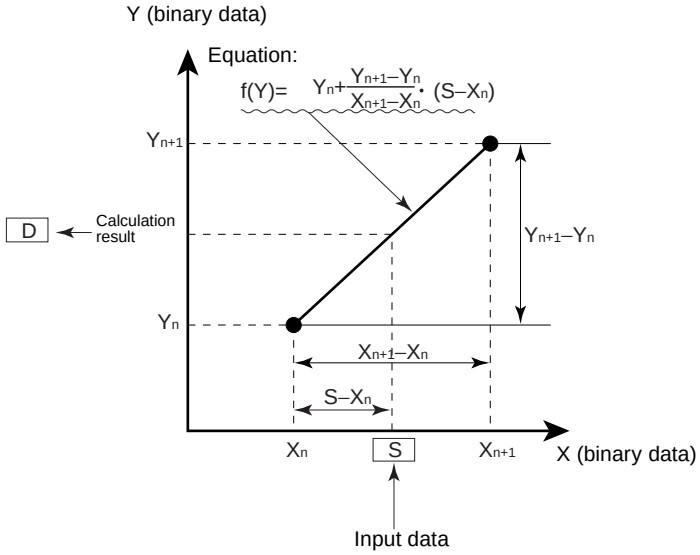
Note The X coordinates must be in ascending order: $X_1 < X_2 < ... < X_m$. Input all values of (X_n, Y_n) as binary data, regardless of the data format specified in control word C.

Operation of the Linear Extrapolation Function

APR(069) processes the input data specified in S with the following equation and the line-segment data (X_n, Y_n) specified in the table beginning at C+1. The result is output to the destination word(s) specified with D.



1. For $S < X_0$
   Converted value = Y_0
2. For $X_0 \leq S \leq X_{max}$, if $X_n < S < X_{n+1}$
   Converted value = $Y_n + [\{Y_n + 1 - Y_n\}/\{X_n + 1 - X_n\}] \times \{$*Input data* $S - X_n\}$



3. $X_{max} < S$
   Converted value = Y_{max}

Up to 256 endpoints can be stored in the line-segment data table beginning at C+1. The following 5 kinds of I/O data can be used:

- 16-bit unsigned BCD data
- 16-bit unsigned binary data
- 16-bit signed binary data (CS1-H/CJ1-H/CJ1M Only)
- 32-bit signed binary data (CS1-H/CJ1-H/CJ1M Only)
- Single-precision floating-point data (CS1-H/CJ1-H/CJ1M Only)

Setting the Data Format in Control Word C

- 16-bit Unsigned BCD Data
  The input data and/or the output data can be 16-bit unsigned BCD data. Also, the linear extrapolation function can be set to operate on the value specified in S directly or on X_m–S. (X_m is the maximum value of X in the line-segment data.)

Setting name	Bit in C	Setting
Input data (S) format	15	0: Binary 1: BCD
Output data (D) format	14	0: Binary 1: BCD
Source data form	13	0: Operate on S 1: Operate on X_m–S
Signed data specification for S and D	11	0: Unsigned data
Data length specification for S and D	10	Invalid (fixed at 16 bits)
Floating-point specification	09	0: Integer data

- 16-bit Unsigned Binary Data
  The input data and/or the output data can be 16-bit unsigned binary data. Also, the linear extrapolation function can be set to operate on the value specified in S directly or on X_m–S. (X_m is the maximum value of X in the line-segment data.)

Setting name	Bit in C	Setting
Input data (S) format	15	0: Binary 1: BCD
Output data (D) format	14	0: Binary 1: BCD
Source data form	13	0: Operate on S 1: Operate on X_m–S
Signed data specification for S and D	11	0: Unsigned data
Data length specification for S and D	10	Invalid (fixed at 16 bits)
Floating-point specification	09	0: Integer data

- 16-bit Signed Binary Data (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 (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	1: 32-bit signed binary data
Floating-point specification	09	0: Integer data

Note If the “Data length specification for S and D” in bit 10 of C is set to 1 and a 16-bit constant is input for S, the input data will be converted to 32-bit signed binary before the linear extrapolation calculation.

- Floating-point Data (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	0
Data length specification for S and D	10	0
Floating-point specification	09	1: Floating-point data

Note If the “Floating-point specification” in bit 09 of C is set to 1, a constant cannot be input for S.

Flags

Name	Label	Operation
Error Flag	ER	ON if C is a constant greater than 0001. ON if C is a word address but the X coordinates are not in ascending order ($X_1 \le X_2 \le ... \le X_m$). ON if C is a word address and bits 9, 11, and 15 of C indicate BCD input, but S is not BCD. ON if C is a word address and bit 9 of C indicates floating-point data, but S is a one-word constant. ON if C is 0000 or 0001 but S is not BCD between 0000 and 0900. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of R is ON. OFF in all other cases.

Precautions

The actual result for SIN(90°) and COS(0°) is 1, but 9999 (0.9999) will be output to R.

An error will occur if C is a constant greater than 0001.

An error will occur if linear extrapolation is specified but the X coordinates are not in ascending order ($X_1 < X_2 < ... < X_n < S < X_{n+1}$).

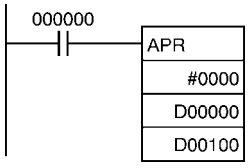
An error will occur if linear extrapolation is specified and BCD input is specified (bit 15 of C ON) but S is not BCD.

An error will occur if a trigonometric function is specified (C=0000 or 0001) but S is not BCD between 0000 and 0900.

Examples

Sine Function (C: #0000)

The following example shows APR(069) used to calculate the sine of 30°.



Source data

S: D00000			
0	10^1	10^0	10^{-1}
0	3	0	0

Set the source data in 10^{-1} degrees. (0000 to 0900, BCD)

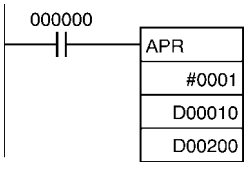
Result

R: D00100			
10^{-1}	10^{-2}	10^{-3}	10^{-4}
5	0	0	0

Result data has four significant digits, fifth and higher digits are ignored. (0000 to 9999, BCD)

Cosine Function (C: #0001)

The following example shows APR(069) used to calculate the cosine of 30°. (SIN(30) = 0.8660)



Source data

S: D00010			
0	10^1	10^0	10^{-1}
0	3	0	0

Set the source data in 10^{-1} degrees. (0000 to 0900, BCD)

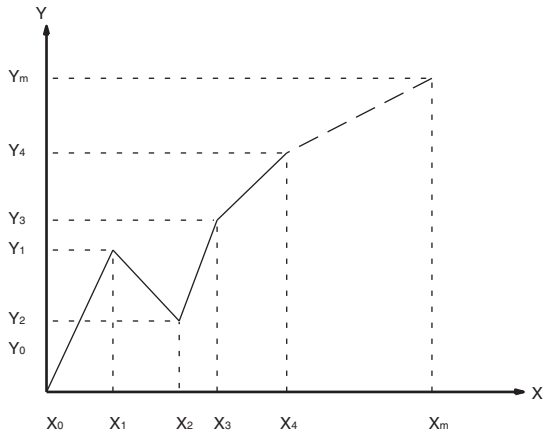
Result

R: D00200			
10^{-1}	10^{-2}	10^{-3}	10^{-4}
8	6	6	0

Result data has four significant digits, fifth and higher digits are ignored. (0000 to 9999, BCD)

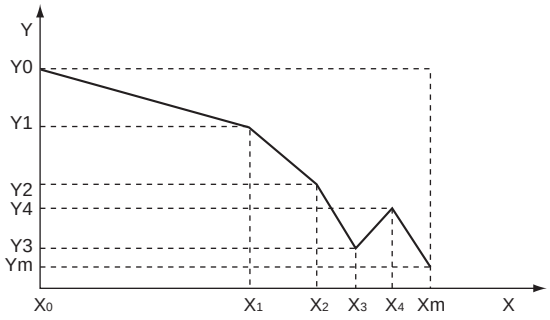
Linear Extrapolation (C: Word Address)
Using 16-bit Unsigned BCD or Binary Data

APR(069) processes the input data specified in S based on the control data in C and the line-segment data specified in the table beginning at C+1. The result is output to D.

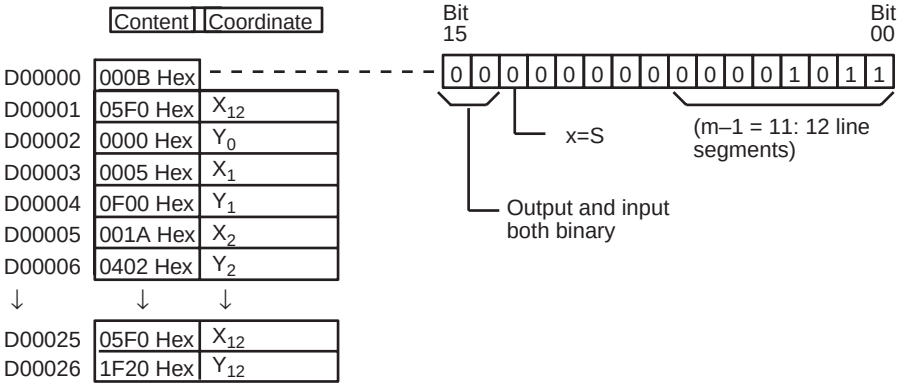


Word	Coordinate
C+1	Xm (max. X value)
C+2	Y_0
C+3	X_1
C+4	Y_1
C+5	X_2
C+6	Y_2
↓	↓
C+(2m+1)	X_m (max. X value)
C+(2m+2)	Y_m

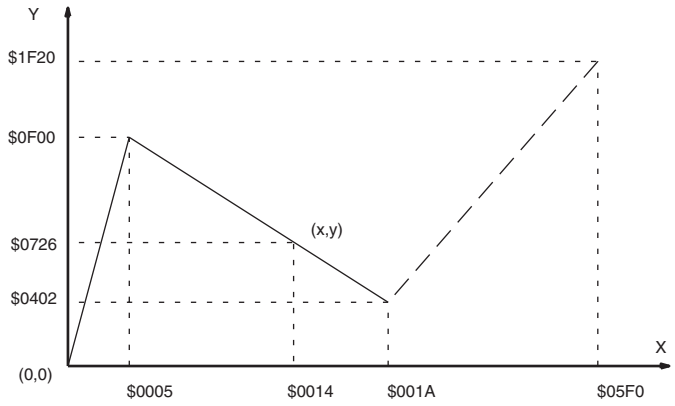
- $Y_n = f(X_n)$, $Y_0 = f(X_0)$
- Be sure that $X_{n-1} < X_n$ in all cases.
- Input all values of (X_n, Y_n) as binary data.



This example shows how to construct a linear extrapolation with 12 coordinates. The block of data is continuous, as it must be, from D00000 to D00026 (C to C + (2 × 12 + 2)). The input data is taken from CIO 0010, and the result is output to CIO 0011.



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.

$$Y = 0F00 + \frac{0402 - 0F00}{001A - 0005} \times (0014 - 0015)$$

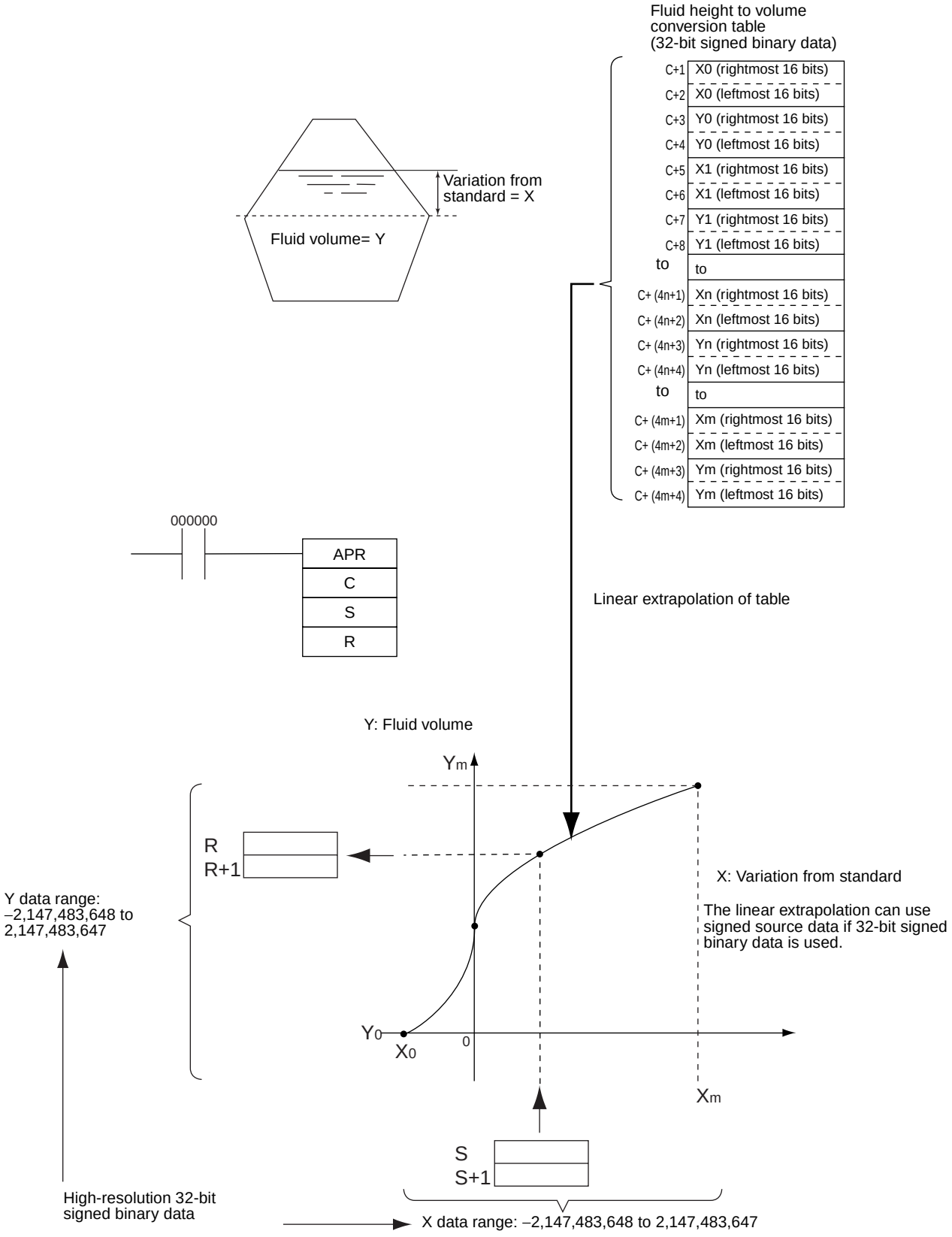
$$= 0F00 - (0086 \times 000F)$$

$$= 0726$$

Values are all hexadecimal (Hex).

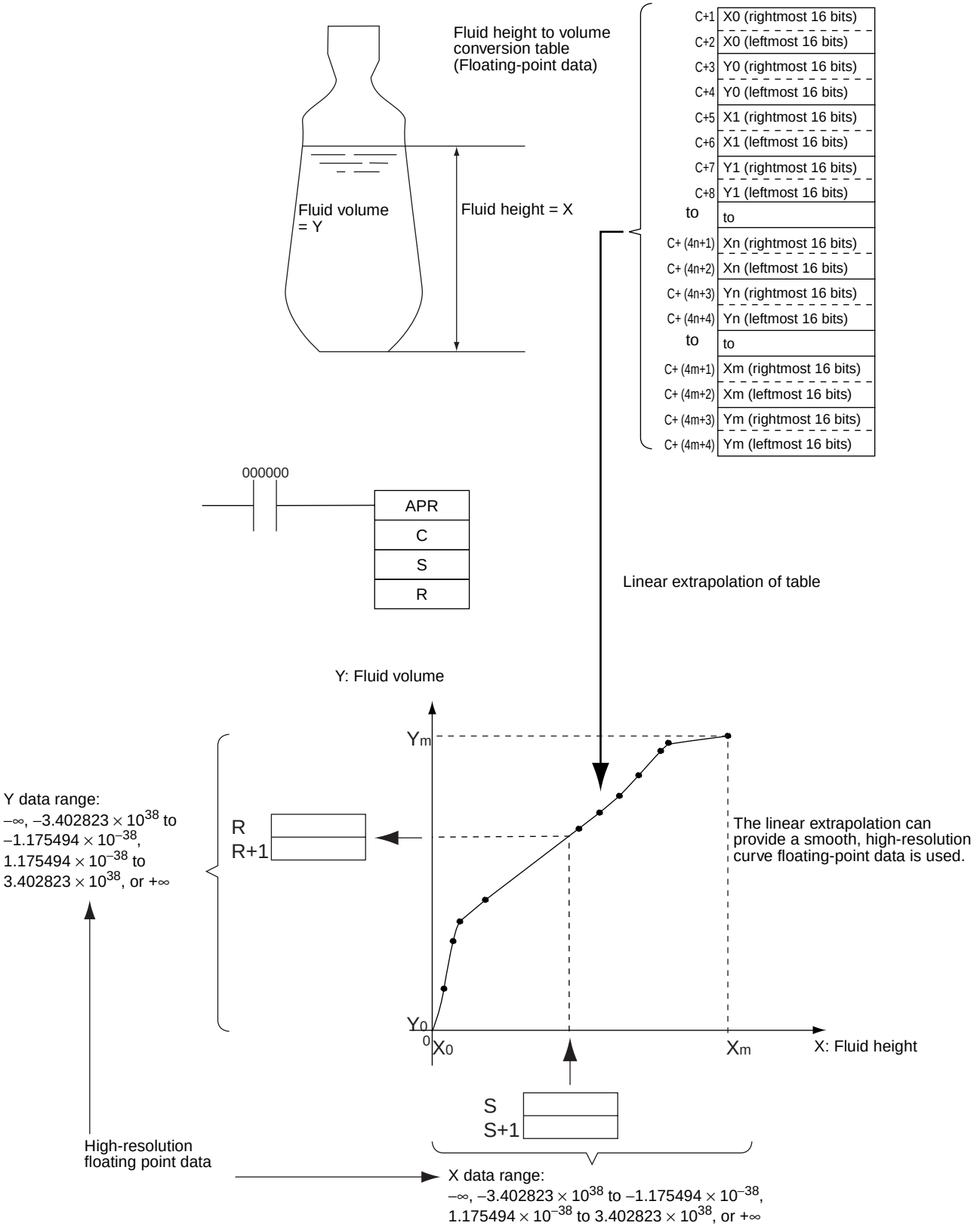
Linear Extrapolation (C: Word Address) Using 32-bit Signed Binary Data (CS1-H, CJ1-H, CJ1M, and CS1D Only)

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.



Linear Extrapolation (C: Word Address) Using Floating-point Data (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.

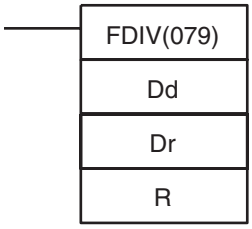


3-14-4 FLOATING POINT DIVIDE: FDIV(079)

Purpose

Divides one 7-digit floating-point number by another. The floating-point numbers are expressed in scientific notation (7-digit mantissa and 1-digit exponent).

Ladder Symbol



Dd: First dividend word

Dr: First divisor word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	FDIV(079)
	Executed Once for Upward Differentiation	@FDIV(079)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

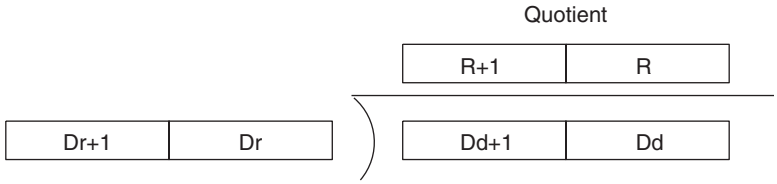
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

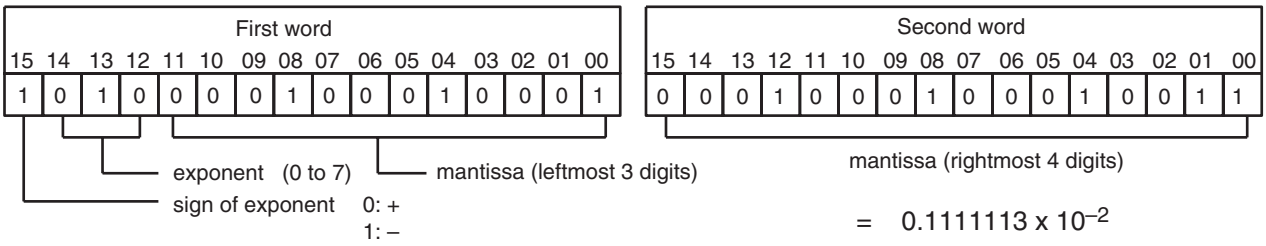
Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

FDIV(079) divides the floating-point value in Dd and Dd+1 by that in Dr and Dr+1 and places the result in R and R+1.



To represent the floating-point values, the rightmost seven digits are used for the mantissa and the leftmost digit is used for the exponent, as shown in the diagram below. The leftmost digit can range from 0 to F; positive exponents range from 0 to 7 and negative exponents range from 8 to F (0 to –7). The rightmost 7 digits must be BCD.



Two more examples of floating-point values are:

6123 4567: 0.1234567×10^{6} (6 = 0110 binary)

B123 4567: 0.1234567×10^{-3} (B = 1011 binary)

The following table shows the maximum and minimum values allowed.

Limit	8-digit hexadecimal	Floating-point
Maximum value	7999 9999	0.9999999×10^{7}
Minimum value (Divisor and dividend)	F000 0001	0.0000001×10^{-7}
Minimum value (Result)	F100 0000	0.1000000×10^{-7}

Flags

Name	Label	Operation
Error Flag	ER	ON if the mantissa (leftmost 7 digits) in Dd+1 and Dd is not BCD. ON if the mantissa (leftmost 7 digits) in Dr+1 and Dr is not BCD. ON if the divisor (Dr+1 and Dr) is 0. ON if the result is not between 0.1000000×10^{-7} and 0.9999999×10^{7}. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.

Precautions

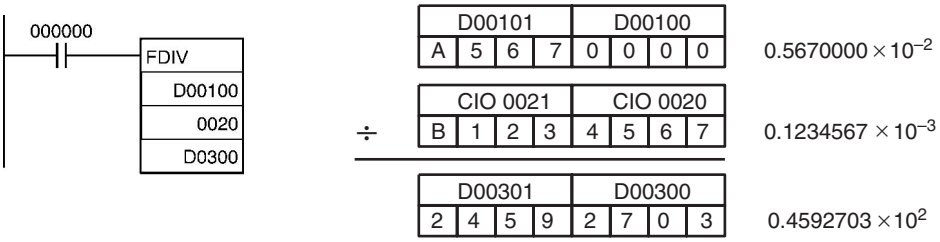
The result is expressed as a floating-point value, so it has 7 significant digits. The eighth and higher digits are eliminated.

The result must be between 0.1000000×10^{-7} and 0.9999999×10^{7}.

Examples

Basic Floating-point Division

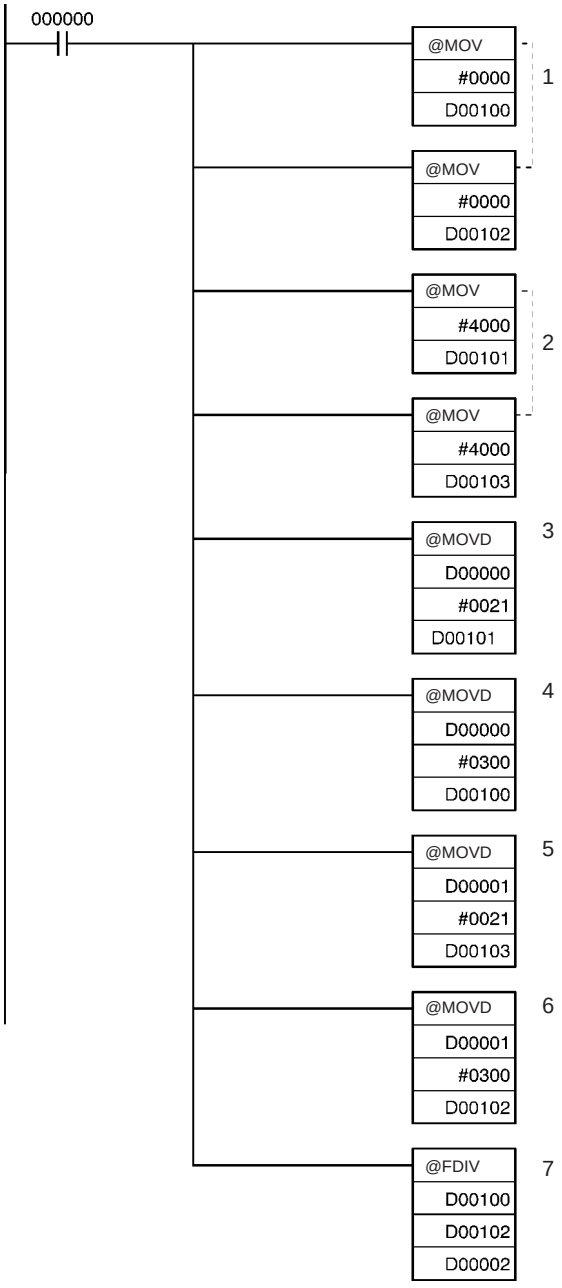
When CIO 000000 is ON in the following example, FDIV(079) divides the floating-point number in D00101 and D00100 by the floating-point number in CIO 0021 and CIO 0020 and writes the result to D00301 and D00300.



Floating-point Division of Two BCD Numbers

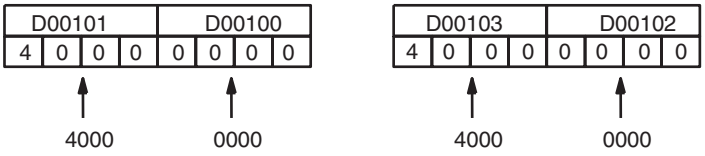
In this example, the 4-digit BCD number in D00000 is divided by the 4-digit BCD number in D00001 and the floating-point result is written to D00003 and D00002.

To perform the floating point division, the BCD value in D00000 is converted to floating-point format in D00101 and D00100 and the BCD value in D00001 is converted to floating-point format in D00103 and D00102.

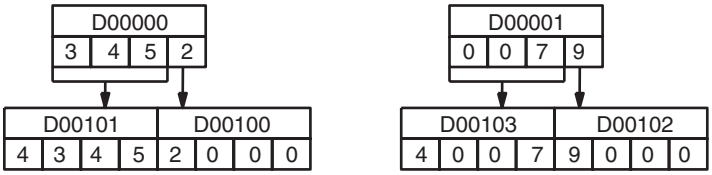


1,2,3... 1. D00100 and D00102 are set to 0000.

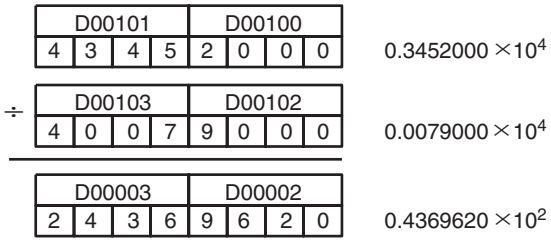
2. D00101 and D00103 are set to 4000.



3. MOVD(083) is used to move the digits of the original source words to the proper digits in the 2-word floating-point formats.



4. FDIV(079) divides the floating-point number in D00101 and D00100 by the floating-point number in D00103 and D00102.

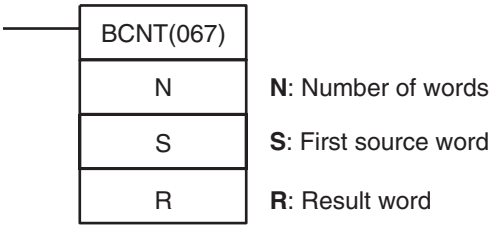


3-14-5 BIT COUNTER: BCNT(067)

Purpose

Counts the total number of ON bits in the specified word(s).

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	BCNT(067)
	Executed Once for Upward Differentiation	@BCNT(067)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Number of words

The number of words must be 0001 to FFFF (1 to 65,535 words).

S: First source word

S and S+(N–1) must be in the same data area.

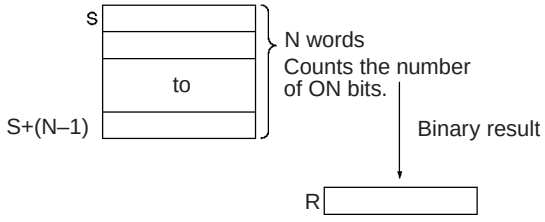
Operand Specifications

Area	N	S	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		

Area	N	S	R
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0001 to #FFFF (binary) or &1 to &65,535	---	
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

BCNT(067) counts the total number of bits that are ON in all words between S and S+(N–1) and places the result in R.



Flags

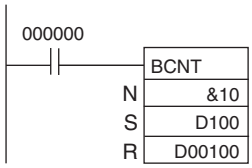
Name	Label	Operation
Error Flag	ER	ON if N is 0000. ON if result exceeds FFFF. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

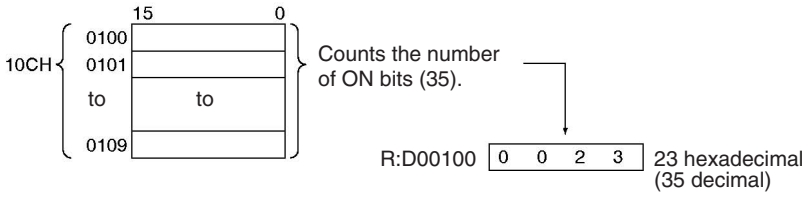
Precautions

An error will occur if N=0000 or the result exceeds FFFF.

Example

When CIO 000000 is ON in the following example, BCNT(067) counts the total number of ON bits in the 10 words from CIO 0100 through CIO 0109 and writes the result to D00100.





3-15 Floating-point Math Instructions

The Floating-point Math Instructions convert data and perform floating-point arithmetic operations. CS/CJ-series CPU Units support the following instructions.

Instruction	Mnemonic	Function code	Page
FLOATING TO 16-BIT	FIX	450	594
FLOATING TO 32-BIT	FIXL	451	596
16-BIT TO FLOATING	FLT	452	597
32-BIT TO FLOATING	FLTL	453	599
FLOATING-POINT ADD	+F	454	601
FLOATING-POINT SUBTRACT	–F	455	603
FLOATING-POINT MULTIPLY	*F	456	605
FLOATING-POINT DIVIDE	/F	457	607
DEGREES TO RADIANS	RAD	458	609
RADIANS TO DEGREES	DEG	459	610
SINE	SIN	460	612
HIGH-SPEED SINE	SINQ	475	614
COSINE	COS	461	615
HIGH-SPEED COSINE	COSQ	476	617
TANGENT	TAN	462	619
HIGH-SPEED TANGENT	TANQ	477	621
ARC SINE	ASIN	463	623
ARC COSINE	ACOS	464	625
ARC TANGENT	ATAN	465	627
SQUARE ROOT	SQRT	466	629
EXPONENT	EXP	467	631
LOGARITHM	LOG	468	633
EXPONENTIAL POWER	PWR	840	635
MOVE FLOATING-POINT (SINGLE)	MOVF	469	649

In addition to the instructions listed above, the CS1-H/CJ1-H CPU Units support the following floating-point comparison and conversion instructions. Refer to *3-16-21 Double-precision Floating-point Input Instructions* for details on double-precision floating-point instructions.

Instruction	Mnemonic	Function code	Page
Single-precision Floating-point Symbol Comparison Instructions (*CS1-H/CJ1-H/CJ1M Only)	LD, AND, OR + =F, <>F, <F, <=F, >F, or >=F	329 to 334	636
FLOATING-POINT TO ASCII (*CS1-H/CJ1-H/CJ1M Only)	FSTR	448	640
ASCII TO FLOATING-POINT (*CS1-H/CJ1-H/CJ1M Only)	FVAL	449	645

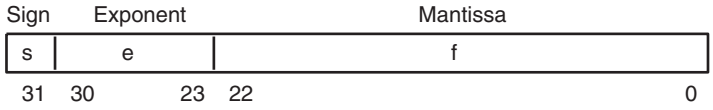
Data Format

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.

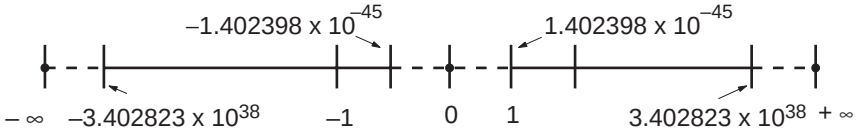
Number of Digits

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:

- $-\infty$
- $-3.402823 \times 10^{38} \leq \text{value} \leq -1.402398 \times 10^{-45}$
- 0
- $1.402398 \times 10^{-45} \leq \text{value} \leq 3.402823 \times 10^{38}$
- $+\infty$
- Not a number (NaN)



Special Numbers

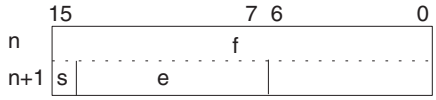
The formats for NaN, ±∞, and 0 are as follows:

NaN*: e = 255, f ≠ 0
+∞: e = 255, f = 0, s= 0
–∞: e = 255, f = 0, s= 1
0: e = 0

*NaN (not a number) is not a valid floating-point number. Executing floating-point calculation instructions will not result in NaN.

Writing Floating-point Data

When floating-point is specified for the data format in the I/O memory edit display in the CX-Programmer, standard decimal numbers input in the display are automatically converted to the floating-point format shown above (IEEE754-format) and written to I/O Memory. Data written in the IEEE754-format is automatically converted to standard decimal format when monitored on the display.



It is not necessary for the user to be aware of the IEEE754 data format when reading and writing floating-point data. It is only necessary to remember that floating point values occupy two words each.

Numbers Expressed as Floating-point Values

The following types of floating-point numbers can be used.

Mantissa (f)	Exponent (e)		
	0	Not 0 and not all 1's	All 1's (255)
0	0	Normalized number	Infinity
Not 0	Non-normalized number		NaN

Note A non-normalized number is one whose absolute value is too small to be expressed as a normalized number. Non-normalized numbers have fewer significant digits. If the result of calculations is a non-normalized number (including intermediate results), the number of significant digits will be reduced.

Normalized Numbers

Normalized numbers express real numbers. The sign bit will be 0 for a positive number and 1 for a negative number.

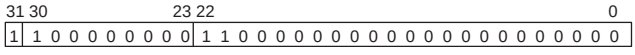
The exponent (e) will be expressed from 1 to 254, and the real exponent will be 127 less, i.e., –126 to 127.

The mantissa (f) will be expressed from 0 to $2^{33} – 1$, and it is assume that, in the real mantissa, bit 2^{33} is 1 and the binary point follows immediately after it.

Normalized numbers are expressed as follows:

$(–1)^{(\text{sign s})} \times 2^{(\text{exponent e})–127} \times (1 + \text{mantissa} \times 2^{–23})$

Example



Sign: –
Exponent: 128 – 127 = 1
Mantissa: $1 + (2^{22} + 2^{21}) \times 2^{–23} = 1 + (2^{–1} + 2^{–2}) = 1 + 0.75 = 1.75$
Value: $–1.75 \times 2^{1} = –3.5$

Non-normalized Numbers

Non-normalized numbers express real numbers with very small absolute values. The sign bit will be 0 for a positive number and 1 for a negative number.

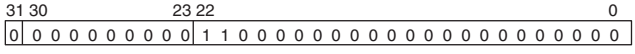
The exponent (e) will be 0, and the real exponent will be –126.

The mantissa (f) will be expressed from 1 to $2^{33} – 1$, and it is assume that, in the real mantissa, bit 2^{33} is 0 and the binary point follows immediately after it.

Non-normalized numbers are expressed as follows:

$(–1)^{(\text{sign s})} \times 2^{–126} \times (\text{mantissa} \times 2^{–23})$

Example



Sign: –
Exponent: –126
Mantissa: $0 + (2^{22} + 2^{21}) \times 2^{–23} = 0 + (2^{–1} + 2^{–2}) = 0 + 0.75 = 0.75$
Value: $–0.75 \times 2^{–126}$

Zero

Values of +0.0 and –0.0 can be expressed by setting the sign to 0 for positive or 1 for negative. The exponent and mantissa will both be 0. Both +0.0 and –0.0 are equivalent to 0.0. Refer to *Floating-point Arithmetic Results*, below, for differences produced by the sign of 0.0.

Infinity

Values of +∞ and –∞ can be expressed by setting the sign to 0 for positive or 1 for negative. The exponent will be 255 ($2^8 - 1$) and the mantissa will be 0.

NaN

NaN (not a number) is produced when the result of calculations, such as 0.0/0.0, ∞/∞, or ∞–∞, does not correspond to a number or infinity. The exponent will be 255 ($2^8 - 1$) and the mantissa will be not 0.

Note There are no specifications for the sign of NaN or the value of the mantissa field (other than to be not 0).

Floating-point Arithmetic Results

Rounding Results

The following methods will be used to round results when the number of digits in the accurate result of floating-point arithmetic exceeds the significant digits of internal processing expressions.

If the result is close to one of two internal floating-point expressions, the closer expression will be used. If the result is midway between two internal floating-point expressions, the result will be rounded so that the last digit of the mantissa is 0.

Overflows, Underflows, and Illegal Calculations

Overflows will be output as either positive or negative infinity, depending on the sign of the result. Underflows will be output as either positive or negative zero, depending on the sign of the result.

Illegal calculations will result in NaN. Illegal calculations include adding infinity to a number with the opposite sign, subtracting infinity from a number with the opposite sign, multiplying zero and infinity, dividing zero by zero, or dividing infinity by infinity.

The value of the result may not be correct if an overflow occurs when converting a floating-point number to an integer.

Precautions in Handling Special Values

The following precautions apply to handling zero, infinity, and NaN.

- The sum of positive zero and negative zero is positive zero.
- The difference between zeros of the same sign is positive zero.
- If any operand is a NaN, the results will be a NaN.
- Positive zero and negative zero are treated as equivalent in comparisons.
- Comparison or equivalency tests on one or more NaN will always be true for != and always be false for all other instructions.

Floating-point Calculation Results

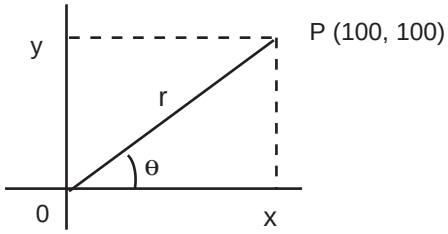
When the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞. If the result is positive, it will be output as +∞; if negative, then –∞.

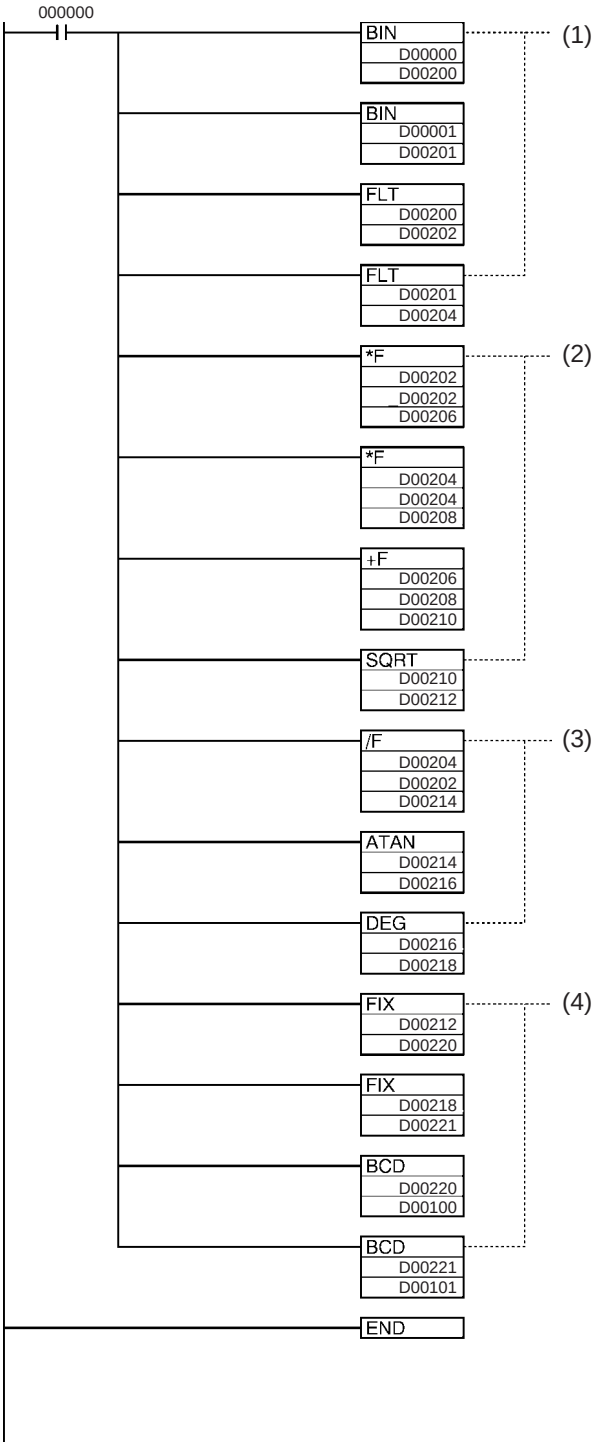
The Equals Flag will only turn ON when both the exponent (e) and the mantissa (f) are zero after a calculation. A calculation result will also be output as zero when the absolute value of the result is less than the minimum value that can be expressed for floating-point data. In that case the Underflow Flag will turn ON.

Example

In this program example, the X-axis and Y-axis coordinates (x, y) are provided by 4-digit BCD content of D00000 and D00001. The distance (r) from the ori-

gin and the angle (θ, in degrees) are found and output to D00100 and D00101. In the result, everything to the right of the decimal point is truncated.





Calculations

Distance r = $\sqrt{x^2 + y^2}$

Angle θ = $\tan^{-1}\left(\frac{y}{x}\right)$

Examples

Distance r = $\sqrt{100^2 + 100^2} = 141.4214$

Angle θ = $\tan^{-1}\left(\frac{100}{100}\right) \times 180 \div \pi = 45.0$

DM Contents

D00000	#0100 (BCD)	x	→	D00100	0 1 4 1 (BCD) r
D00001	#0100 (BCD)	y		D00101	0 0 4 5 (BCD)

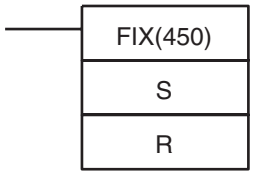
1. This section of the program converts the data from BCD to floating-point.
   a) The data area from D00200 onwards is used as a work area.
   b) First BIN(023) is used to temporarily convert the BCD data to binary data, and then FLT(452) is used to convert the binary data to floating-point data.
   c) The value of x that has been converted to floating-point data is output to D00203 and D00202.
   d) The value of y that has been converted to floating-point data is output to D00205 and D00204.
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 D00213 and D00212 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 D00219 and D00218 as floating-point data.
4. The data is converted back from floating-point to BCD.
   a) First FIX(450) is used to temporarily convert the floating-point data to binary data, and then BCD(024) is used to convert the binary data to BCD data.
   b) The distance r is output to D00100.
   c) The angle θ is output to D00101.

3-15-1 FLOATING TO 16-BIT: FIX(450)

Purpose

Converts a 32-bit floating-point value to 16-bit signed binary data and places the result in the specified result word.

Ladder Symbol



S: First source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	FIX(450)
	Executed Once for Upward Differentiation	@FIX(450)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

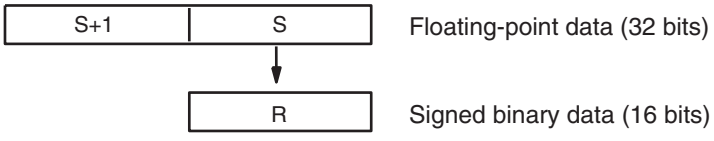
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W510	W000 to W511
Holding Bit Area	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to A958	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

FIX(450) converts the integer portion of the 32-bit floating-point number in S+1 and S (IEEE754-format) to 16-bit signed binary data and places the result in R.



Only the integer portion of the floating-point data is converted, and the fraction portion is truncated. The integer portion of the floating-point data must be within the range of –32,768 to 32,767.

Example conversions:
A floating-point value of 3.5 is converted to 3.
A floating-point value of –3.5 is converted to –3.

Flags

Name	Label	Operation
Error Flag	ER	ON if the data in S+1 and S is not a number (NaN). ON if the integer portion of S+1 and S is not within the range of –32,768 to 32,767. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of the result is ON. OFF in all other cases.

Precautions

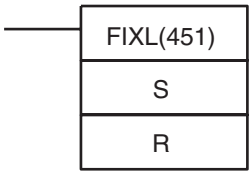
The content of S+1 and S must be floating-point data and the integer portion must be in the range of –32,768 to 32,767.

3-15-2 FLOATING TO 32-BIT: FIXL(451)

Purpose

Converts a 32-bit floating-point value to 32-bit signed binary data and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	FIXL(451)
	Executed Once for Upward Differentiation	@FIXL(451)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

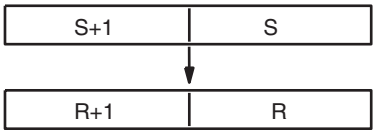
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	

Area	S	R
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –()IR15	

Description

FIXL(451) converts the integer portion of the 32-bit floating-point number in S+1 and S (IEEE754-format) to 32-bit signed binary data and places the result in R+1 and R.



Only the integer portion of the floating-point data is converted, and the fraction portion is truncated. (The integer portion of the floating-point data must be within the range of –2,147,483,648 to 2,147,483,647.)

Example conversions:
A floating-point value of 2,147,483,640.5 is converted to 2,147,483,640.
A floating-point value of –214,748,340.5 is converted to –214,748,340.

Flags

Name	Label	Operation
Error Flag	ER	ON if the data in S+1 and S is not a number (NaN). ON if the integer portion of S+1 and S is not within the range of –2,147,483,648 to 2,147,483,647. OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of R+1 is ON after execution. OFF in all other cases.

Precautions

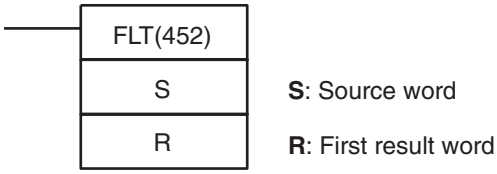
The content of S+1 and S must be floating-point data and the integer portion must be in the range of –2,147,483,648 to 2,147,483,647.

3-15-3 16-BIT TO FLOATING: FLT(452)

Purpose

Converts a 16-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	FLT(452)
	Executed Once for Upward Differentiation	@FLT(452)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

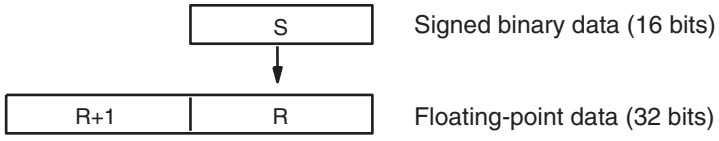
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142
Work Area	W000 to W511	W000 to W510
Holding Bit Area	H000 to H511	H000 to H510
Auxiliary Bit Area	A000 to A959	A448 to A958
Timer Area	T0000 to T4095	T0000 to T4094
Counter Area	C0000 to C4095	C0000 to C4094
DM Area	D00000 to D32767	D00000 to D32766
EM Area without bank	E00000 to E32767	E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n= 0 to C)	En_00000 to En_32766 (n= 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

FLT(452) converts the 16-bit signed binary value in S to 32-bit floating-point data (IEEE754-format) and places the result in R+1 and R. A single 0 is added after the decimal point in the floating-point result.



Only values within the range of –32,768 to 32,767 can be specified for S. To convert signed binary data outside of that range, use FLTL(453).

Example conversions:
A signed binary value of 3 is converted to 3.0.
A signed binary value of –3 is converted to –3.0.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

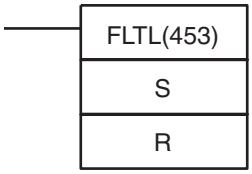
The content of S must contain signed binary data with a (decimal) value in the range of –32,768 to 32,767.

3-15-4 32-BIT TO FLOATING: FLTL(453)

Purpose

Converts a 32-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	FLTL(453)
	Executed Once for Upward Differentiation	@FLTL(453)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

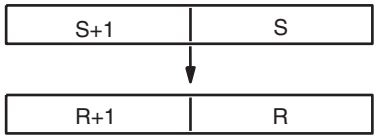
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	

Area	S	R
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

FLTL(453) converts the 32-bit signed binary value in S+1 and S to 32-bit floating-point data (IEEE754-format) and places the result in R+1 and R. A single 0 is added after the decimal point in the floating-point result.



Signed binary data within the range of –2,147,483,648 to 2,147,483,647 can be specified for S+1 and S. The floating point value has 24 significant binary digits (bits). The result will not be exact if a number greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted by FLTL(453).

Example Conversions:
A signed binary value of 16,777,215 is converted to 16,777,215.0.
A signed binary value of –16,777,215 is converted to –15,777,215.0.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

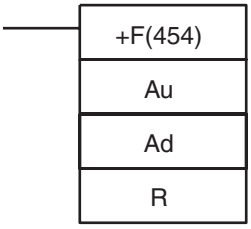
The result will not be exact if a number with an absolute value greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted.

3-15-5 FLOATING-POINT ADD: +F(454)

Purpose

Adds two 32-bit floating-point numbers and places the result in the specified result words.

Ladder Symbol



Au: First augend word

AD: First addend word

R: First result word

Variations

Variations	Executed Each Cycle for ON Condition	+F(454)
	Executed Once for Upward Differentiation	@+F(454)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

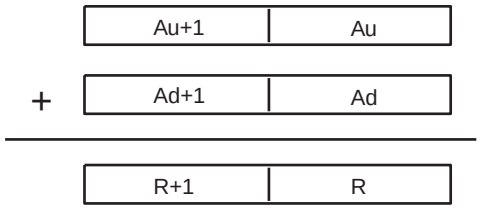
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+F(454) adds the 32-bit floating-point number in Ad+1 and Ad to the 32-bit floating-point number in Au+1 and Au and places the result in R+1 and R. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of augend and addend data will produce the results shown in the following table.

	Augend				
Addend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	
Numeral	Numeral	See note 1.	+∞ (See note 2.)	–∞ (See note 2.)	
+∞	+∞	+∞ (See note 2.)	+∞	See note 3.	
–∞	–∞	–∞ (See note 2.)	See note 3.	–∞	
NaN	See note 3.				

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.
3. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the augend or addend data is not recognized as floating-point data. ON if the augend or addend data is not a number (NaN). ON if +∞ and –∞ are added. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

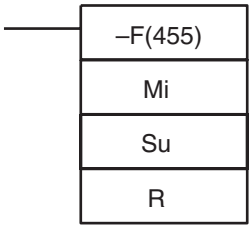
The augend (Au+1 and Au) and Addend (Ad+1 and Ad) data must be in IEEE754 floating-point data format.

3-15-6 FLOATING-POINT SUBTRACT: –F(455)

Purpose

Subtracts one 32-bit floating-point number from another and places the result in the specified result words.

Ladder Symbol



Mi: First Minuend word

Su: First Subtrahend word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–F(455)
	Executed Once for Upward Differentiation	@–F(455)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

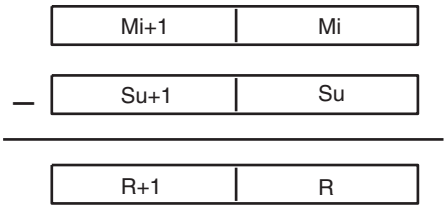
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–F(455) subtracts the 32-bit floating-point number in Su+1 and Su from the 32-bit floating-point number in Mi+1 and Mi and places the result in R+1 and R. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of minuend and subtrahend data will produce the results shown in the following table.

	Minuend				
Subtrahend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	
Numeral	Numeral	See note 1.	+∞ (See note 2.)	–∞ (See note 2.)	
+∞	–∞ (See note 2.)	–∞ (See note 2.)	See note 3.	–∞	
–∞	+∞	+∞	+∞	See note 3.	
NaN					See note 3.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. With CJ1H-CPU□□H-R CPU Units, an undetermined value will be output.
3. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the minuend or subtrahend data is not recognized as floating-point data. ON if the minuend or subtrahend is not a number (NaN). ON if +∞ is subtracted from +∞. ON if –∞ is subtracted from –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The Minuend (Mi+1 and Mi) and Subtrahend (Su+1 and Su) data must be in IEEE754 floating-point data format.

3-15-7 FLOATING-POINT MULTIPLY: ∗F(456)

Purpose

Multiplies two 32-bit floating-point numbers and places the result in the specified result words.

Ladder Symbol

Md: First Multiplicand word

Mr: First Multiplier word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*F(456)
	Executed Once for Upward Differentiation	@*F(456)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

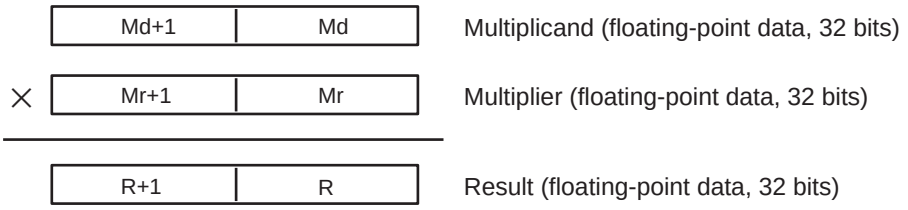
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

*F(456) multiplies the 32-bit floating-point number in Md+1 and Md by the 32-bit floating-point number in Mr+1 and Mr and places the result in R+1 and R. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of multiplicand and multiplier data will produce the results shown in the following table.

	Multiplicand				
Multiplier	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	0	See note 3.	See note 3.	
Numeral	0	See note 1.	+/–∞ (See note 2.)	+/–∞ (See note 2.)	
+∞	See note 3.	+/–∞ (See note 2.)	+∞	–∞	
–∞	See note 3.	+/–∞ (See note 2.)	–∞	+∞	
NaN	See note 3.				

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.
3. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the multiplicand or multiplier data is not recognized as floating-point data. ON if the multiplicand or multiplier is not a number (NaN). ON if +∞ and 0 are multiplied. ON if –∞ and 0 are multiplied. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

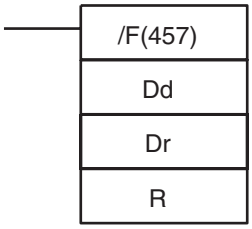
The Multiplicand (Md+1 and Md) and Multiplier (Mr+1 and Mr) data must be in IEEE754 floating-point data format.

3-15-8 FLOATING-POINT DIVIDE: /F(457)

Purpose

Divides one 32-bit floating-point number by another and places the result in the specified result words.

Ladder Symbol



Dd: First Dividend word

Dr: First Divisor word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/F(457)
	Executed Once for Upward Differentiation	@/F(457)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

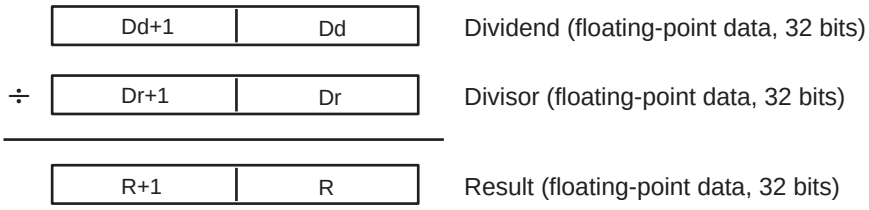
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

/F(457) divides the 32-bit floating-point number in Dd+1 and Dd by the 32-bit floating-point number in Dr+1 and Dr and places the result in R+1 and R. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of dividend and divisor data will produce the results shown in the following table.

	Multiplicand				
Multiplier	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	See note 4.	+/–∞ (See note 3.)	+∞ (See note 3.)	–∞ (See note 3.)	
Numeral	0	See note 2.	+/–∞	+/–∞	
+∞	0	0 (See notes 1 and 3.)	See note 4.	See note 4.	
–∞	0	0 (See notes 1 and 3.)	See note 4.	See note 4.	
NaN	See note 4.				

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.
4. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the dividend or divisor data is not recognized as floating-point data. ON if the dividend or divisor is not a number (NaN). ON if the dividend and divisor are both 0. ON if the dividend and divisor are both +∞ or –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

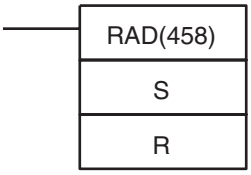
The Dividend (Dd+1 and Dd) and Divisor (Dr+1 and Dr) data must be in IEEE754 floating-point data format.

3-15-9 DEGREES TO RADIANS: RAD(458)

Purpose

Converts a 32-bit floating-point number from degrees to radians and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	RAD(458)
	Executed Once for Upward Differentiation	@RAD(458)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

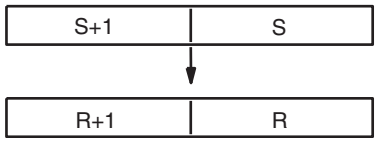
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

RAD(458) converts the 32-bit floating-point number in S+1 and S from degrees to radians and places the result in R and R+1. (The floating point source data must be in IEEE754 format.)



Source (degrees, 32-bit floating-point data)

Result (radians, 32-bit floating-point data)

Degrees are converted to radians by means of the following formula:

Degrees × π/180 = radians

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

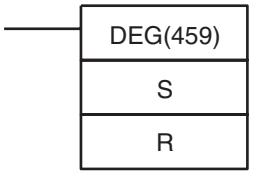
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-10 RADIANS TO DEGREES: DEG(459)

Purpose

Converts a 32-bit floating-point number from radians to degrees and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations	**Executed Each Cycle for ON Condition**	DEG(459)
	Executed Once for Upward Differentiation	@DEG(459)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

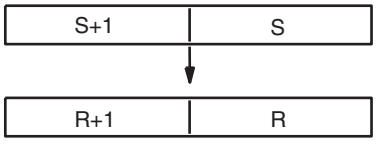
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to+2047 ,IR0 to –2048 to+2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

DEG(459) converts the 32-bit floating-point number in S+1 and S from radians to degrees and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



Source (radians, 32-bit floating-point data)

Result (degrees, 32-bit floating-point data)

Radians are converted to degrees by means of the following formula:

Radians $\times$ 180/π = degrees

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as $\pm\infty$.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

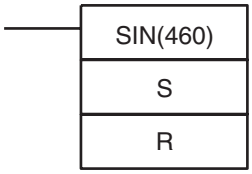
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-11 SINE: SIN(460)

Purpose

Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	SIN(460)
	Executed Once for Upward Differentiation	@SIN(460)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

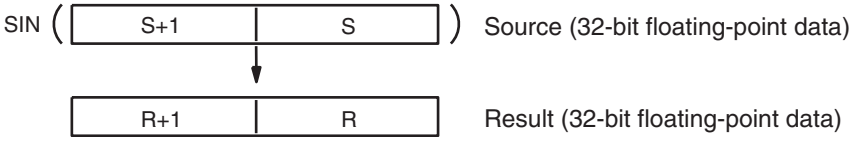
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	

Area	S	R
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

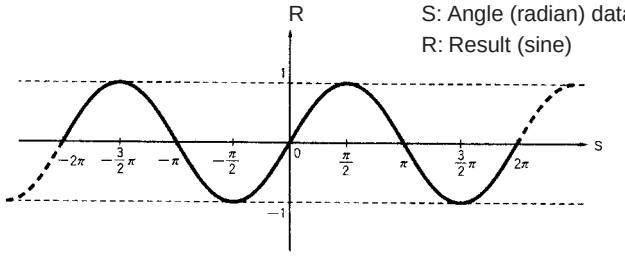
Description

SIN(460) calculates the sine of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R.
(The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in S+1 and S. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting from degrees to radians, see *3-15-22 LOGARITHM: LOG(468) DEGREES TO RADIANS: RAD(458)*.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF

Name	Label	Operation
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

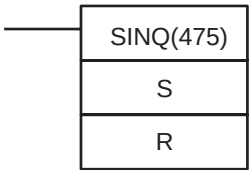
Precautions The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-12 HIGH-SPEED SINE: SINQ(475)

Purpose Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words.

Note These instructions can be used in the CJ1-H-R CPU Units only.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	SINQ(475)
	Executed Once for Upward Differentiation	@SINQ(475)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

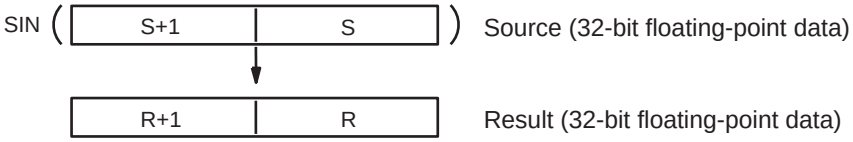
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	Can be specified.	---
Data Registers	---	

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

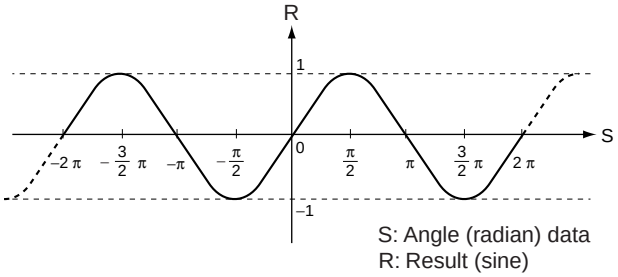
Description

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. 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. 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).*

The following diagram shows the relationship between the angle and result.



Precautions

SINQ(475) differs from SIN(460) in the following respects:

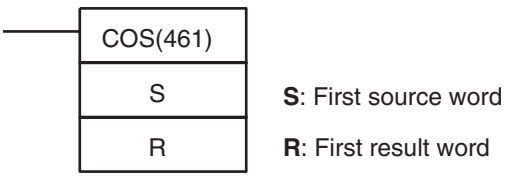
- The instruction has improved performance.
- The instruction length is 8 steps.
- The Condition Flags are not refreshed.
- An unpredictable value will be output if the angle data is out-of-range.
- The data cannot be input or output at a Programming Console. A question mark will be displayed.

3-15-13 COSINE: COS(461)

Purpose

Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words.

Ladder Symbol



Variations

Variations	Executed Each Cycle for ON Condition	COS(461)
	Executed Once for Upward Differentiation	@COS(461)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

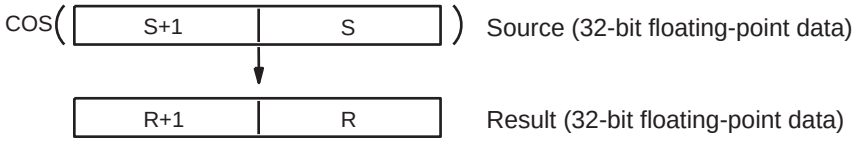
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

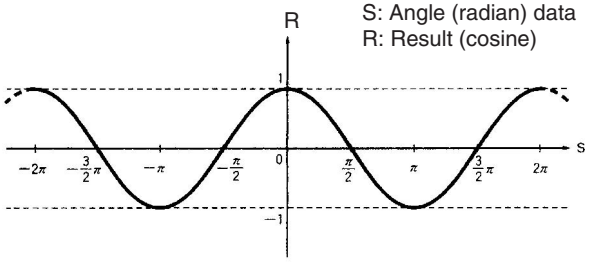
COS(461) calculates the cosine of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in S+1 and S. If the angle is outside of the range –65,535 to 65,535, an error will occur and the

instruction will not be executed. For information on converting from degrees to radians, see *3-15-9 DEGREES TO RADIANS: RAD(458)*.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

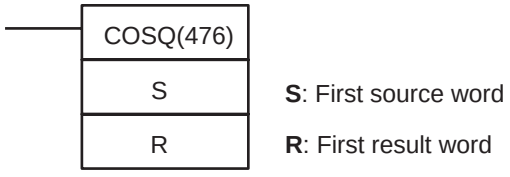
Precautions The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-14 HIGH-SPEED COSINE: COSQ(476)

Purpose Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words.

Note These instructions can be used in the CJ1-H-R CPU Units only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	COSQ(476)
	Executed Once for Upward Differentiation	@COSQ(476)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

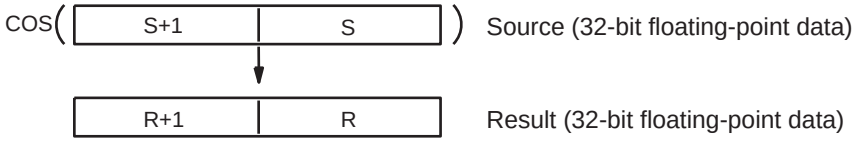
Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	Can be specified.	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

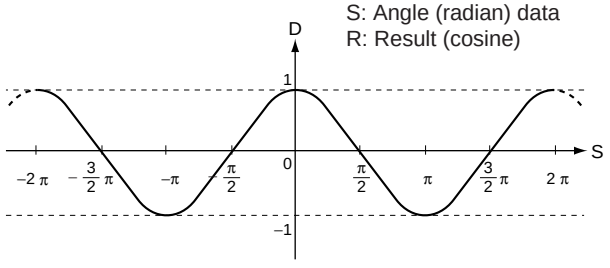
Description

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. 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. 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).*

The following diagram shows the relationship between the angle and result.



Precautions

COSQ(476) differs from COS(461) in the following respects:

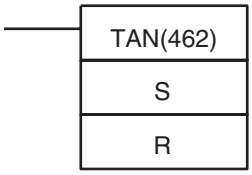
- The instruction has improved performance.
- The instruction length is 8 steps.
- The Condition Flags are not refreshed.
- An unpredictable value will be output if the angle data is out-of-range.
- The data cannot be input or output at a Programming Console. A question mark will be displayed.

3-15-15 TANGENT: TAN(462)

Purpose

Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	TAN(462)
	Executed Once for Upward Differentiation	@TAN(462)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

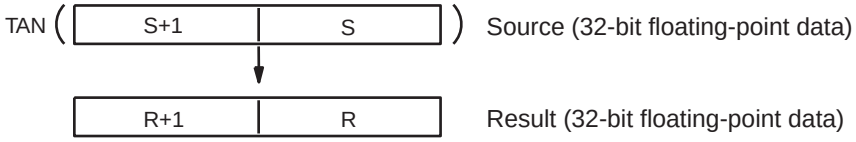
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---

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

Description

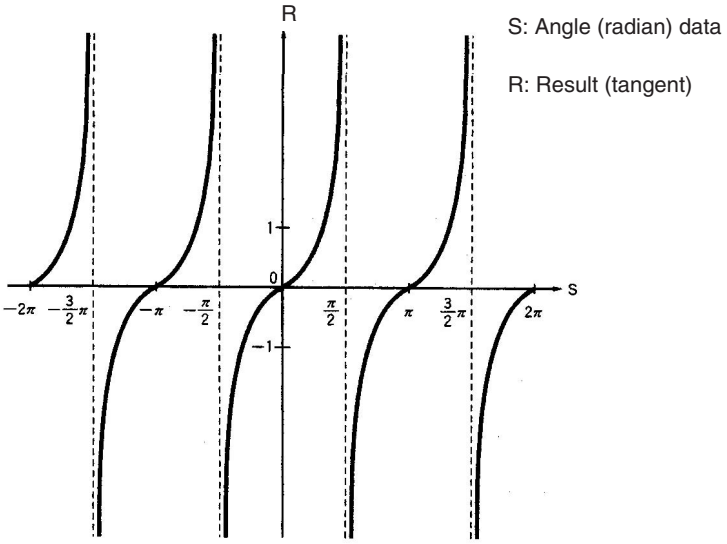
TAN(462) calculates the tangent of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in S+1 and S. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting from degrees to radians, see *3-15-9 DEGREES TO RADIANS: RAD(458)*.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data 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.

Name	Label	Operation
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

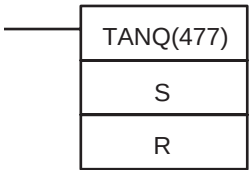
Precautions The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-16 HIGH-SPEED TANGENT: TANQ(477)

Purpose Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words.

Note These instructions can be used in the CJ1-H-R CPU Units only.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	TANQ(477)
	Executed Once for Upward Differentiation	@TANQ(477)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

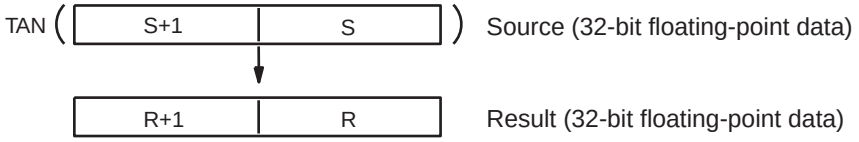
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	Can be specified.	---

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

Description

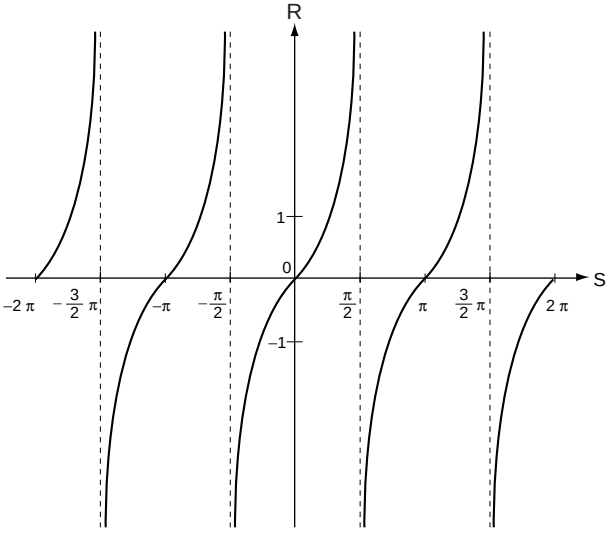
TANQ(477) calculates the tangent of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in S+1 and S. 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. 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 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 angle and result.



S: Angle (radian) data
R: Result (tangent)

Precautions

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.

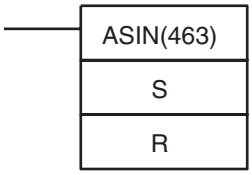
- An unpredictable value will be output if the angle data is nπ/2 (n = ..., -3, -1, 1, 3.....).

3-15-17 ARC SINE: ASIN(463)

Purpose

Calculates the arc sine of a 32-bit floating-point number and places the result in the specified result words. (The arc sine function is the inverse of the sine function; it returns the angle that produces a given sine value between –1 and 1.)

Ladder Symbol



S: First source word

R: First result word

Variations

Variations	**Executed Each Cycle for ON Condition**	ASIN(463)
	Executed Once for Upward Differentiation	@ASIN(463)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	

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

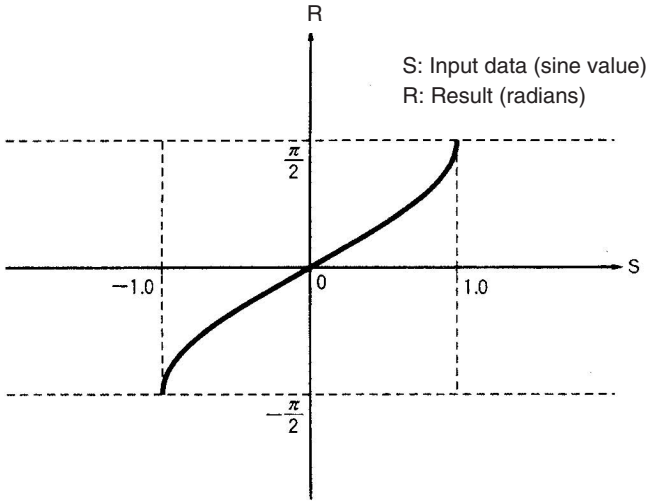
Description

ASIN(463) computes the angle (in radians) for a sine value expressed as a 32-bit floating-point number in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)

The source data must be between –1.0 and 1.0. If the absolute value of the source data exceeds 1.0, an error will occur and the instruction will not be executed.

The result is output to words R+1 and R as an angle (in radians) within the range of $-\pi/2$ to $\pi/2$.

The following diagram shows the relationship between the input data and result.



Flags

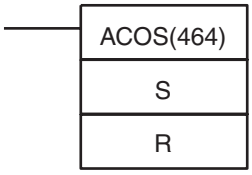
Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-18 ARC COSINE: ACOS(464)

Purpose Calculates the arc cosine of a 32-bit floating-point number and places the result in the specified result words. (The arc cosine function is the inverse of the cosine function; it returns the angle that produces a given cosine value between –1 and 1.)

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	ACOS(464)
	Executed Once for Upward Differentiation	@ACOS(464)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

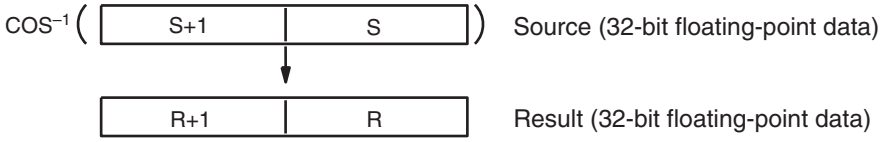
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	

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

Description

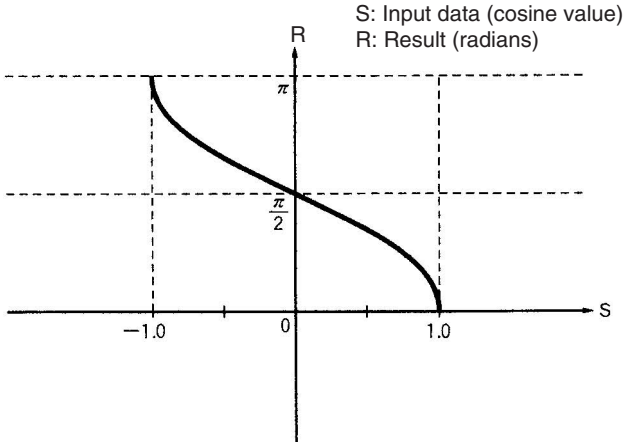
ACOS(464) computes the angle (in radians) for a cosine value expressed as a 32-bit floating-point number in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



The source data must be between –1.0 and 1.0. If the absolute value of the source data exceeds 1.0, an error will occur and the instruction will not be executed.

The result is output to words R+1 and R as an angle (in radians) within the range of 0 to π.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

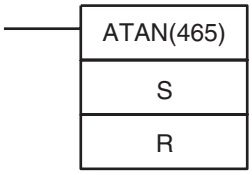
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-19 ARC TANGENT: ATAN(465)

Purpose

Calculates the arc tangent of a 32-bit floating-point number and places the result in the specified result words. (The arc tangent function is the inverse of the tangent function; it returns the angle that produces a given tangent value.)

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ATAN(465)
	Executed Once for Upward Differentiation	@ATAN(465)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

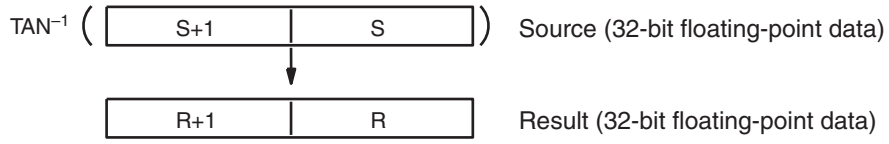
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

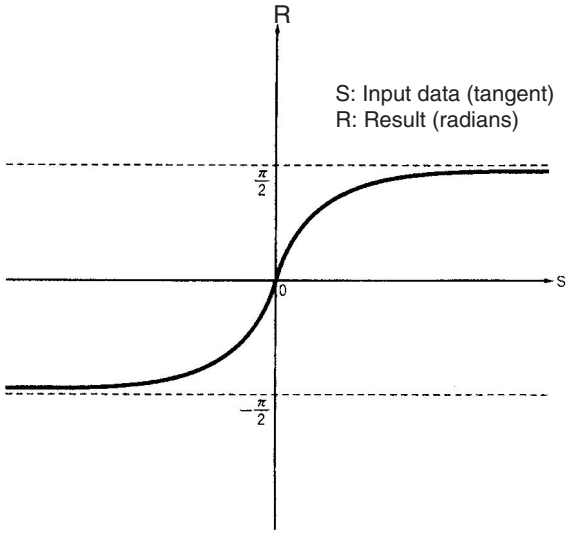
Description

ATAN(465) computes the angle (in radians) for a tangent value expressed as a 32-bit floating-point number in S+1 and S and places the result in R+1 and R.
(The floating point source data must be in IEEE754 format.)



The result is output to words R+1 and R as an angle (in radians) within the range of $-\pi/2$ to $\pi/2$.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

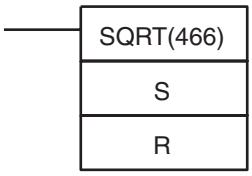
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-20 SQUARE ROOT: SQRT(466)

Purpose

Calculates the square root of a 32-bit floating-point number and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	SQRT(466)
	Executed Once for Upward Differentiation	@SQRT(466)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

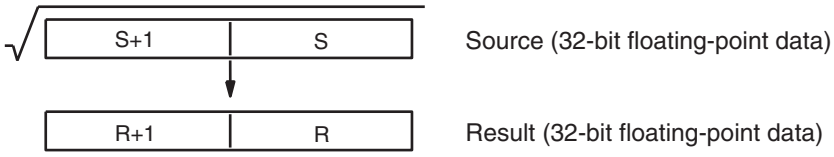
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

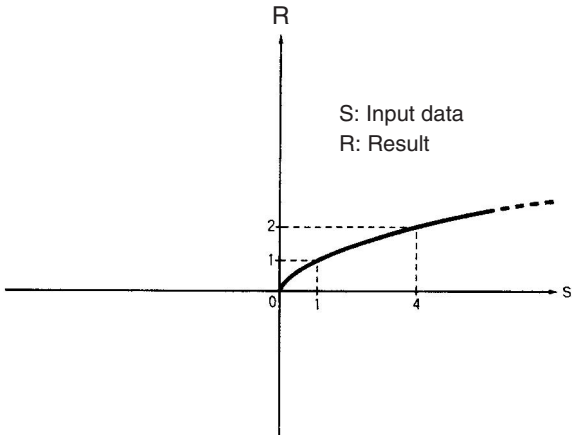
SQRT(466) calculates the square root of the 32-bit floating-point number in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



The source data must be positive; if it is negative, an error will occur and the instruction will not be executed.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is negative. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	OFF
Negative Flag	N	OFF

Precautions

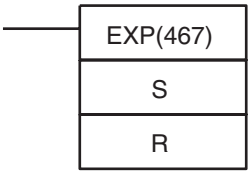
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-21 EXPONENT: EXP(467)

Purpose

Calculates the natural (base e) exponential of a 32-bit floating-point number and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	EXP(467)
	Executed Once for Upward Differentiation	@EXP(467)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

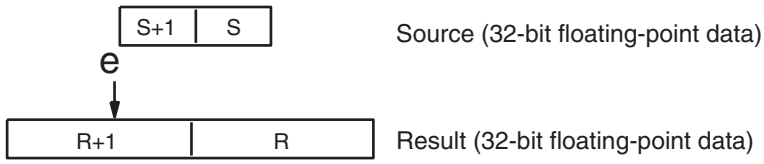
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to 4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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.

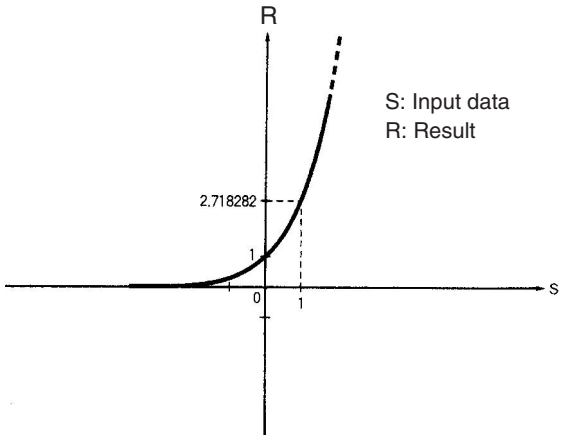


If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Note The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	OFF

Precautions

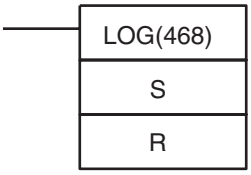
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-22 LOGARITHM: LOG(468)

Purpose

Calculates the natural (base e) logarithm of a 32-bit floating-point number and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations	**Executed Each Cycle for ON Condition**	LOG(468)
	Executed Once for Upward Differentiation	@LOG(468)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

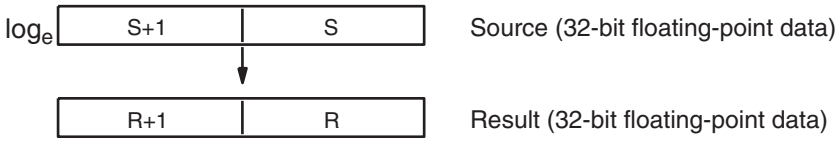
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

LOG(468) calculates the natural (base e) logarithm of the 32-bit floating-point number in S+1 and S and places the result in R+1 and R.

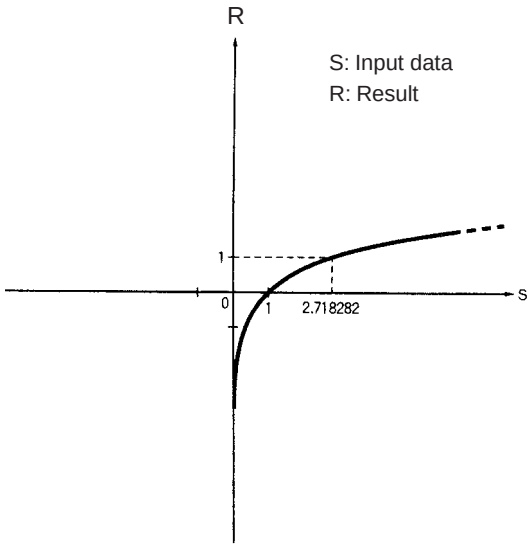


The source data must be positive; if it is negative, an error will occur and the instruction will not be executed.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

Note The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is negative. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

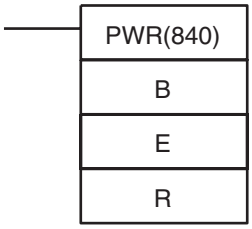
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-15-23 EXPONENTIAL POWER: PWR(840)

Purpose

Raises a 32-bit floating-point number to the power of another 32-bit floating-point number.

Ladder Symbol



B: First base word

E: First exponent word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	PWR(840)
	Executed Once for Upward Differentiation	@PWR(840)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

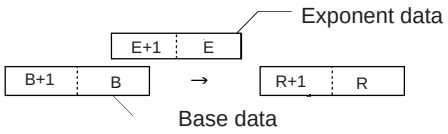
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	B	E	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W510		
Holding Bit Area	H000 to H510		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T4094		
Counter Area	C0000 to C4094		
DM Area	D00000 to D32766		
EM Area without bank	E00000 to E32766		
EM Area with bank	En_00000 to En_32766 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#00000000 to #FFFFFFFF (binary)		---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

PWR(840) raises the 32-bit floating-point number in B+1 and B to the power of the 32-bit floating-point number in E+1 and E. In other words, PWR(840) calculates X^Y (X = B+1 and B; Y = E+1 and E).



For example, when the base words (B+1 and B) contain 3.1 and the exponent words (E+1 and E) contain 3, the result is 3.1^3 or 29.791.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if the base (B+1 and B) or exponent (E+1 and E) is not recognized as floating-point data. ON if the base (B+1 and B) or exponent (E+1 and E) is not a number (NaN). ON if the base (B+1 and B) is 0 and the exponent (E+1 and E) is less than 0. (Division by 0) ON if the base (B+1 and B) is negative and the exponent (E+1 and E) is non-integer. (Root of a negative number) OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The base (B+1 and B) and the exponent (E+1 and E) must be in IEEE754 floating-point data format.

3-15-24 Single-precision Floating-point Comparison Instructions

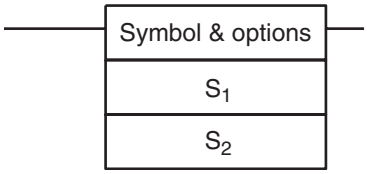
Purpose

These input comparison instructions compare two single-precision floating point values (32-bit IEEE754 constants and/or the contents of specified words) and create an ON execution condition when the comparison condition is true.

These instructions are supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Note Refer to *3-7-1 Input Comparison Instructions (300 to 328)* for details on the signed and unsigned binary input comparison instructions and *3-16-21 Double-precision Floating-point Input Instructions* for details on double-precision floating-point input comparison instructions.

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations	Creates ON Each Cycle Comparison is True	Input comparison instruction
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	
Data Registers	---	
Index Registers	IR0 to IR15 (for unsigned data only)	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

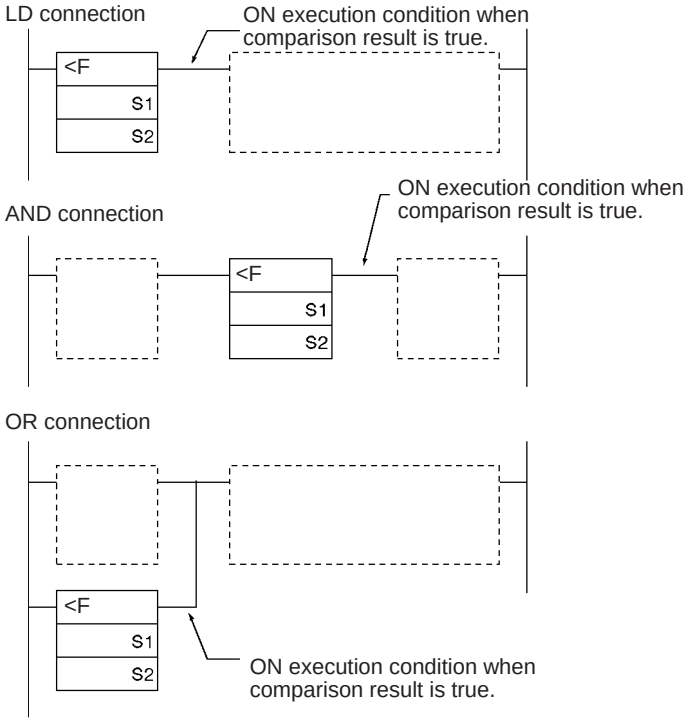
Description

The input comparison instruction compares the data specified in S_1 and S_2 as single-precision floating point values (32-bit IEEE754 data) and creates an ON execution condition when the comparison condition is true. When the data is stored in words, S_1 and S_2 specify the first of two words containing the 32-bit data. It is also possible to input the floating-point data as an 8-digit hexadecimal constant.

Inputting the Instructions

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

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



Options

With the three input types and six symbols, there are 18 different possible combinations.

Symbol		Option (data format)
=	(Equal)	F: Single-precision floating-point data
< >	(Not equal)	
<	(Less than)	
<=	(Less than or equal)	
>	(Greater than)	
>=	(Greater than or equal)	

Summary of Input Comparison Instructions

The following table shows the function codes, mnemonics, names, and functions of the 18 single-precision floating-point input comparison instructions. (C1=S_1+1, S_1 and C2=S_2+1, S_2.)

Code	Mnemonic	Name	Function
329	LD=F	LOAD FLOATING EQUAL	True if C1 = C2
	AND=F	AND FLOATING EQUAL	
	OR=F	OR FLOATING EQUAL	

Code	Mnemonic	Name	Function
330	LD<>F	LOAD FLOATING NOT EQUAL	True if C1 ≠ C2
	AND<>F	AND FLOATING NOT EQUAL	
	OR<>F	OR FLOATING NOT EQUAL	
331	LD<F	LOAD FLOATING LESS THAN	True if C1 < C2
	AND<F	AND FLOATING LESS THAN	
	OR<F	OR FLOATING LESS THAN	
332	LD<=F	LOAD FLOATING LESS THAN OR EQUAL	True if C1 ≤ C2
	AND<=F	AND FLOATING LESS THAN OR EQUAL	
	OR<=F	OR FLOATING LESS THAN OR EQUAL	
333	LD>F	LOAD FLOATING GREATER THAN	True if C1 > C2
	AND>F	AND FLOATING GREATER THAN	
	OR>F	OR FLOATING GREATER THAN	
325	LD>=F	LOAD FLOATING GREATER THAN OR EQUAL	True if C1 ≥ C2
	AND>=F	AND FLOATING GREATER THAN OR EQUAL	
	OR>=F	OR FLOATING GREATER THAN OR EQUAL	

Flags

Name	Label	Operation
Error Flag	ER	OFF
Greater Than Flag	>	ON if S_1+1, S_1 > S_2+1, S_2. OFF in all other cases.
Greater Than or Equal Flag	> =	ON if S_1+1, S_1 ≥ S_2+1, S_2. OFF in all other cases.
Equal Flag	=	ON if S_1+1, S_1 = S_2+1, S_2. OFF in all other cases.
Not Equal Flag	=	ON if S_1+1, S_1 ≠ S_2+1, S_2. OFF in all other cases.
Less Than Flag	<	ON if S_1+1, S_1 < S_2+1, S_2. OFF in all other cases.
Less Than or Equal Flag	< =	ON if S_1+1, S_1 ≤ S_2+1, S_2. OFF in all other cases.
Negative Flag	N	Unchanged

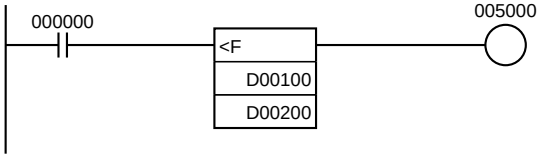
Precautions

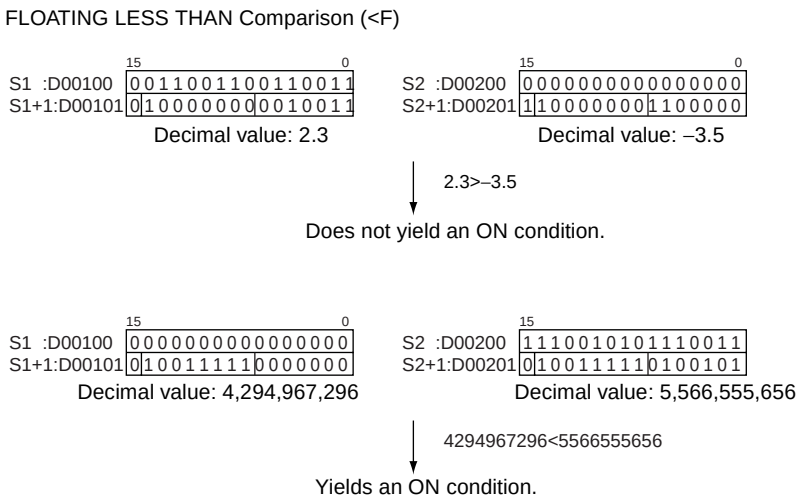
Input comparison instructions cannot be used as right-hand instructions, i.e., another instruction must be used between them and the right bus bar.

Example

AND FLOATING LESS THAN: AND<F(331)

When CIO 000000 is ON in the following example, the floating point data in D00101, D00100 is compared to the floating point data in D00201, D00200. If the content of D00101, D00100 is less than that of D00201, D00200, execution proceeds to the next line and CIO 005000 is turned ON. If the content of D00101, D00100 is not less than that of D00201, D00200, execution does not proceed to the next instruction line.





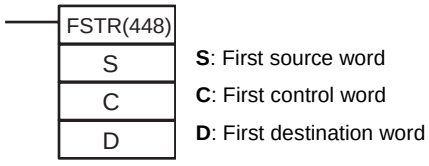
3-15-25 FLOATING-POINT TO ASCII: FSTR(448)

Purpose

Expresses a 32-bit floating-point value (IEEE754-format) in standard decimal notation or scientific notation and converts that value to ASCII text.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
C: First control word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FSTR(448)
	Executed Once for Upward Differentiation	@FSTR(448)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

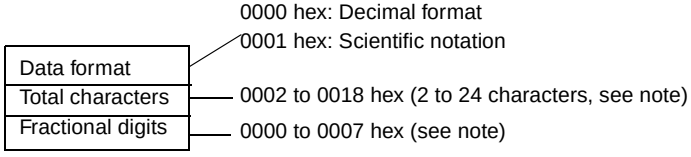
Area	S	C	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6141	CIO 0000 to CIO 6143
Work Area	W000 to W510	W000 to W509	W000 to W511
Holding Bit Area	H000 to H510	H000 to H509	H000 to H511
Auxiliary Bit Area	A000 to A958	A000 to A957	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4093	T0000 to T4095
Counter Area	C0000 to C4094	C0000 to C4093	C0000 to C4095
DM Area	D00000 to D32766	D00000 to D32765	D00000 to D32767
EM Area without bank	E00000 to E32766	E00000 to E32765	E00000 to E32767

Area	S	C	D
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32765 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	#00000000 to #FFFFFFFF (binary)	---	
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –()IR15 ,IR0 to ,IR15		

Description

FSTR(448) expresses the 32-bit floating-point number in S+1 and S (IEEE754-format) in decimal notation or scientific notation according to the control data in words C to C+2, converts the number to ASCII text, and outputs the result to the destination words starting at D.

The following diagram shows the contents of the 3 control words.

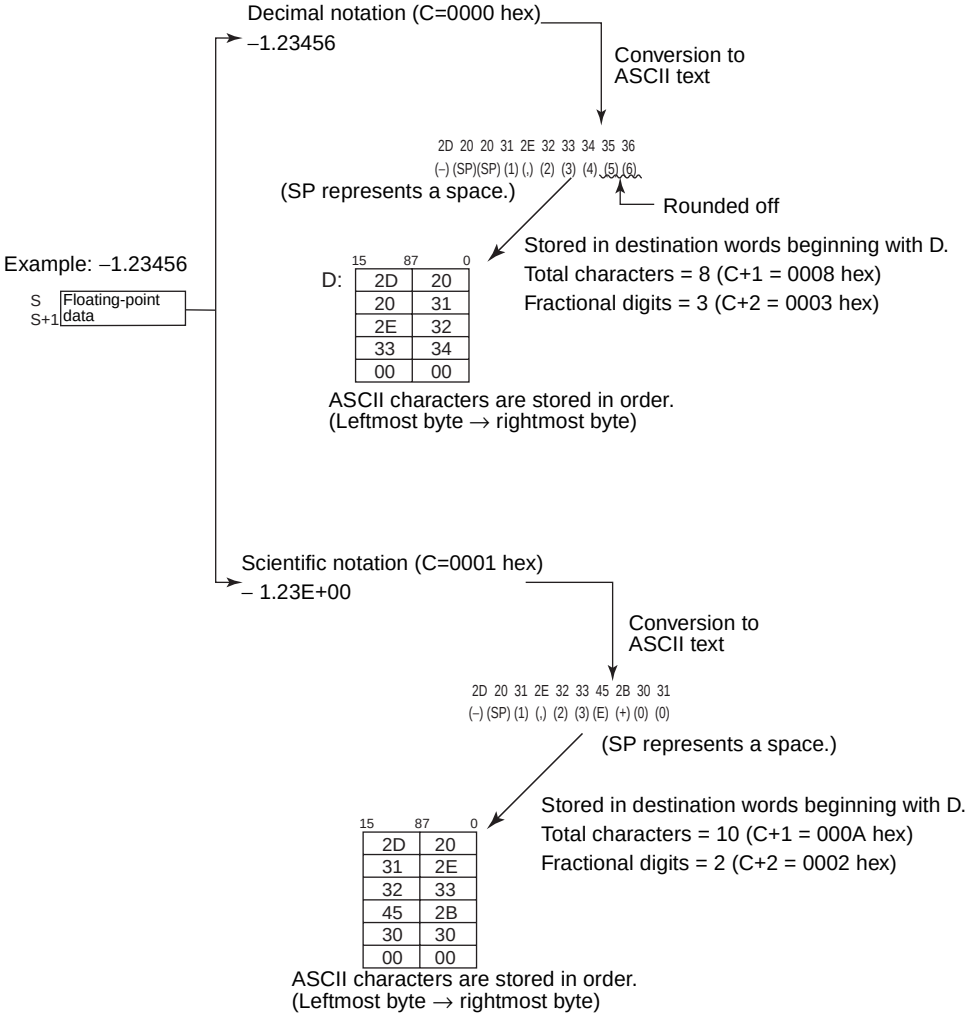


Note: There are limits on the total number of characters and the number of fractional digits. See *Limits on the Number of ASCII Characters on page 643* for details.

- The content of C (Data format) specifies whether to express the number in S+1, S in decimal notation or scientific notation.
  - Decimal notation
    Expresses a real number as an integer and fractional part.
    Example: 124.56
  - Scientific notation
    Expresses a real number as an integer part, fractional part, and exponent part.
    Example: 1.2456E-2 (1.2456×10^{-2})

- The content of C+1 (Total characters) specifies the number of ASCII characters after conversion including the sign symbol, numbers, decimal point and spaces.
- The content of C+2 (Fractional digits) specifies the number of digits (characters) below the decimal point.

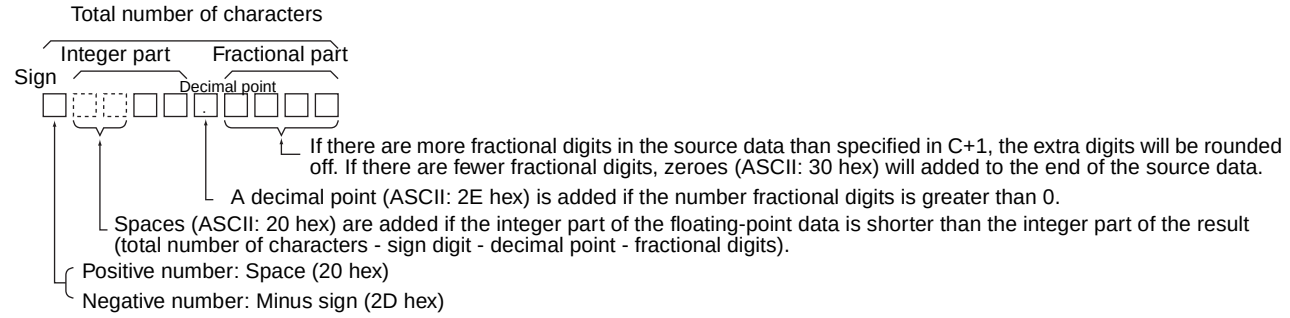
The ASCII text is stored in D and subsequent words in the following order: leftmost byte of D, rightmost byte of D, leftmost byte of D+1, rightmost byte of D+1, etc.



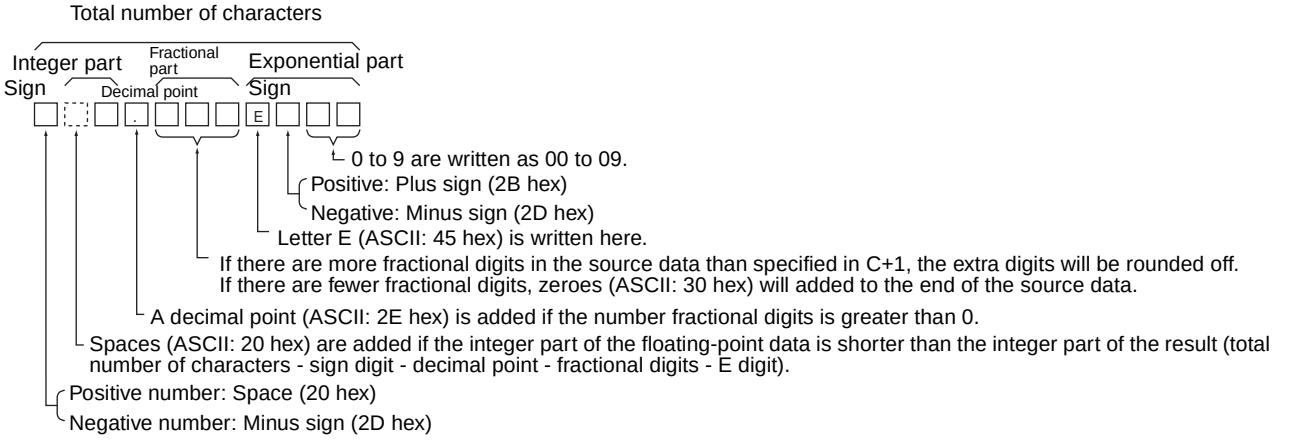
Storage of ASCII Text

After the floating-point number is converted to ASCII text, the ASCII characters are stored in the destination words beginning with D, as shown in the following diagrams. Different storage methods are used for decimal notation and scientific notation.

Decimal notation (C=0000 hex)



Scientific notation (C=0001 hex)



Note Either one or two bytes of zeroes are added to the end of ASCII text as an end code.
Total number of characters odd: 00 hex is stored after the ASCII text.
Total number of characters even: 0000 hex is stored after the ASCII text.

Limits on the Number of ASCII Characters

There are limits on the number of ASCII characters in the converted number. The Error Flag will be turned ON if the number of characters exceeds the maximum allowed.

1. Limits on the Total Number of ASCII Characters
   a) Decimal Notation (C = 0000 hex)
      - When there is no fractional part (C+2 = 0000 hex):
        $2 \leq$ Total Characters ≤ 24
      - When there is a fractional part (C+2 = 0001 to 0007 hex):
        (Fractional digits + 3) $\leq$ Total Characters ≤ 24

   b) Scientific Notation (C = 0001 hex)
      - When there is no fractional part (C+2 = 0000 hex):
        $6 \leq$ Total Characters ≤ 24
      - When there is a fractional part (C+2 = 0001 to 0007 hex):
        (Fractional digits + 7) $\leq$ Total Characters ≤ 24
2. Limits on the Number of Digits in the Integer Part

a) Decimal Notation (C = 0000 hex)
- When there is no fractional part (C+2 = 0000 hex):
  1 ≤ Number of Integer Digits ≤ 24
- When there is a fractional part (C+2 = 0001 to 0007 hex):
  1 ≤ Number of Integer Digits ≤ (24 – Fractional digits – 2)

b) Scientific Notation (C = 0001 hex)
1 digit (fixed)

3. Limits on the Number of Digits in the Fractional Part
   a) Decimal Notation (C = 0000 hex)
   - Fractional Digits ≤ 7
   - Also: Fractional Digits ≤ (Total Number of ASCII Characters – 3)

   b) Scientific Notation (C = 0001 hex)
   - Fractional Digits ≤ 7
   - Also: Fractional Digits ≤ (Total Number of ASCII Characters – 3)

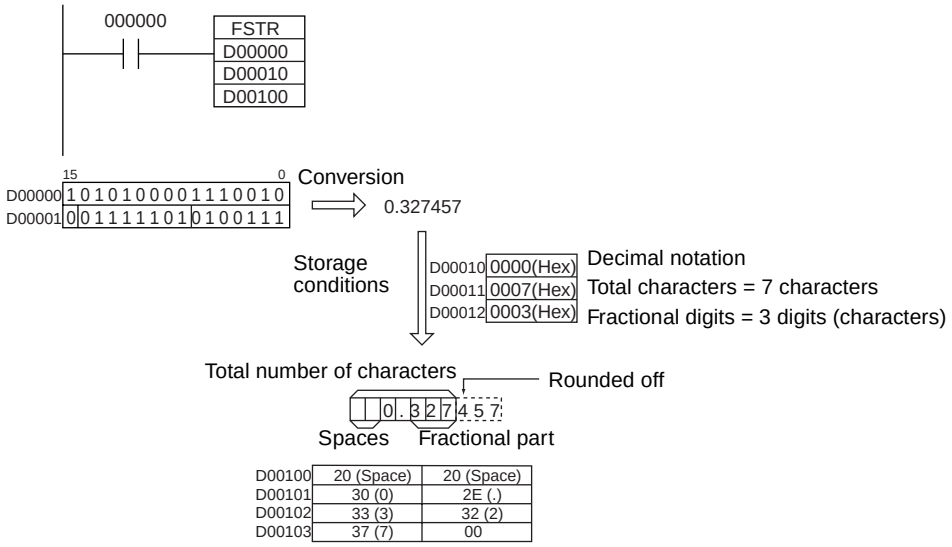
Flags

Name	Label	Operation
Error Flag	ER	ON if the data in S+1 and S is not a valid floating-point number (NaN). ON if the data in S+1 and S is +∞ or –∞. ON if the Data Format setting in C is not 0000 or 0001. ON if the Total Characters setting in C+1 is not within the allowed range. (See *1. Limits on the Total Number of ASCII Characters* above for details.) ON if the Fractional Digits setting in C+2 is not within the allowed range. (See *3. Limits on the Number of Digits in the Fractional Part* above for details.) OFF in all other cases.
Equals Flag	=	ON if the conversion result is 0. OFF in all other cases.

Examples

Converting to ASCII Text in Decimal Notation

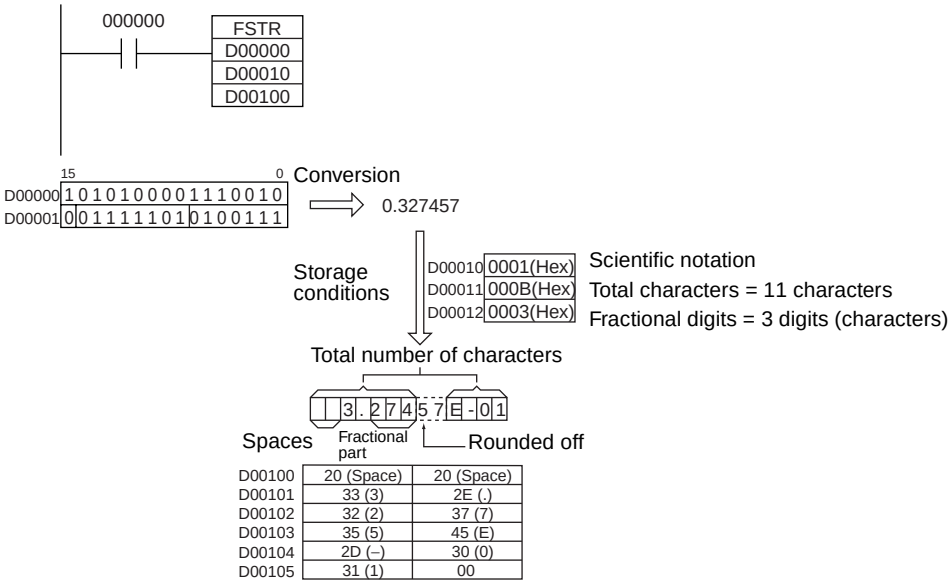
When CIO 000000 is ON in the following example, FSTR(448) converts the floating-point data in D00001 and D00000 to decimal-notation ASCII text and writes the ASCII text to the destination words beginning with D00100. The contents of the control words (D00010 to D00012) specify the details on the data format (decimal notation, 7 characters total, 3 fractional digits).



D00100	20 (Space)	20 (Space)
D00101	30 (0)	2E (.)
D00102	33 (3)	32 (2)
D00103	37 (7)	00

Converting to ASCII Text in Scientific Notation

When CIO 000000 is ON in the following example, FSTR(448) converts the floating-point data in D00001 and D00000 to scientific-notation ASCII text and writes the ASCII text to the destination words beginning with D00100. The contents of the control words (D00010 to D00012) specify the details on the data format (scientific notation, 11 characters total, 3 fractional digits).



D00100	20 (Space)	20 (Space)
D00101	33 (3)	2E (.)
D00102	32 (2)	37 (7)
D00103	35 (5)	45 (E)
D00104	2D (–)	30 (0)
D00105	31 (1)	00

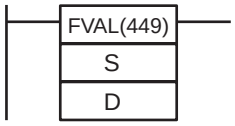
3-15-26 ASCII TO FLOATING-POINT: FVAL(449)

Purpose

Converts a number expressed in ASCII text (decimal or scientific notation) to a 32-bit floating-point value (IEEE754-format) and outputs the floating-point value to the specified words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FVAL(449)
	Executed Once for Upward Differentiation	@FVAL(449)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Timer Input Turns OFF before Completion Flag

Description

FVAL(449) converts the specified ASCII text number (starting at word S) to a 32-bit floating-point number (IEEE754-format) and outputs the result to the destination words starting at D.

FVAL(449) can convert ASCII text in decimal or scientific notation if it meets the following conditions:

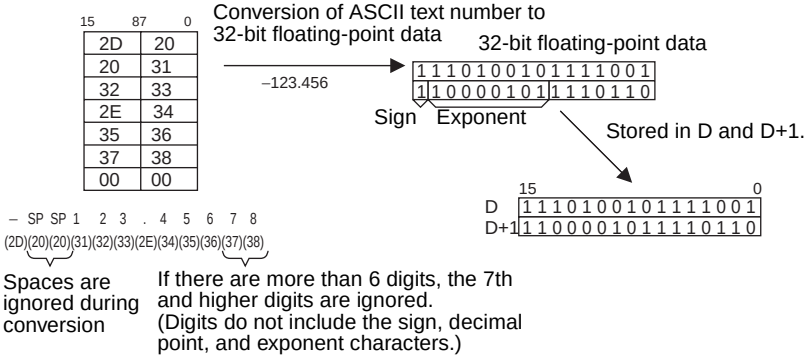
Up to 6 characters are valid, excluding the sign, decimal point, and exponent. Any characters beyond the 6th character will be ignored.

- 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.

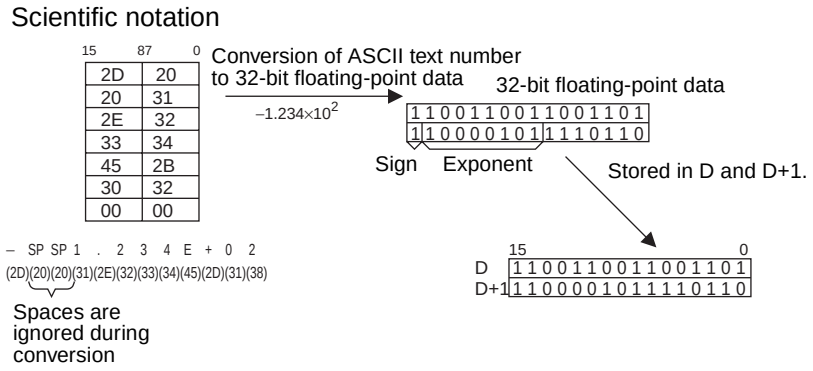
The ASCII text must be stored in S and subsequent words in the following order: leftmost byte of S, rightmost byte of S, leftmost byte of S+1, rightmost byte of S+1, etc.

Decimal notation



Spaces are ignored during conversion

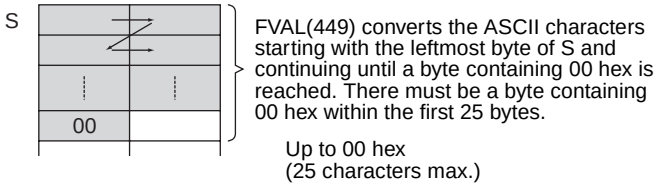
If there are more than 6 digits, the 7th and higher digits are ignored. (Digits do not include the sign, decimal point, and exponent characters.)



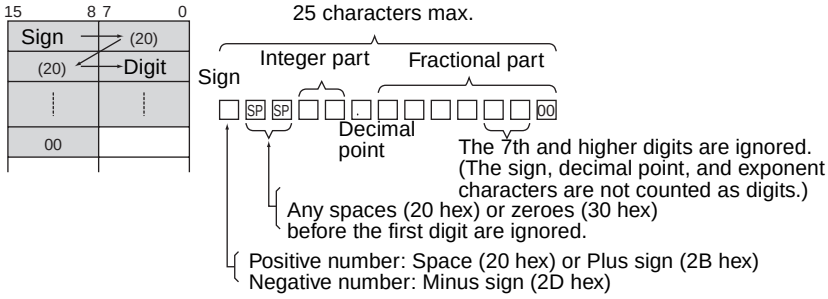
Storage of ASCII Text

The following diagrams show how the ASCII text number is converted to floating-point data. Different conversion methods are used for numbers stored with decimal notation and scientific notation.

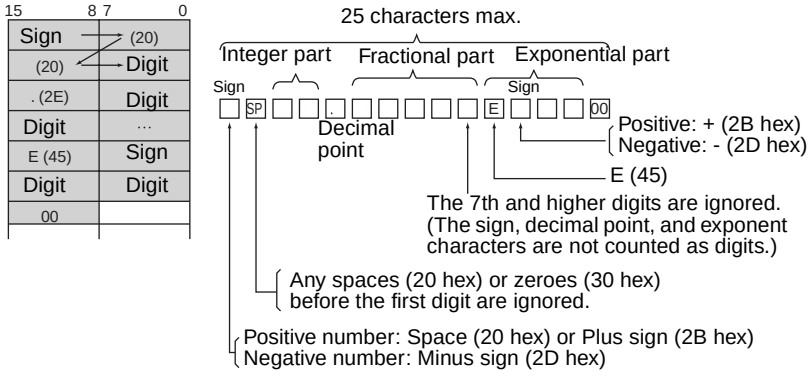
ASCII Character Storage



Decimal notation



Scientific notation



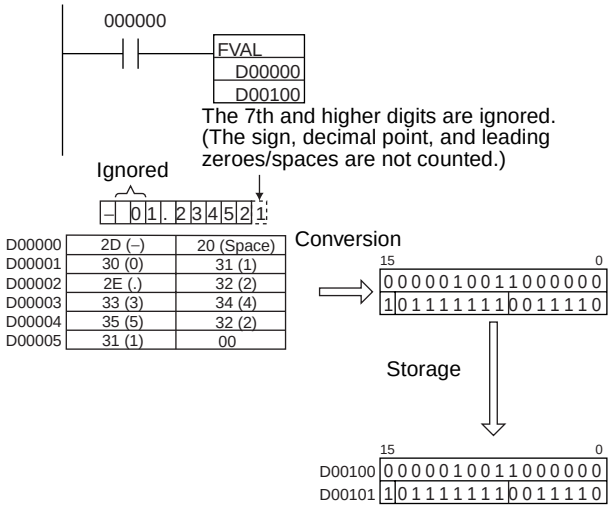
Flags

Name	Label	Operation
Error Flag	ER	ON if the digits (integer and fractional parts) in the source data starting at S are not 30 to 39 hex (0 to 9). ON if the first two digits of the exponential part do not contain 45 and 2B hex (E+) or 45 and 2D hex (E-). (integer and fractional parts) in the source data starting at S are not 30 to 39 hex (0 to 9). ON if there are two or more exponential parts in the source data. ON if the data is +∞ or –∞ after conversion. ON is the are 0 characters in the text data. ON if a byte containing 00 hex is not found within the first 25 characters. OFF in all other cases.
Equals Flag	=	ON if the conversion result is 0. OFF in all other cases.

Examples

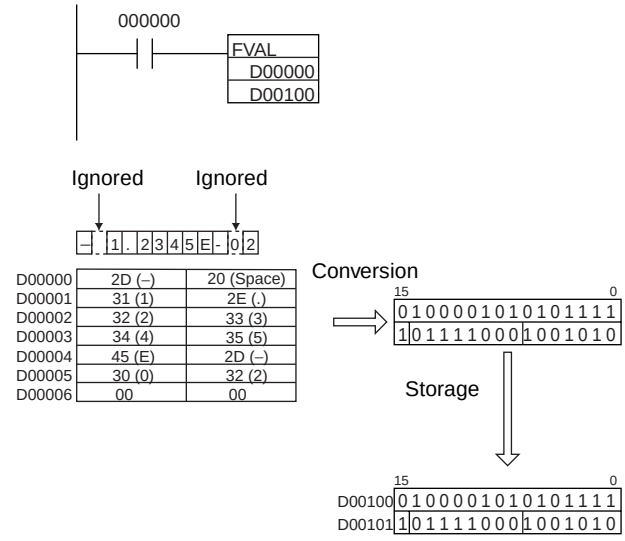
Converting ASCII Text in Decimal Notation to Floating-point Data

When CIO 000000 is ON in the following example, FVAL(449) converts the specified decimal-notation ASCII text number in the source words starting at D00000 to floating-point data and writes the result to destination words D00100 and D00101.



Converting ASCII Text in Scientific Notation

When CIO 000000 is ON in the following example, FVAL(449) converts the specified scientific-notation ASCII text number in the source words starting at D00000 to floating-point data and writes the result to destination words D00100 and D00101.

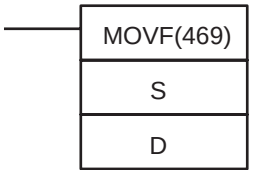


3-15-27 MOVE FLOATING-POINT (SINGLE): MOVF(469)

Purpose

Transfers the specified 32-bit floating-point number to the destination words. This instruction is supported by CJ1-H-R CPU Units only.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVF(469)
	Executed Once for Upward Differentiation	@MOVF(469)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

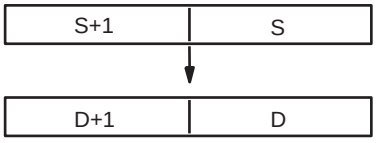
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	

Area	S	R
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the source data is 0. OFF in all other cases.
Negative Flag	N	ON if the source data is negative. OFF in all other cases.

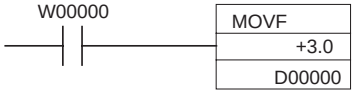
Precautions

When MOVF(469) is executed, the Error Flag is turned OFF.

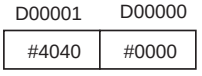
If the source data in S+1 and S is 0, the Equals Flag is turned ON. If the source data is non-zero, the Equals Flag is turned OFF.

If the source data in S+1 and S is negative, the Negative Flag is turned ON.

Operation Example



When input condition W00000 is ON, the content of D00000 and D00001 (+3.0) is stored in floating-point format (IEEE754 format).



Equivalent to +3.0 in floating-point format

3-16 Double-precision Floating-point Instructions (CS1-H, CJ1-H, CJ1M, or CS1D Only)

The Double-precision Floating-point Instructions convert data and perform floating-point arithmetic operations on double-precision floating-point data. The CS1-H/CJ1-H CPU Units support the following 20 instructions.

Instruction	Mnemonic	Function code	Page
DOUBLE FLOATING TO 16-BIT	FIXD	841	657
DOUBLE FLOATING TO 32-BIT	FIXLD	842	658
16-BIT TO DOUBLE FLOATING	DBL	843	660
32-BIT TO DOUBLE FLOATING	DBLL	844	661
DOUBLE FLOATING-POINT ADD	+D	845	663
DOUBLE FLOATING-POINT SUBTRACT	–D	846	665
DOUBLE FLOATING-POINT MULTIPLY	∗D	847	667
DOUBLE FLOATING-POINT DIVIDE	/D	848	669
DOUBLE DEGREES TO RADIANS	RADD	849	671
DOUBLE RADIANS TO DEGREES	DEGD	850	673
DOUBLE SINE	SIND	851	674
DOUBLE COSINE	COSD	852	676
DOUBLE TANGENT	TAND	853	678
DOUBLE ARC SINE	ASIND	854	680
DOUBLE ARC COSINE	ACOSD	855	682
DOUBLE ARC TANGENT	ATAND	856	684
DOUBLE SQUARE ROOT	SQRTD	857	686
DOUBLE EXPONENT	EXPD	858	688
DOUBLE LOGARITHM	LOGD	859	690
DOUBLE EXPONENTIAL POWER	PWRD	860	692
Double-precision Floating-point Symbol Comparison Instructions	LD, AND, OR + =D, <>D, <D, <=D, >D, or >=D	335 to 340	694

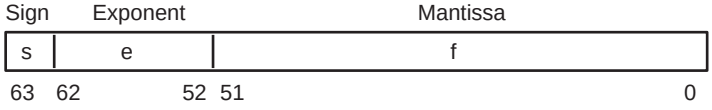
Data Format

Floating-point data expresses real numbers using a sign, exponent, and mantissa. When data is expressed in floating-point format, the following formula applies.

Real number = $(-1)^{s}\ 2^{e-1,023}\ (1.f)$

s: Sign
e: Exponent
f: Mantissa

The floating-point data format conforms to the IEEE754 standards. Data is expressed in 32 bits, as follows:



Data	No. of bits	Contents
s: sign	1	0: positive; 1: negative
e: exponent	11	The exponent (e) value ranges from 0 to 2,047. The actual exponent is the value remaining after 1,023 is subtracted from e, resulting in a range of –1,023 to 1,024. “e=0” and “e=2,047” express special numbers.
f: mantissa	52	The mantissa portion of binary floating-point data fits the format 2.0 > 1.f ≥1.0.

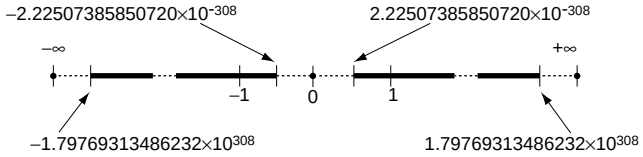
Number of Digits

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

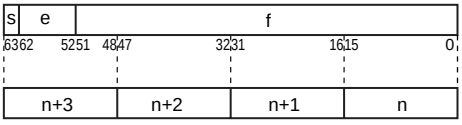
The formats for NaN, ±∞, and 0 are as follows:

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.

Normalized Numbers

Normalized numbers express real numbers. The sign bit will be 0 for a positive number and 1 for a negative number.

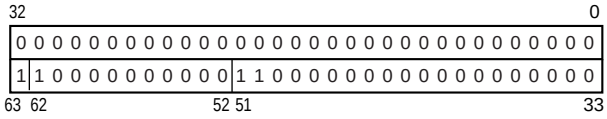
The exponent (e) will be expressed from 1 to 2,046, and the real exponent will be 1,023 less, i.e., –1,022 to 1,023.

The mantissa (f) will be expressed from 0 to $(2^{52} – 1)$, and it is assumed that, in the real mantissa, bit 2^{52} is 1 and the decimal point follows immediately after it.

Normalized numbers are expressed as follows:

$(–1)^{(\text{sign s})} \times 2^{(\text{exponent e})–1,023} \times (1 + \text{mantissa} \times 2^{–52})$

Example



Sign: –
Exponent: 1,024 – 1,023 = 1
Mantissa: $1 + (2^{51} + 2^{50}) \times 2^{–52} = 1 + (2^{–1} + 2^{–2}) = 1 + (0.75) = 1.75$
Value: $–1.75 \times 2^{1} = –3.5$

Non-normalized numbers

Non-normalized numbers express real numbers with very small absolute values. The sign bit will be 0 for a positive number and 1 for a negative number.

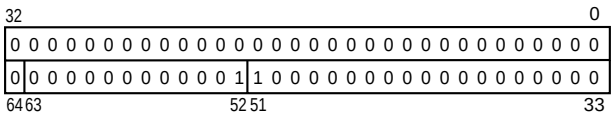
The exponent (e) will be 0, and the real exponent will be –1,022.

The mantissa (f) will be expressed from 1 to $(2^{52} – 1)$, and it is assumed that, in the real mantissa, bit 2^{52} is 0 and the decimal point follows immediately after it.

Non-normalized numbers are expressed as follows:

$(–1)^{(\text{sign s})} \times 2^{–1,022} \times (\text{mantissa} \times 2^{–52})$

Example



Sign: –
Exponent: –1,022
Mantissa: $0 + (2^{51} + 2^{50}) \times 2^{–52} = 0 + (2^{–1} + 2^{–2}) = 0 + (0.75) = 0.75$
Value: $–0.75 \times 2^{–1,022} = 1.668805 \times 10^{–308}$

Zero

Values of +0.0 and –0.0 can be expressed by setting the sign to 0 for positive or 1 for negative. The exponent and mantissa will both be 0. Both +0.0 and –0.0 are equivalent to 0.0. Refer to *Floating-point Arithmetic Results*, below, for differences produced by the sign of 0.0.

Infinity

Values of +∞ and –∞ can be expressed by setting the sign to 0 for positive or 1 for negative. The exponent will be 2,047 ($2^{11} - 1$) and the mantissa will be 0.

NaN

NaN (not a number) is produced when the result of calculations, such as 0.0/0.0, ∞/∞, or ∞–∞, does not correspond to a number or infinity. The exponent will be 255 ($2^8 - 1$) and the mantissa will be not 0.

Note There are no specifications for the sign of NaN or the value of the mantissa field (other than to be not 0).

Floating-point Arithmetic Results

Rounding Results

The following methods will be used to round results when the number of digits in the accurate result of floating-point arithmetic exceeds the significant digits of internal processing expressions.

If the result is close to one of two internal floating-point expressions, the closer expression will be used. If the result is midway between two internal floating-point expressions, the result will be rounded so that the last digit of the mantissa is 0.

Overflows, Underflows, and Illegal Calculations

Overflows will be output as either positive or negative infinity, depending on the sign of the result. Underflows will be output as either positive or negative zero, depending on the sign of the result.

Illegal calculations will result in NaN. Illegal calculations include adding infinity to a number with the opposite sign, subtracting infinity from a number with the opposite sign, multiplying zero and infinity, dividing zero by zero, or dividing infinity by infinity.

The value of the result may not be correct if an overflow occurs when converting a floating-point number to an integer.

Precautions in Handling Special Values

The following precautions apply to handling zero, infinity, and NaN.

- The sum of positive zero and negative zero is positive zero.
- The difference between zeros of the same sign is positive zero.
- If any operand is a NaN, the results will be a NaN.
- Positive zero and negative zero are treated as equivalent in comparisons.
- Comparison or equivalency tests on one or more NaN will always be true for != and always be false for all other instructions.

Double-precision Floating-point Calculation Results

When the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞. If the result is positive, it will be output as +∞; if negative, then –∞.

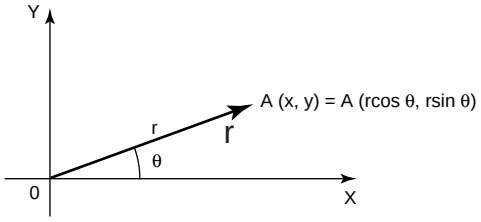
The Equals Flag will only turn ON when both the exponent (e) and the mantissa (f) are zero after a calculation. A calculation result will also be output as zero when the absolute value of the result is less than the minimum value that can be expressed for floating-point data. In that case the Underflow Flag will turn ON.

Comparing Single-precision and Double-precision Calculations

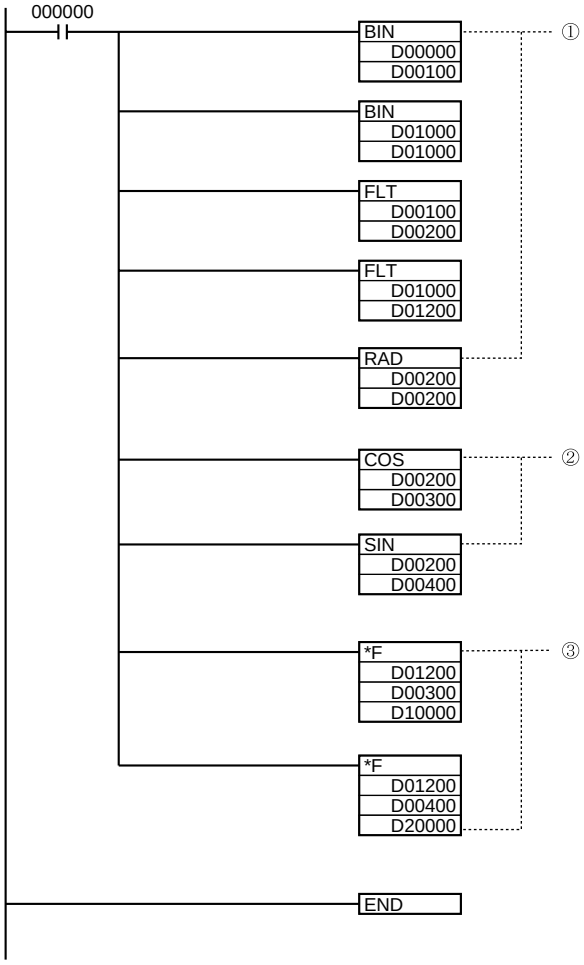
This example shows the differences in between single-precision and double-precision calculations when the following vector expressed in polar coordinates is converted to rectangular coordinates A (x,y).

$$r = re^{j\left(\frac{\pi}{360}\right)\theta}$$

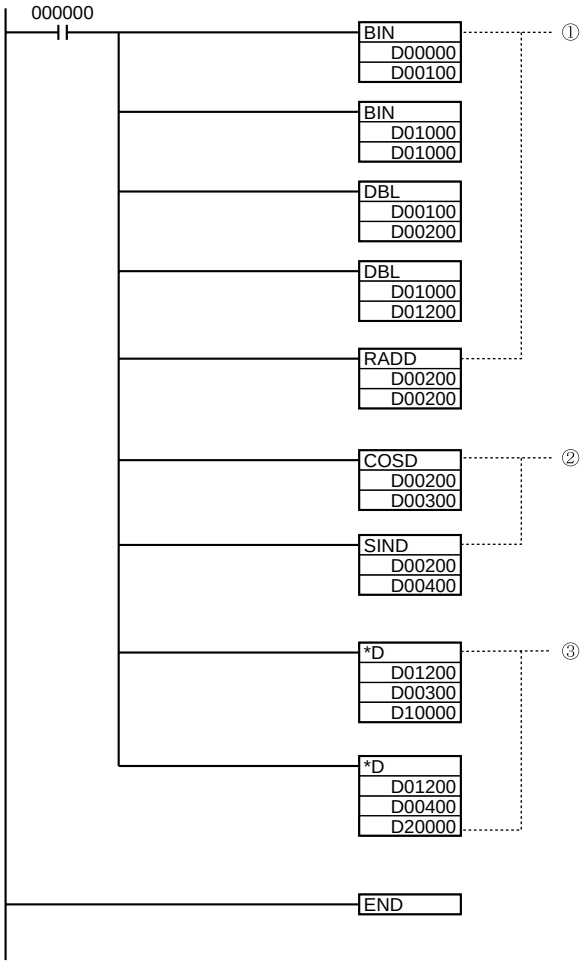
In this example, the 4-digit BCD angle (θ, in degrees) is read from D00000 and the 4-digit BCD distance (r) is read from D01000.



- Ladder Program for the Single-precision Calculation



- Ladder Program for the Double-precision Calculation



1. This program section converts the BCD data to single-precision floating-point data (32 bits, IEEE754-format).
   a) The BIN(023) instructions convert the BCD data to binary and the FLT(452) instructions convert the binary data to single-precision floating-point data.
   b) The floating-point data for the angle θ is output to D00200 and D00201.
   c) RAD(458) converts the angle data in D00200 and D00201 to radians.
   d) The floating-point data for the radius r is output to D01200 and D01201.
2. This program section calculates the sin θ and the cos θ as single-precision floating-point values.
   a) The value for cos θ is output to D00300 and D00301.
   b) The value for sin θ is output to D00400 and D00401.
3. This program section calculates x (r × cos θ) and y (r × sin θ).
   a) The value for x (r × cos θ) is output to D10000 and D10001.
   b) The value for y (r × sin θ) is output to D20000 and D20001.

Coordinate	Floating-point number	Real number
x	4116 59CF	3.4202015399933
y	405A E495	9.3969259262085

1. This program section converts the BCD data to double-precision floating-point data (64 bits, IEEE754-format).
   a) The BIN(023) instructions convert the BCD data to binary and the DBL(843) instructions convert the binary data to double-precision floating-point data.
   b) The floating-point data for the angle θ is output to words D00200 to D00203.
   c) RADD(849) converts the angle data in words D00200 to D00203 to radians.
   d) The floating-point data for the radius r is output to words D01200 to D01203.
2. This program section calculates the sin θ and the cos θ as double-precision floating-point values.
   a) The value for cos θ is output to words D00300 to D00303.
   b) The value for sin θ is output to words D00400 and D00403.
3. This program section calculates x (r × cos θ) and y (r × sin θ).
   a) The value for x (r × cos θ) is output to words D10000 to D10003.
   b) The value for y (r × sin θ) is output to D20000 and D20003.

Coordinate	Floating-point number	Real number
x	4022 CB39 E973 5C32	3.4202014332567
y	400B 5C92 91AC 8EEB	9.3969262078591

Comparison of the Calculation Results

When the real-number results are compared, it is clear that the double-precision calculation yields a more accurate result.

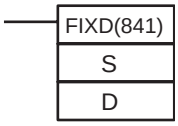
3-16-1 DOUBLE FLOATING TO 16-BIT: FIXD(841)

Purpose

Converts a double-precision (64-bit) floating-point value to 16-bit signed binary data and places the result in the specified result word.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FIXD(841)
	Executed Once for Upward Differentiation	@FIXD(841)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

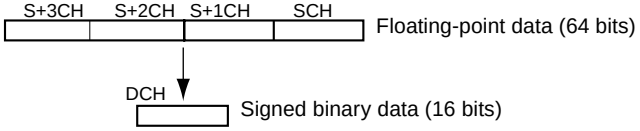
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	CIO 0000 to CIO 6143
Work Area	W000 to W508	W000 to W511
Holding Bit Area	H000 to H508	H000 to H511
Auxiliary Bit Area	A000 to A956	A448 to A959
Timer Area	T0000 to T4092	T0000 to T4095
Counter Area	C0000 to C4092	C0000 to C4095
DM Area	D00000 to D32764	D00000 to D32767
EM Area without bank	E00000 to E32764	E00000 to E32767
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

FIXD(841) converts the integer portion of the double-precision (64-bit) floating-point number in words S to S+3 (IEEE754-format) to 16-bit signed binary data and places the result in D.



Only the integer portion of the floating-point data is converted, and the fraction portion is truncated. The integer portion of the floating-point data must be within the range of –32,768 to 32,767.

Example conversions:
A floating-point value of 3.5 is converted to 3.
A floating-point value of –3.5 is converted to –3.

Flags

Name	Label	Operation
Error Flag	ER	ON if the source data (S to S+3) is not a number (NaN). ON if the integer portion of the source data (S to S+3) is not within the range of –32,768 to 32,767. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of the result is ON. OFF in all other cases.

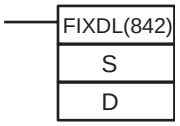
3-16-2 DOUBLE FLOATING TO 32-BIT: FIXLD(842)

Purpose

Converts a double-precision (64-bit) floating-point value to 32-bit signed binary data and places the result in the specified result words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FIXLD(842)
	Executed Once for Upward Differentiation	@FIXLD(842)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

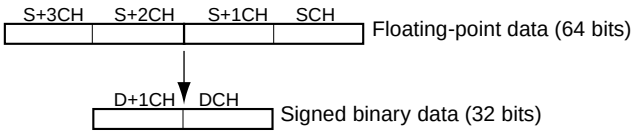
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	CIO 0000 to CIO 6142
Work Area	W000 to W508	W000 to W510

Area	S	D
Holding Bit Area	H000 to H508	H000 to H510
Auxiliary Bit Area	A000 to A956	A448 to A958
Timer Area	T0000 to T4092	T0000 to T4094
Counter Area	C0000 to C4092	C0000 to C4094
DM Area	D00000 to D32764	D00000 to D32766
EM Area without bank	E00000 to E32764	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

FIXLD(842) converts the integer portion of the double-precision (64-bit) floating-point number in words S to S+3 (IEEE754-format) to 32-bit signed binary data and places the result in D+1 and D.



Only the integer portion of the floating-point data is converted, and the fraction portion is truncated. (The integer portion of the floating-point data must be within the range of –2,147,483,648 to 2,147,483,647.)

Example conversions:
A floating-point value of 2,147,483,640.5 is converted to 2,147,483,640.
A floating-point value of –2,147,483,640.5 is converted to –2,147,483,640.

Flags

Name	Label	Operation
Error Flag	ER	ON if the data in words S to S+3 is not a number (NaN). ON if the integer portion of words S to S+3 is not within the range of –2,147,483,648 to 2,147,483,647. OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of D+1 is ON after execution. OFF in all other cases.

Precautions

The content of words S to S+3 must be floating-point data and the integer portion must be in the range of –2,147,483,648 to 2,147,483,647.

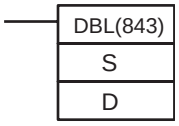
3-16-3 16-BIT TO DOUBLE FLOATING: DBL(843)

Purpose

Converts a 16-bit signed binary value to double-precision (64-bit) floating-point data and places the result in the specified destination words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: Source word
D: First destination word

Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	DBL(843)
	Executed Once for Upward Differentiation	@DBL(843)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

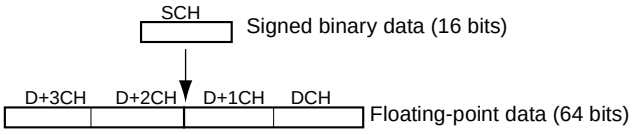
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6140
Work Area	W000 to W511	W000 to W508
Holding Bit Area	H000 to H511	H000 to H508
Auxiliary Bit Area	A000 to A959	A448 to A956
Timer Area	T0000 to T4095	T0000 to T4092
Counter Area	C0000 to C4095	C0000 to C4092
DM Area	D00000 to D32767	D00000 to D32764
EM Area without bank	E00000 to E32767	E00000 to E32764
EM Area with bank	En_00000 to En_32767 (n= 0 to C)	En_00000 to En_32764 (n= 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	---

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

Description

DBL(843) converts the 16-bit signed binary value in S to double-precision (64-bit) floating-point data (IEEE754-format) and places the result in words D to D+3. A single 0 is added after the decimal point in the floating-point result.



Only values within the range of –32,768 to 32,767 can be specified for S. To convert signed binary data outside of that range, use DBLL(844).

Example conversions:
A signed binary value of 3 is converted to 3.0.
A signed binary value of –3 is converted to –3.0.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The content of S must contain signed binary data with a (decimal) value in the range of –32,768 to 32,767.

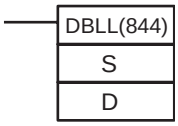
3-16-4 32-BIT TO DOUBLE FLOATING: DBLL(844)

Purpose

Converts a 32-bit signed binary value to double-precision (64-bit) floating-point data and places the result in the specified destination words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	DBLL(844)
	Executed Once for Upward Differentiation	@DBLL(844)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

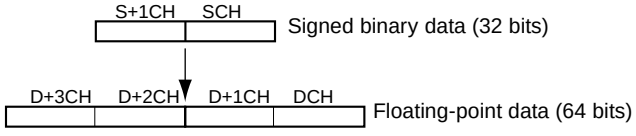
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6140
Work Area	W000 to W510	W000 to W508
Holding Bit Area	H000 to H510	H000 to H508
Auxiliary Bit Area	A000 to A958	A448 to A956
Timer Area	T0000 to T4094	T0000 to T4092
Counter Area	C0000 to C4094	C0000 to C4092
DM Area	D00000 to D32766	D00000 to D32764
EM Area without bank	E00000 to E32766	E00000 to E32764
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32764 (n= 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

DBLL(844) converts the 32-bit signed binary value in S+1 and S to double-precision (64-bit) floating-point data (IEEE754-format) and places the result in words D to D+3. A single 0 is added after the decimal point in the floating-point result.



Signed binary data within the range of –2,147,483,648 to 2,147,483,647 can be specified for S+1 and S. The floating point value has 24 significant binary digits (bits). The result will not be exact if a number greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted by DBLL(844).

Example Conversions:

A signed binary value of 16,777,215 is converted to 16,777,215.0.
A signed binary value of –16,777,215 is converted to –15,777,215.0.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The result will not be exact if a number with an absolute value greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted.

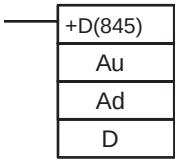
3-16-5 DOUBLE FLOATING-POINT ADD: +D(845)

Purpose

Adds two double-precision (64-bit) floating-point numbers and places the result in the specified destination words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Au: First augend word
Ad: First addend word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	+D(845)
	Executed Once for Upward Differentiation	@+D(845)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

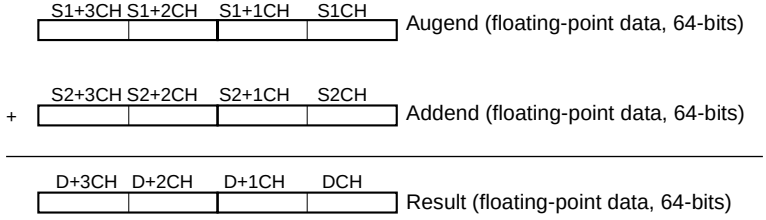
Operand Specifications

Area	Au	Ad	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W508		
Holding Bit Area	H000 to H508		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D00000 to D32764		
EM Area without bank	E00000 to E32764		
EM Area with bank	En_00000 to En_32764 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		

Area	Au	Ad	D
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+D(845) adds the double-precision (64-bit) floating-point number in words Ad to Ad+3 the double-precision (64-bit) floating-point number in words Au to Au+3 and places the result in words D to D+3. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of augend and addend data will produce the results shown in the following table.

	Augend				
Addend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	
Numeral	Numeral	See note 1.	+∞	–∞	
+∞	+∞	+∞	+∞	See note 2.	
–∞	–∞	–∞	See note 2.	–∞	
NaN					See note 2.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the augend or addend data is not recognized as floating-point data. ON if the augend or addend data is not a number (NaN). ON if +∞ is to –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The augend (Au to Au+3) and Addend (Ad to Ad+3) data must be in IEEE754 floating-point data format.

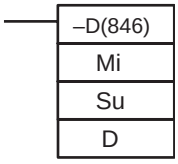
3-16-6 DOUBLE FLOATING-POINT SUBTRACT: –D(846)

Purpose

Subtracts one double-precision (64-bit) floating-point number from another and places the result in the specified destination words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Mi: First Minuend word
Su: First Subtrahend word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	–D(846)
	Executed Once for Upward Differentiation	@–D(846)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

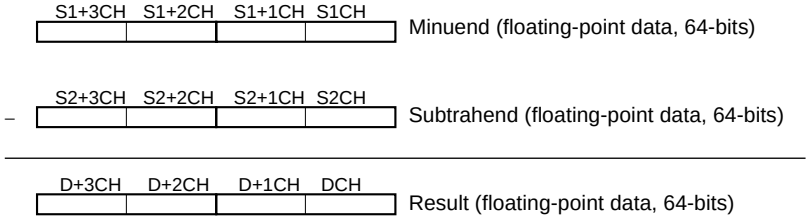
Operand Specifications

Area	Mi	Su	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W508		
Holding Bit Area	H000 to H508		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D00000 to D32764		
EM Area without bank	E00000 to E32764		

Area	Mi	Su	D
EM Area with bank	En_00000 to En_32764 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

–D(846) subtracts the double-precision (64-bit) floating-point number in words Su to Su+3 from the double-precision (64-bit) floating-point number in Mi to Mi+3 and places the result in words D to D+3. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of minuend and subtrahend data will produce the results shown in the following table.

	Minuend				
Subtrahend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	
Numeral	Numeral	See note 1.	+∞	–∞	
+∞	–∞	–∞	See note 2.	–∞	
–∞	+∞	+∞	+∞	See note 2.	
NaN					See note 2.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the minuend or subtrahend data is not recognized as floating-point data. ON if the minuend or subtrahend is not a number (NaN). ON if +∞ is subtracted from +∞. ON if –∞ is subtracted from –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The Minuend (Mi to Mi+3) and Subtrahend (Su to Su+3) data must be in IEEE754 floating-point data format.

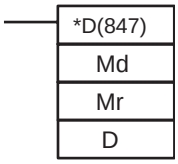
3-16-7 DOUBLE FLOATING-POINT MULTIPLY: ✳D(847)

Purpose

Multiplies two double-precision (64-bit) floating-point numbers and places the result in the specified result words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Md: First Multiplicand word
Mr: First Multiplier word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	*D(847)
	Executed Once for Upward Differentiation	@*D(847)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

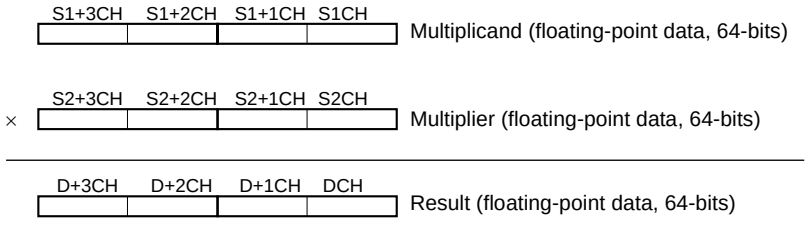
Operand Specifications

Area	Md	Mr	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W508		
Holding Bit Area	H000 to H508		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D00000 to D32764		

Area	Md	Mr	D
EM Area without bank	E00000 to E32764		
EM Area with bank	En_00000 to En_32764 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

∗D(847) multiplies the double-precision (64-bit) floating-point number in words Md to Md+3 by the double-precision (64-bit) floating-point number in words Mr to Mr+3 and places the result in words D to D+3. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of multiplicand and multiplier data will produce the results shown in the following table.

	Multiplicand				
Multiplier	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	0	See note 2.	See note 2.	
Numeral	0	See note 1.	+/–∞	+/–∞	
+∞	See note 2.	+/–∞	+∞	–∞	
–∞	See note 2	+/–∞	–∞	+∞	
NaN					See note 2.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the multiplicand or multiplier data is not recognized as floating-point data. ON if the multiplicand or multiplier is not a number (NaN). ON if +∞ and 0 are multiplied. ON if –∞ and 0 are multiplied. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The Multiplicand (Md to Md+3) and Multiplier (Mr to Mr+3) data must be in IEEE754 floating-point data format.

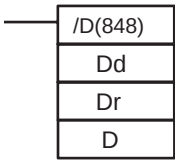
3-16-8 DOUBLE FLOATING-POINT DIVIDE: /D(848)

Purpose

Divides one double-precision (64-bit) floating-point number by another and places the result in the specified destination words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Dd: First Dividend word
Dr: First Divisor word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	/D(848)
	Executed Once for Upward Differentiation	@/D(848)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

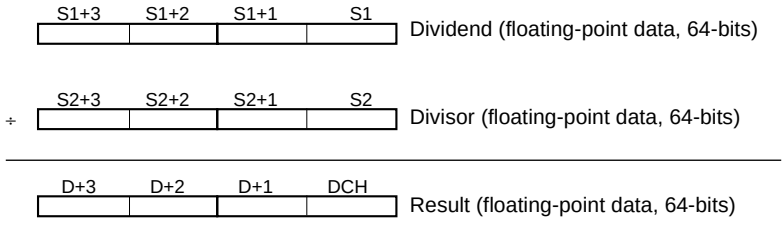
Operand Specifications

Area	Dd	Dr	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W508		
Holding Bit Area	H000 to H508		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D00000 to D32764		

Area	Dd	Dr	D
EM Area without bank	E00000 to E32764		
EM Area with bank	En_00000 to En_32764 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

/D(848) divides the double-precision (64-bit) floating-point number in words Dd to Dd+3 by the double-precision (64-bit) floating-point number in words Dr to Dr+3 and places the result in words D to D+3. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of dividend and divisor data will produce the results shown in the following table.

	Dividend				
Divisor	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	See note 3.	+/–∞	+∞	–∞	
Numeral	0	See note 1.	+/–∞	+/–∞	
+∞	0	See note 2.	See note 3.	See note 3.	
–∞	0	See note 2.	See note 3.	See note 3.	
NaN	See note 3.				

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The results will be zero for underflows.
3. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the dividend or divisor data is not recognized as floating-point data. ON if the dividend or divisor is not a number (NaN). ON if the dividend and divisor are both 0. ON if the dividend and divisor are both +∞ or –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The Dividend (Dd to Dd+3) and Divisor (Dr to Dr+3) data must be in IEEE754 floating-point data format.

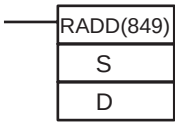
3-16-9 DOUBLE DEGREES TO RADIANS: RADD(849)

Purpose

Converts a double-precision (64-bit) floating-point number from degrees to radians and places the result in the specified result words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	RADD(849)
	Executed Once for Upward Differentiation	@RADD(849)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

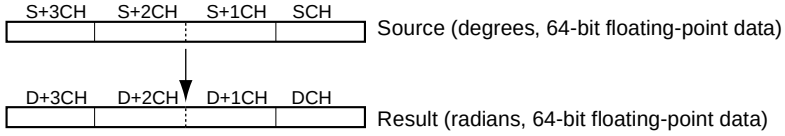
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	

Area	S	D
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

RADD(849) converts the double-precision (64-bit) floating-point number in words S to S+3 from degrees to radians and places the result in words D to D+3. (The floating point source data must be in IEEE754 format.)



Degrees are converted to radians by means of the following formula:

Degrees $\times \pi/180$ = radians

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as $\pm\infty$.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

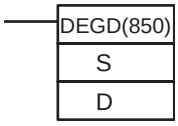
3-16-10 DOUBLE RADIANS TO DEGREES: DEGD(850)

Purpose

Converts a double-precision (64-bit) floating-point number from radians to degrees and places the result in the specified result words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	DEGD(850)
	Executed Once for Upward Differentiation	@DEGD(850)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

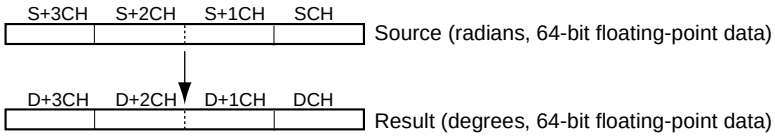
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	

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

Description

DEGD(850) converts the double-precision (64-bit) floating-point number in words S to S+3 from radians to degrees and places the result in words D to D+3. (The floating point source data must be in IEEE754 format.)



Radians are converted to degrees by means of the following formula:

Radians $\times$ 180/π = degrees

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as $\pm\infty$.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

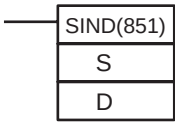
3-16-11 DOUBLE SINE: SIND(851)

Purpose

Calculates the sine of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	SIND(851)
	Executed Once for Upward Differentiation	@SIND(851)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

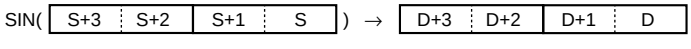
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

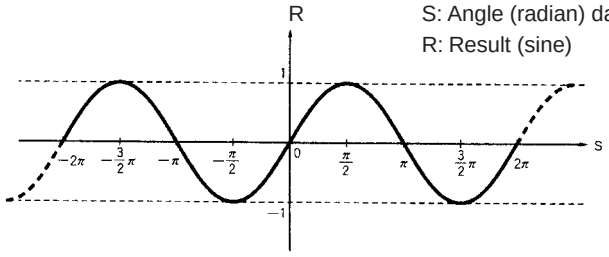
Description

SIND(851) calculates the sine of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in words S to S+3. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting between degrees and radians, see *3-16-9 DOUBLE DEGREES TO RADIANS: RADD(849)* or *3-16-10 DOUBLE RADIANS TO DEGREES: DEGD(850)*.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

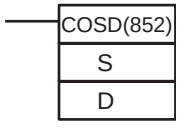
3-16-12 DOUBLE COSINE: COSD(852)

Purpose

Calculates the cosine of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	COSD(852)
	Executed Once for Upward Differentiation	@COSD(852)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

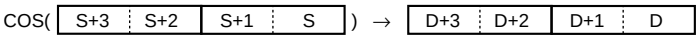
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

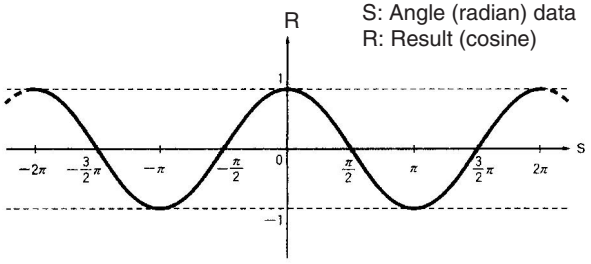
Description

COSD(852) calculates the cosine of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in words S to S+3. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting between degrees and radians, see *3-16-9 DOUBLE DEGREES TO RADIANS: RADD(849)* or *3-16-10 DOUBLE RADIANS TO DEGREES: DEGD(850)*.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

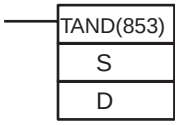
3-16-13 DOUBLE TANGENT: TAND(853)

Purpose

Calculates the tangent of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	TAND(853)
	Executed Once for Upward Differentiation	@TAND(853)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	

Area	S	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

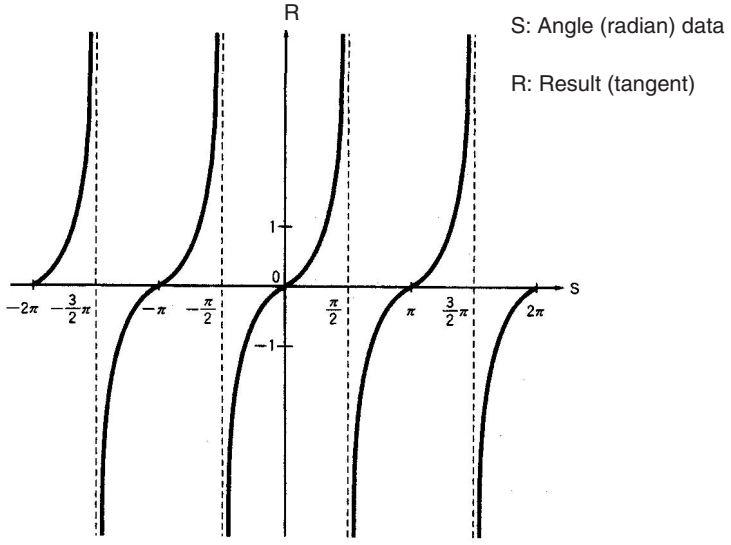
TAND(853) calculates the tangent of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)

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. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting between degrees and radians, see *3-16-9 DOUBLE DEGREES TO RADIANS: RADD(849)* or *3-16-10 DOUBLE RADIANS TO DEGREES: DEGD(850)*.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value.
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

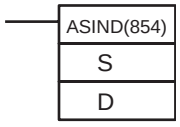
3-16-14 DOUBLE ARC SINE: ASIND(854)

Purpose

Calculates the arc sine of a double-precision (64-bit) floating-point number and places the result in the specified destination words. (The arc sine function is the inverse of the sine function; it returns the angle that produces a given sine value between –1 and 1.)

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASIND(854)
	Executed Once for Upward Differentiation	@ASIND(854)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

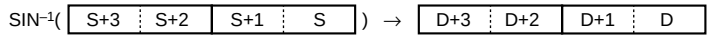
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	

Area	S	D
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

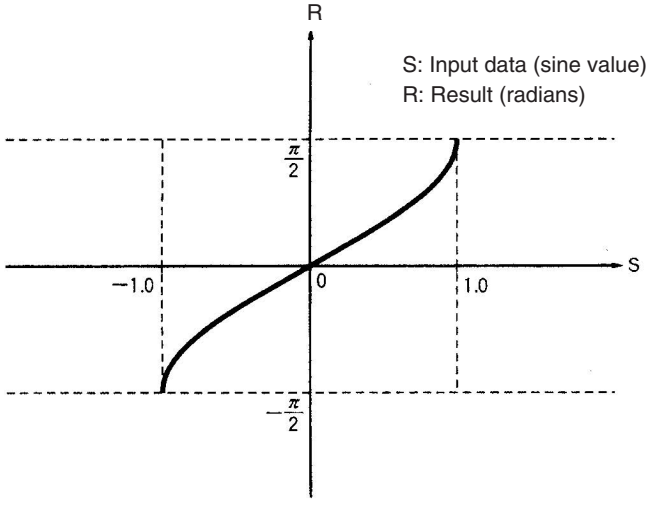
ASIND(854) computes the angle (in radians) for a sine value expressed as a double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



The source data must be between –1.0 and 1.0. If the absolute value of the source data exceeds 1.0, an error will occur and the instruction will not be executed.

The result is output to words D to D+3 as an angle (in radians) within the range of $-\pi/2$ to $\pi/2$.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

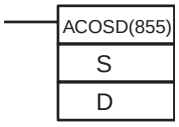
3-16-15 DOUBLE ARC COSINE: ACOSD(855)

Purpose

Calculates the arc cosine of a double-precision (64-bit) floating-point number and places the result in the specified result words. (The arc cosine function is the inverse of the cosine function; it returns the angle that produces a given cosine value between –1 and 1.)

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	ACOSD(855)
	Executed Once for Upward Differentiation	@ACOSD(855)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

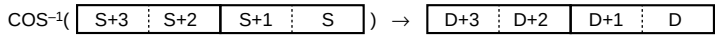
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	

Area	S	D
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

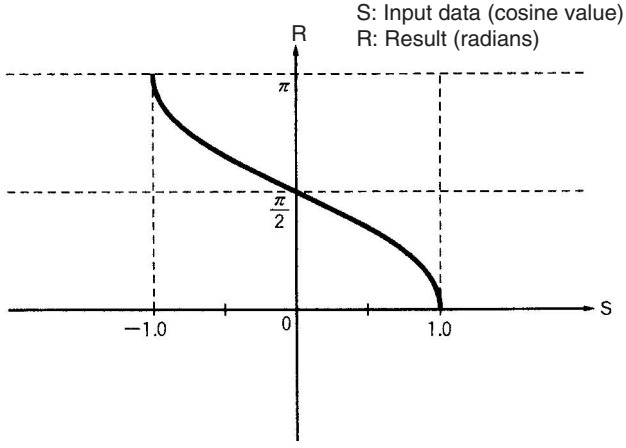
ACOSD(855) computes the angle (in radians) for a cosine value expressed as a double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



The source data must be between –1.0 and 1.0. If the absolute value of the source data exceeds 1.0, an error will occur and the instruction will not be executed.

The result is output to words D to D+3 as an angle (in radians) within the range of 0 to π.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	Unchanged

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

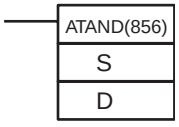
3-16-16 DOUBLE ARC TANGENT: ATAND(856)

Purpose

Calculates the arc tangent of a double-precision (64-bit) floating-point number and places the result in the specified result words. (The arc tangent function is the inverse of the tangent function; it returns the angle that produces a given tangent value.)

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	ATAND(856)
	Executed Once for Upward Differentiation	@ATAND(856)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

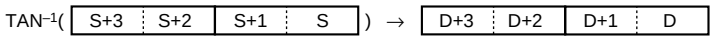
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	

Area	S	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

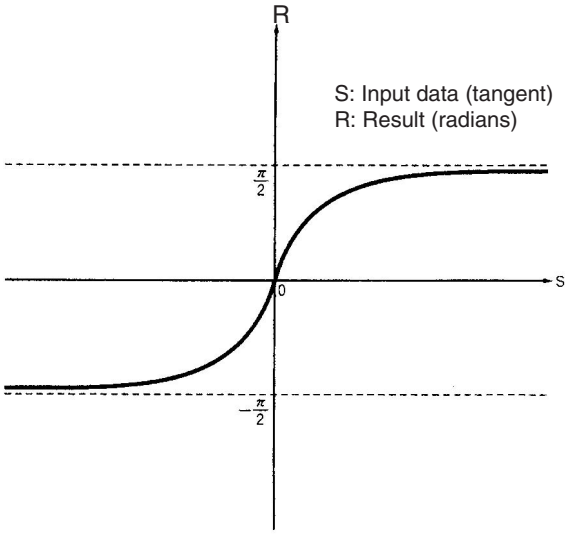
Description

ATAND(856) computes the angle (in radians) for a tangent value expressed as a double-precision (64-bit) floating-point number in words S to S+3 and places the result in D to D+3.
(The floating point source data must be in IEEE754 format.)



The result is output to words D to D+3 as an angle (in radians) within the range of $-\pi/2$ to $\pi/2$.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

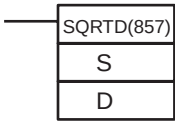
3-16-17 DOUBLE SQUARE ROOT: SQRTD(857)

Purpose

Calculates the square root of a double-precision (64-bit) floating-point number and places the result in the specified result words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	SQRTD(857)
	Executed Once for Upward Differentiation	@SQRTD(857)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

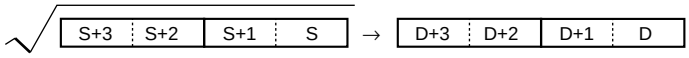
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	

Area	S	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

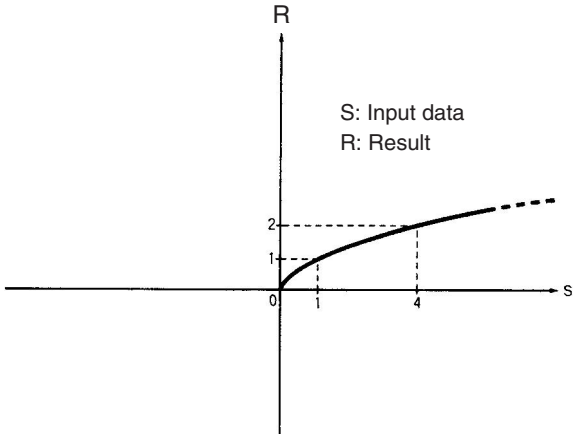
SQRTD(857) calculates the square root of the double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3. (The floating point source data must be in IEEE754 format.)



The source data must be positive; if it is negative, an error will occur and the instruction will not be executed.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is negative. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value.
Underflow Flag	UF	Unchanged
Negative Flag	N	Unchanged

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

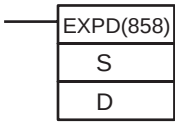
3-16-18 DOUBLE EXPONENT: EXPD(858)

Purpose

Calculates the natural (base e) exponential of a double-precision (64-bit) floating-point number and places the result in the specified result words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	EXPD(858)
	Executed Once for Upward Differentiation	@EXPD(858)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

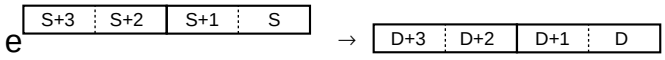
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	

Area	S	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

EXPD(858) calculates the natural (base e) exponential of the double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3. In other words, EXP(467) calculates e^x (x = source) and places the result in words D to D+3.

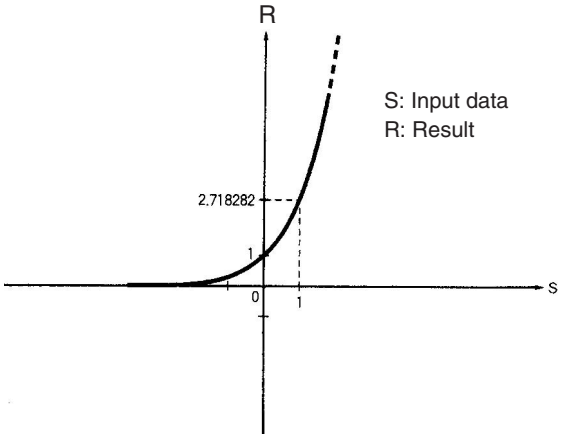


If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Note The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision (64-bit) floating-point value.
Negative Flag	N	Unchanged

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

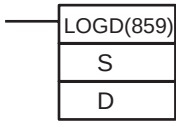
3-16-19 DOUBLE LOGARITHM: LOGD(859)

Purpose

Calculates the natural (base e) logarithm of a double-precision (64-bit) floating-point number and places the result in the specified destination words.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	LOGD(859)
	Executed Once for Upward Differentiation	@LOGD(859)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	

Area	S	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

LOGD(859) calculates the natural (base e) logarithm of the double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3.

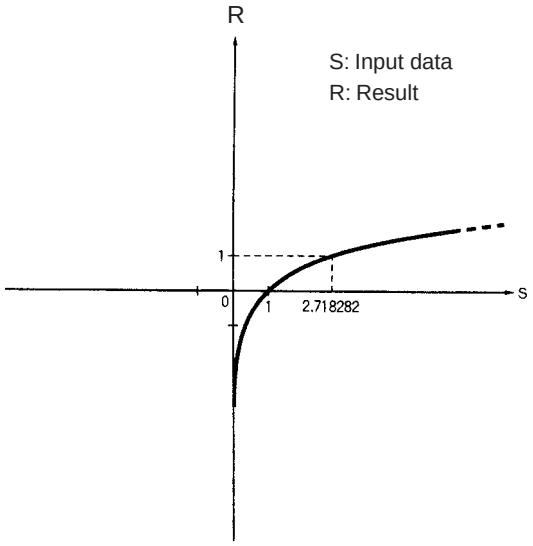
$\log_e$ [S+3 | S+2 | S+1 | S] → [D+3 | D+2 | D+1 | D]

The source data must be positive; if it is negative, an error will occur and the instruction will not be executed.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

Note The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is negative. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value.
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The source data in words S to S+3 must be in IEEE754 floating-point data format.

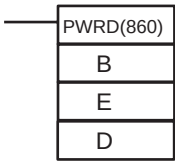
3-16-20 DOUBLE EXPONENTIAL POWER: PWRD(860)

Purpose

Raises a double-precision (64-bit) floating-point number to the power of another double-precision (64-bit) floating-point number.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



B: First base word
E: First exponent word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	PWRD(860)
	Executed Once for Upward Differentiation	@PWRD(860)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

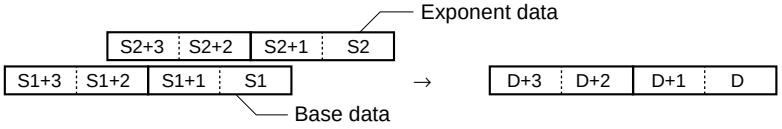
Operand Specifications

Area	B	E	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W508		
Holding Bit Area	H000 to H508		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T4092		
Counter Area	C0000 to C4092		
DM Area	D00000 to D32764		
EM Area without bank	E00000 to E32764		

Area	B	E	D
EM Area with bank	En_00000 to En_32764 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

PWRD(860) raises the double-precision (64-bit) floating-point number in words B to B+3 to the power of the double-precision (64-bit) floating-point number in words E to E+3. In other words, PWR(840) calculates X^Y (X = content of B to B+3; Y = content of E to E+3).



For example, when the base words (B to B+3) contain 3.1 and the exponent words (E to E+3) contain 3, the result is 3.1^3 or 29.791.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if the base data (B to B+3) or exponent data (E to E+3) is not recognized as floating-point data. ON if the base data (B to B+3) or exponent data (E to E+3) is not a number (NaN). ON if the base data (B to B+3) is 0 and the exponent data (E to E+3) is less than 0. (Division by 0) ON if the base data (B to B+3) is negative and the exponent data (E to E+3) is non-integer. (Root of a negative number) OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.

Name	Label	Operation
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The base data (B to B+3) and the exponent data (E to E+3) must be in IEEE754 floating-point data format.

3-16-21 Double-precision Floating-point Input Instructions

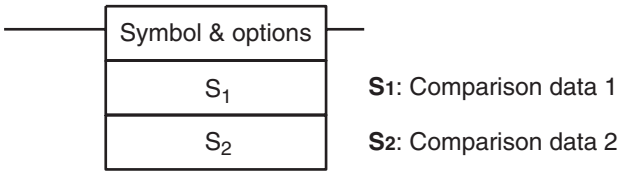
Purpose

These input comparison instructions compare two double-precision floating point values (64-bit IEEE754 format) and create an ON execution condition when the comparison condition is true.

These instructions are supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Note Refer to *3-7-1 Input Comparison Instructions (300 to 328)* for details on the signed and unsigned binary input comparison instructions and *3-15-24 Single-precision Floating-point Comparison Instructions* for details on single-precision floating-point input comparison instructions.

Ladder Symbol



Variations

Variations	Creates ON Each Cycle Comparison is True	Input comparison instruction
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W508	
Holding Bit Area	H000 to H508	
Auxiliary Bit Area	A000 to A956	
Timer Area	T0000 to T4092	
Counter Area	C0000 to C4092	
DM Area	D00000 to D32764	
EM Area without bank	E00000 to E32764	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	

Area	S_1	S_2
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

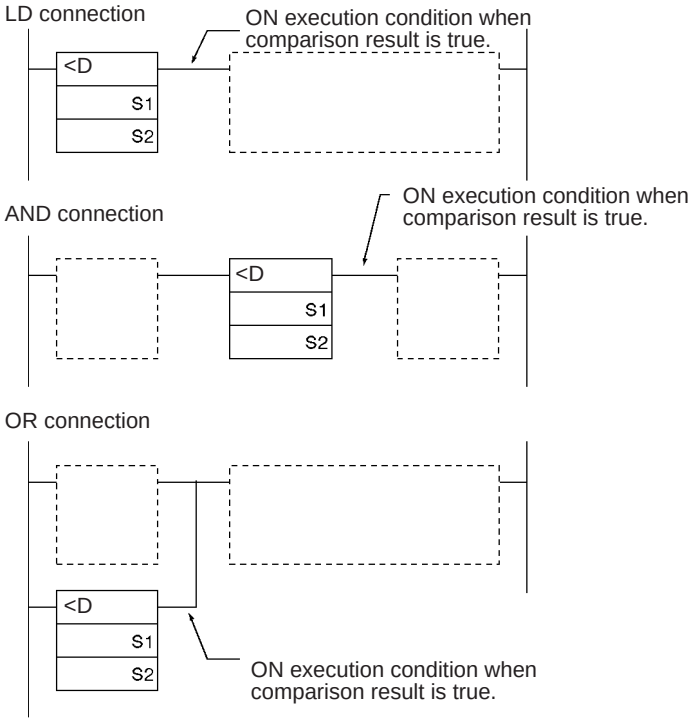
Description

The input comparison instruction compares the data specified in S_1 and S_2 as double-precision floating point values (64-bit IEEE754 data) and creates an ON execution condition when the comparison condition is true. When the data is stored in words, S_1 and S_2 specify the first of four words containing the 64-bit data. The 64-bit floating-point data cannot be input as constants.

Inputting the Instructions

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

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



Options

With the three input types and six symbols, there are 18 different possible combinations.

Symbol		Option (data format)
=	(Equal)	D: Double-precision floating-point data
< >	(Not equal)	
<	(Less than)	
<=	(Less than or equal)	
>	(Greater than)	
>=	(Greater than or equal)	

Summary of Input Comparison Instructions

The following table shows the function codes, mnemonics, names, and functions of the 18 single-precision floating-point input comparison instructions. (C1=S_1+3, S_1+2, S_1+1, S_1 and C2=S_2+3, S_2+2, S_2+1, S_2.)

Code	Mnemonic	Name	Function
335	LD=D	LOAD DOUBLE FLOATING EQUAL	True if C1 = C2
	AND=D	AND DOUBLE FLOATING EQUAL	
	OR=D	OR DOUBLE FLOATING EQUAL	
336	LD<>D	LOAD DOUBLE FLOATING NOT EQUAL	True if C1 ≠ C2
	AND<>D	AND DOUBLE FLOATING NOT EQUAL	
	OR<>D	OR DOUBLE FLOATING NOT EQUAL	
337	LD<D	LOAD DOUBLE FLOATING LESS THAN	True if C1 < C2
	AND<D	AND DOUBLE FLOATING LESS THAN	
	OR<D	OR DOUBLE FLOATING LESS THAN	
338	LD<=D	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	True if C1 ≤ C2
	AND<=D	AND DOUBLE FLOATING LESS THAN OR EQUAL	
	OR<=D	OR DOUBLE FLOATING LESS THAN OR EQUAL	
339	LD>D	LOAD DOUBLE FLOATING GREATER THAN	True if C1 > C2
	AND>D	AND DOUBLE FLOATING GREATER THAN	
	OR>D	OR DOUBLE FLOATING GREATER THAN	
340	LD>=D	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	True if C1 ≥ C2
	AND>=D	AND DOUBLE FLOATING GREATER THAN OR EQUAL	
	OR>=D	OR DOUBLE FLOATING GREATER THAN OR EQUAL	

Flags

In this table, C1 = content of S1 to S1+3 and C2 = content of S2 to S2+3.

Name	Label	Operation
Error Flag	ER	OFF
Greater Than Flag	>	ON if C1 > C2. OFF in all other cases.
Greater Than or Equal Flag	> =	ON if C1 ≥ C2. OFF in all other cases.
Equal Flag	=	ON if C1 = C2. OFF in all other cases.
Not Equal Flag	=	ON if C1 ≠ C2. OFF in all other cases.

Name	Label	Operation
Less Than Flag	<	ON if C1 < C2. OFF in all other cases.
Less Than or Equal Flag	< =	ON if C1 ≤ C2. OFF in all other cases.
Negative Flag	N	Unchanged

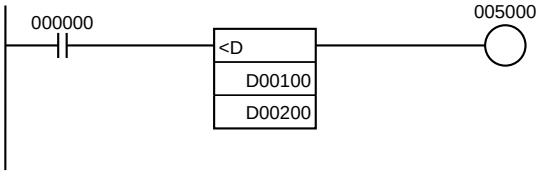
Precautions

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

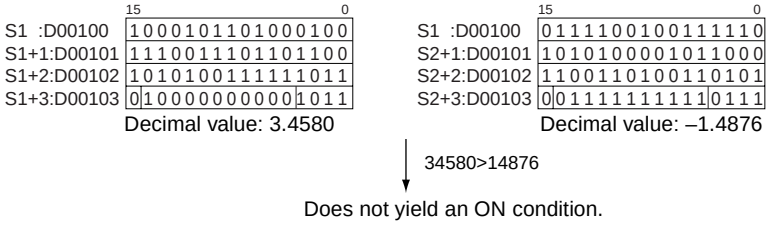
Example

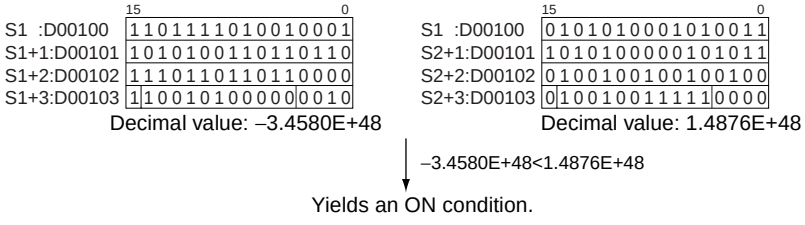
AND DOUBLE FLOATING LESS THAN: AND<D(331)

When CIO 000000 is ON in the following example, the floating point data in words D00100 to D00103 is compared to the floating point data in words D00200 to D00203. If the content of D00100 to D00103 is less than that of D00200 to D00203, execution proceeds to the next line and CIO 005000 is turned ON. If the content of D00100 to D00103 is not less than that of D00200 to D00203, execution does not proceed to the next instruction line.



DOUBLE FLOATING LESS THAN Comparison (<D)





3-17 Table Data Processing Instructions

This section describes instructions used to handle table data, stacks, and other ranges of data. The 5 instructions at the bottom of the table (marked with an asterisk) are supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Instruction	Mnemonic	Function code	Page
SET STACK	SSET	630	703
PUSH ONTO STACK	PUSH	632	706
FIRST IN FIRST OUT	FIFO	633	709
LAST IN FIRST OUT	LIFO	634	712
DIMENSION RECORD TABLE	DIM	631	715

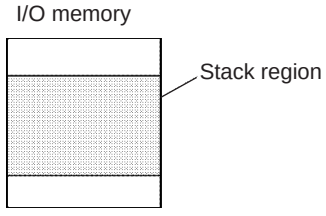
Instruction	Mnemonic	Function code	Page
SET RECORD LOCATION	SETR	635	718
GET RECORD NUMBER	GETR	636	720
DATA SEARCH	SRCH	181	722
SWAP BYTES	SWAP	637	725
FIND MAXIMUM	MAX	182	727
FIND MINIMUM	MIN	183	731
SUM	SUM	184	735
FRAME CHECKSUM	FCS	180	738
STACK NUMBER OUTPUT	SNUM	638	742
STACK DATA READ	SREAD	639	744
STACK DATA OVERWRITE	SWRIT	640	747
STACK DATA INSERT	SINS	641	750
STACK DATA DELETE	SDEL	642	753

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.

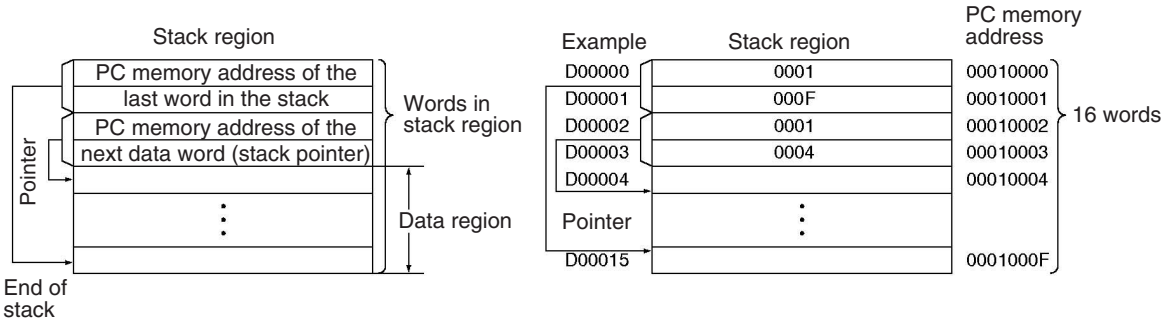
Group	Purpose	Instructions
Stack	Operate FIFO (first-in first-out) or LIFO (last-in first-out) data tables.	SSET(630), PUSH(632), FIFO(633), LIFO(634), SREAD(639), SWRIT(640), SINS(641), SDEL(642), and SNUM(638)
Record-table	Operate tables of data made up of records. (Record size is user-defined.)	DIM(631), 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.

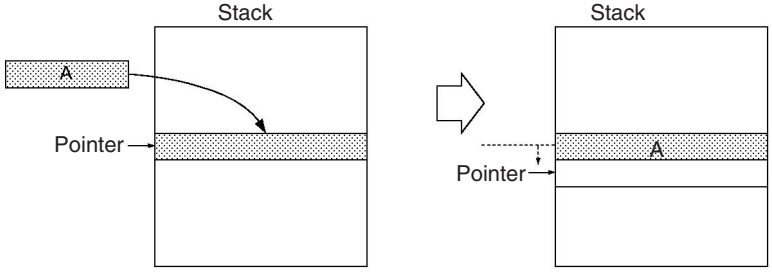


The following instructions define or act on stack regions. Basically, PUSH(632) stores data in the next available data word in the stack. FIFO(633) and LIFO(634) read data from the stack. FIFO(633) reads the first word that was stored, while LIFO(634) reads the last word that was stored.

The last five instructions are supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only. SNUM(638) counts the number of data elements (words) in the specified stack; for example, this instruction could be used to indicate the number of items on a conveyor. Use the SREAD(639), SWRIT(640), SINS(641), and SDEL(642) instructions to read, overwrite, insert, and delete data elements in a stack. For example, when items are being handled on a conveyor, these instructions can add, remove, or change a data element in the stack that corresponds to an item on the conveyor.

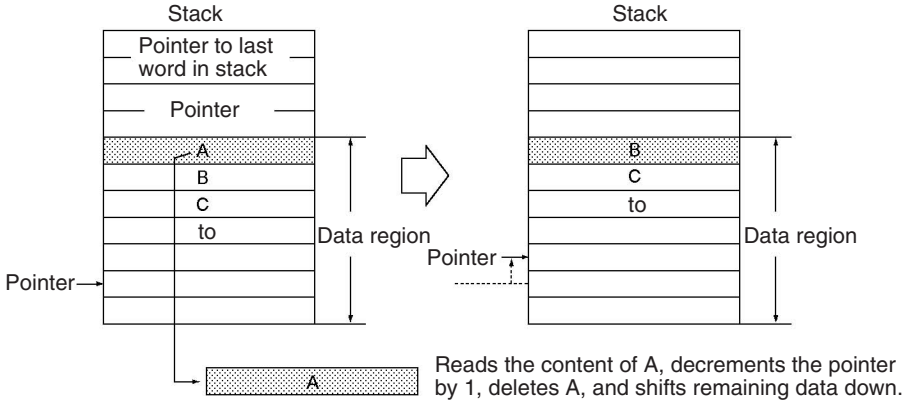
PUSH(632)

Stores data in the address indicated by the stack pointer and increments the pointer by one.



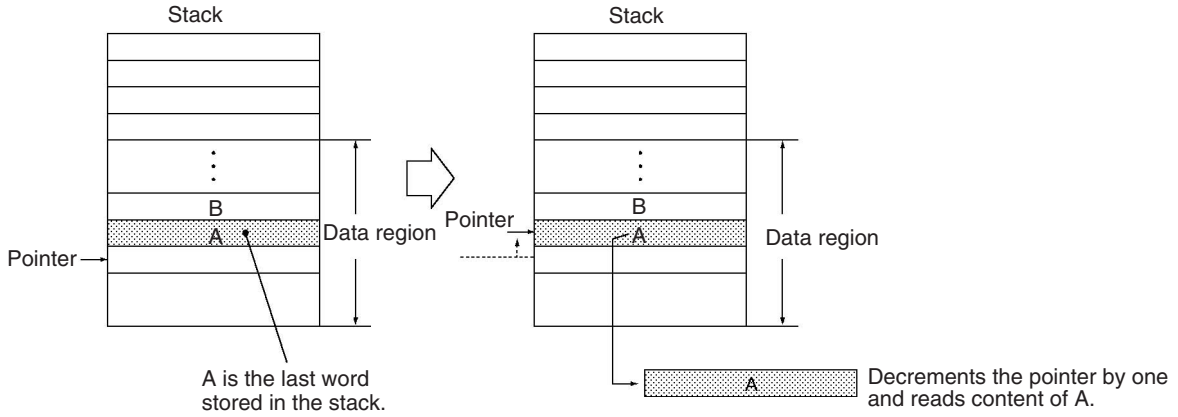
FIFO(633)

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.



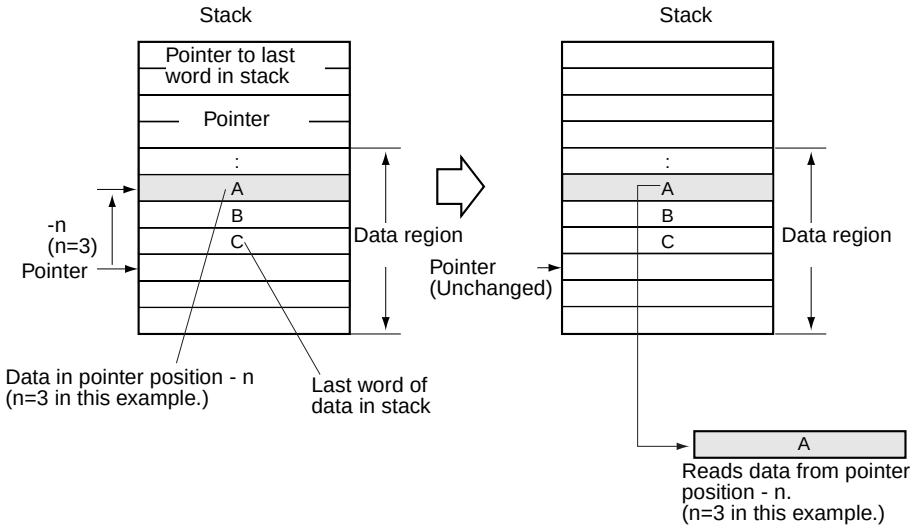
LIFO(634)

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.



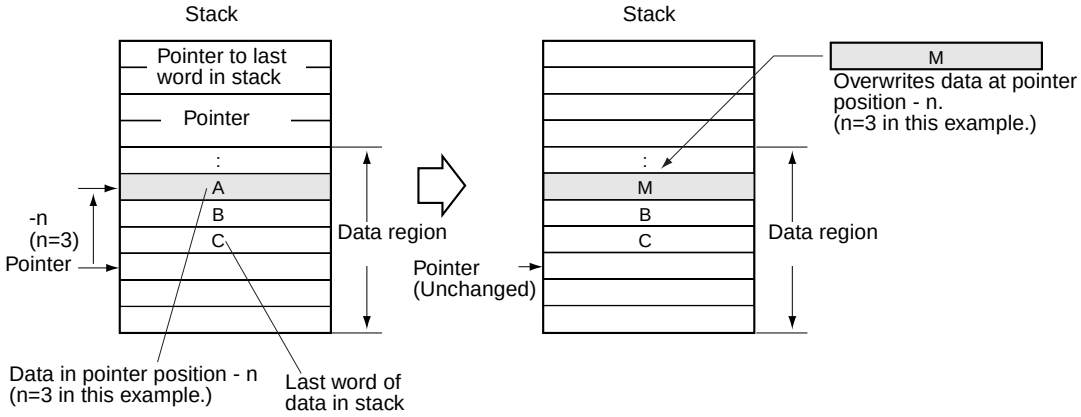
SREAD(639)

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).



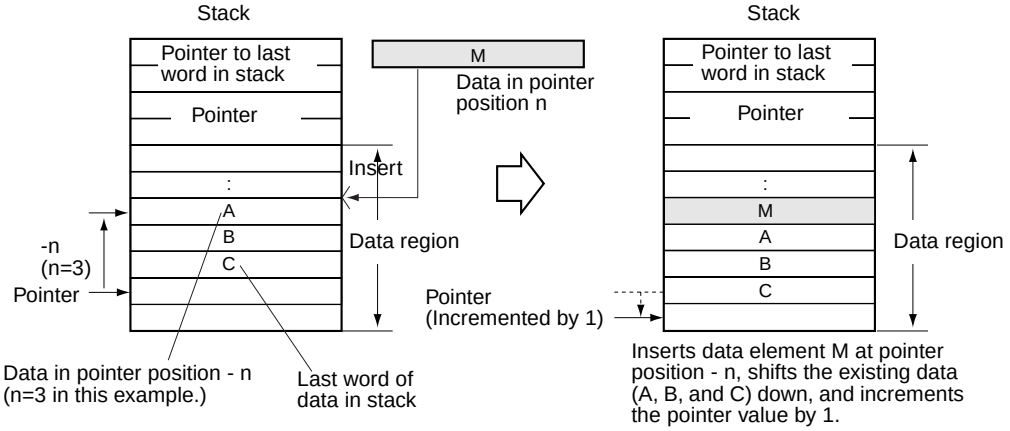
SWRIT(640)

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).



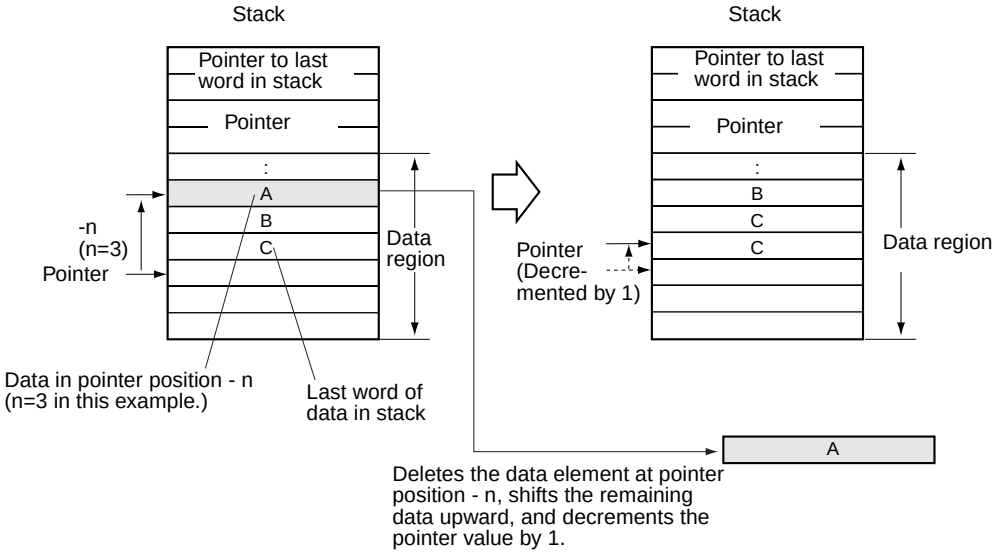
SINS(641)

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).



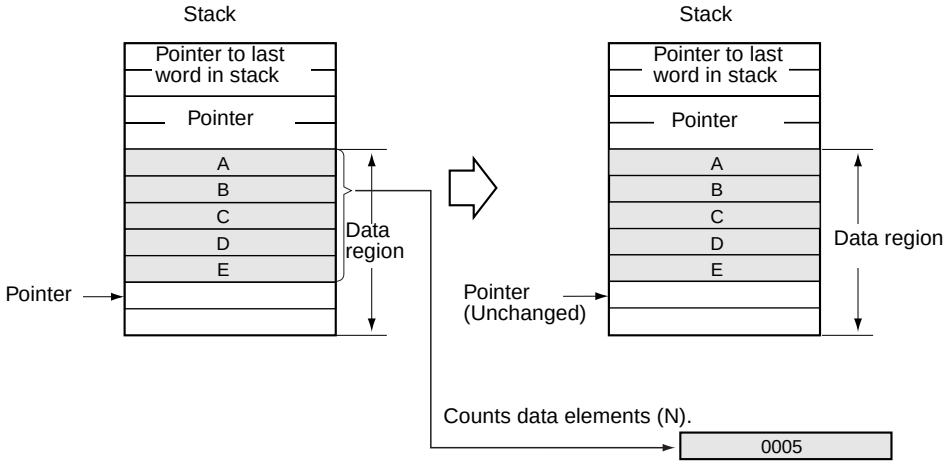
SDEL(642)

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).



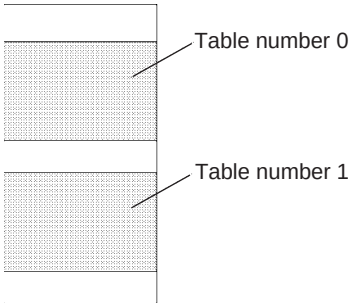
SNUM(638)

Counts the amount of stack data (number of words of data) from the stack pointer to the beginning of the data region.

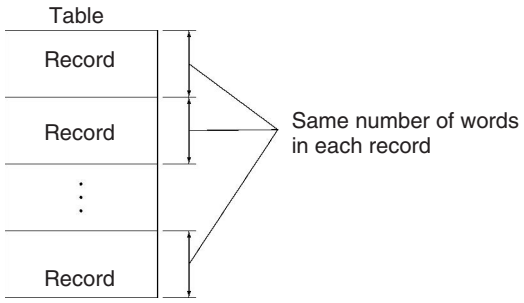


Record-table Instructions

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.



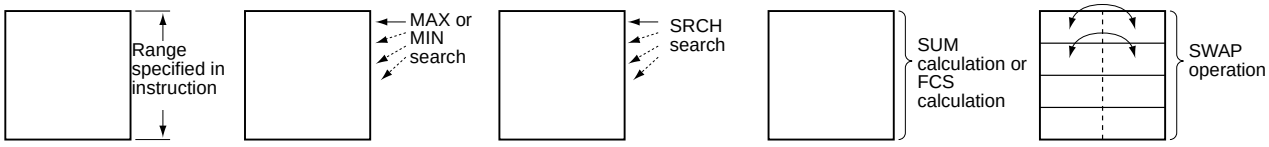
The following diagram shows the basic structure of a record table. Each record in a table has the same number of words.



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)).

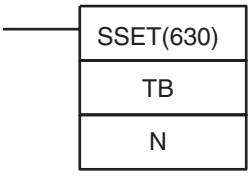


3-17-1 SET STACK: SSET(630)

Purpose

Defines a stack of the specified length beginning at the specified word.

Ladder Symbol



TB: First stack address

N: Number of words

Variations

Variations		
	Executed Each Cycle for ON Condition	SSET(630)
	Executed Once for Upward Differentiation	@SSET(630)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

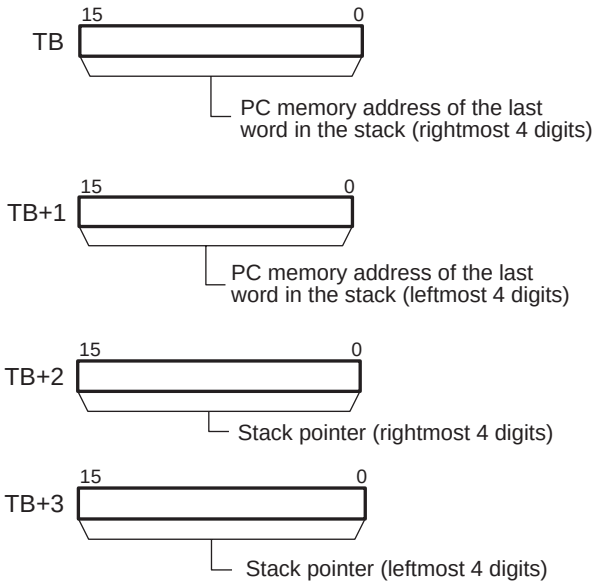
Applicable Program Areas

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

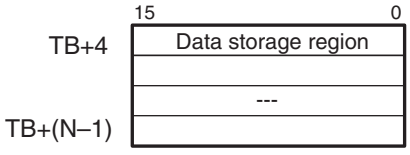
Operands

TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next word to be overwritten by PUSH(632)).



TB+4 through TB+(N–1): Data storage region
The remainder of the stack is used to store data.



Note
1. The initial value of the stack pointer is always the PLC memory address of TB+4.
2. TB and TB+(N–1) must be in the same data area.

Operand Specifications

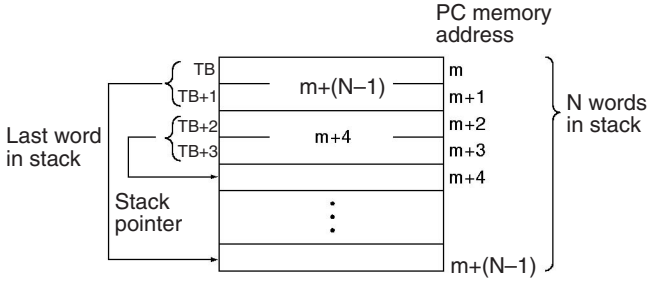
Area	TB	N
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	A000 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	#0005 to #FFFF (binary) or &5 to &65,535

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

Description

SSET(630) secures a stack with N words beginning at TB and ending at TB+(N–1). The first two words of the stack (TB+1 and TB) contain the 8-digit hexadecimal PLC memory address of the last word in the stack. The next two words (TB+3 and TB+2) contain the stack pointer. The stack pointer is the PLC memory address of the next word in the stack that will be overwritten by PUSH(632); its initial value is the address of TB+4.

SSET(630) automatically initializes the data region of the stack (TB+4 through TB+(N–1)) to zeroes. The following diagram shows the basic structure of a stack.



SSET(630) just establishes and initializes a stack. Use the following instructions to store in the stack and read data from the stack.

1,2,3...

1. PUSH(632) stores data in the stack one word at a time.
2. FIFO(633) and LIFO(634) read data from the stack. FIFO(633) reads the first word that was stored; LIFO(634) reads the last word that was stored.
3. The stack pointer value in the stack control word is automatically updated when PUSH(632), FIFO(633), or LIFO(634) is executed. Normally, users need not be concerned about the stack control word. When accessing the contents of the stack other than by using the above instructions, set the stack pointer value using the Index Register (IR) for indirect referencing.

Flags

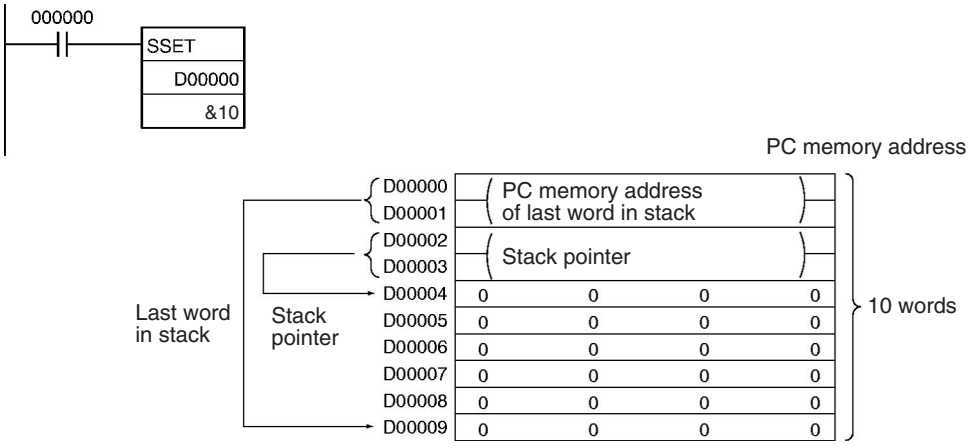
Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0005 to FFFF. OFF in all other cases.

Precautions

The minimum value for the number of words in the stack (N) is 5 because N includes the four words that contain the pointer to the last word in the stack and the stack pointer. An error will occur if N is not in the range 0005 to FFFF.

Examples

When CIO 000000 is ON in the following example, SSET(630) secures a 10-word stack from D00000 to D00009. D00000 and D00001 contain the PLC memory address of the last word in the stack. D00002 and D00003 contain the stack pointer. The stack itself begins in D00004.

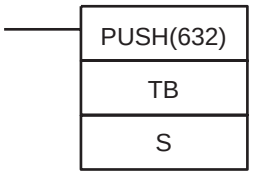


3-17-2 PUSH ONTO STACK: PUSH(632)

Purpose

Writes one word of data to the specified stack.

Ladder Symbol



TB: First stack address

S: Source word

Variations

Variations		
	Executed Each Cycle for ON Condition	PUSH(632)
	Executed Once for Upward Differentiation	@PUSH(632)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

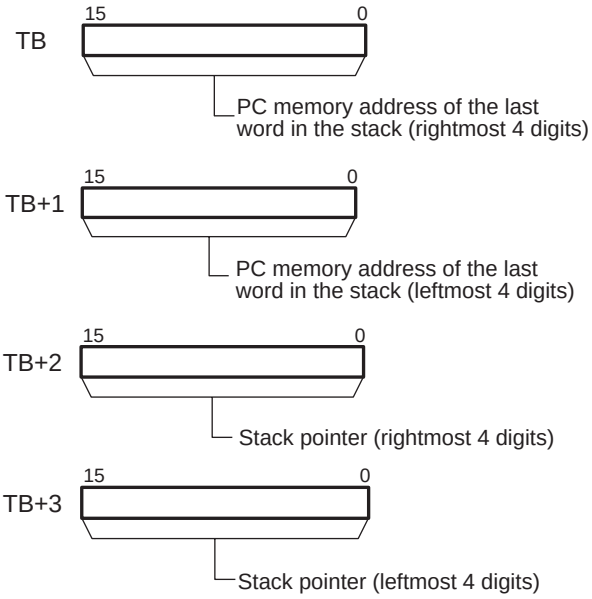
Applicable Program Areas

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

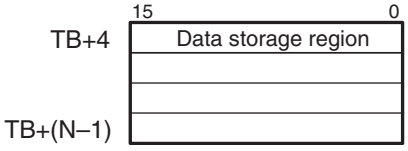
Operands

TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next word to be overwritten by PUSH(632)).



TB+4 through TB+(N–1): Data storage region
The remainder of the stack is used to store data.



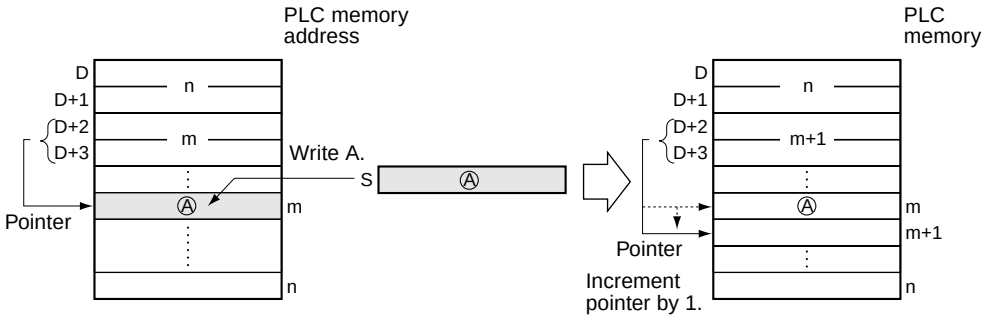
Operand Specifications

Area	TB	S
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	A000 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	#0000 to #FFFF (binary)
Data Registers	---	DR0 to DR15

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

Description

PUSH(632) writes the content of S to the address indicated by the stack pointer (TB+3 and TB+2) and increments the stack pointer by one.



After PUSH(632) has been used to write data into a stack, FIFO(633) and LIFO(634) can be used to read data from the stack.

Flags

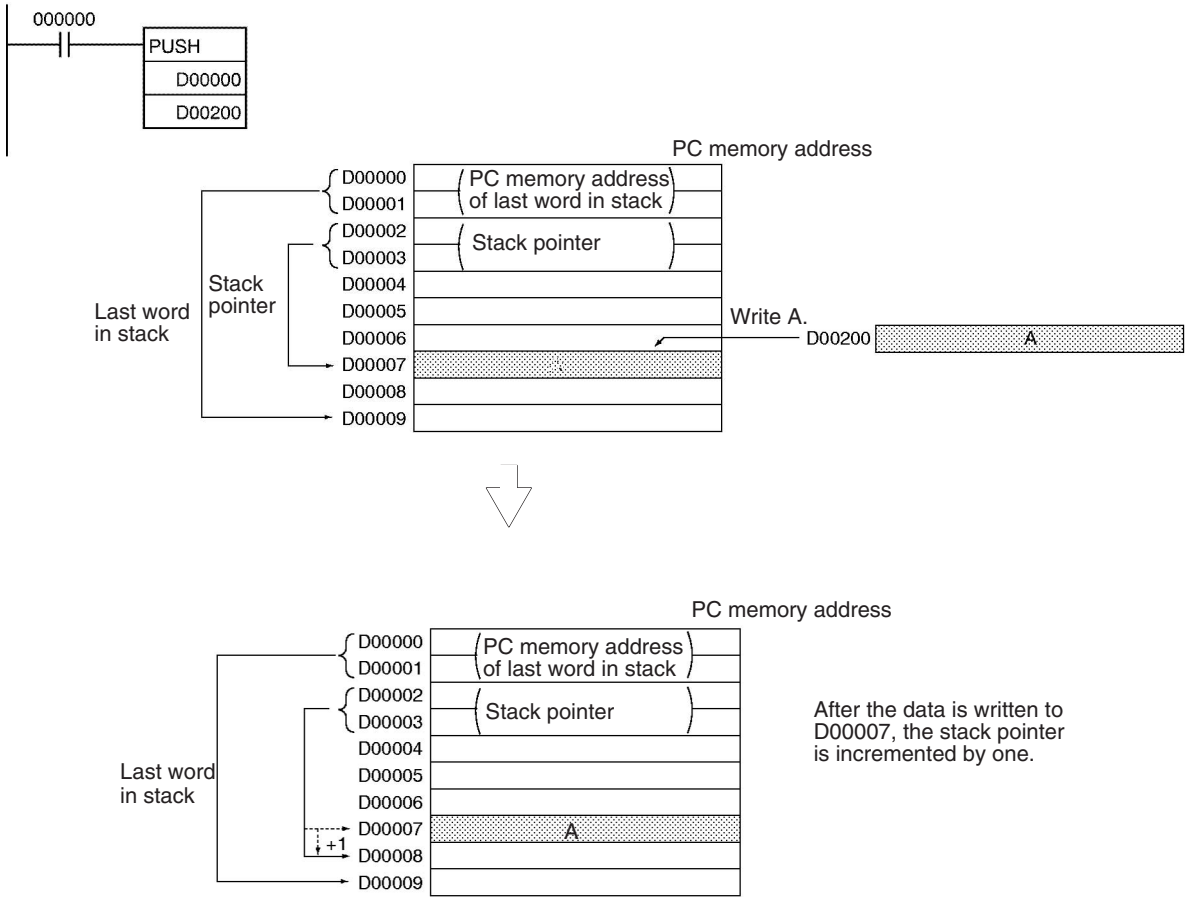
Name	Label	Operation
Error Flag	ER	ON if the address specified by the stack pointer (TB+3 and TB+2) exceeds the last word in the stack. (This is a stack overflow error.) OFF in all other cases.

Precautions

The stack must be defined in advance with SSET(630).

Examples

When CIO 000000 is ON in the following example, PUSH(632) copies the content of D00200 to the stack beginning at D00000. In this case, the stack pointer indicates D00007.

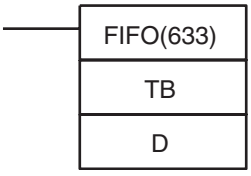


3-17-3 FIRST IN FIRST OUT: FIFO(633)

Purpose

Reads the first word of data written to the specified stack (the oldest data in the stack).

Ladder Symbol



TB: First stack address

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FIFO(633)
	Executed Once for Upward Differentiation	@FIFO(633)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

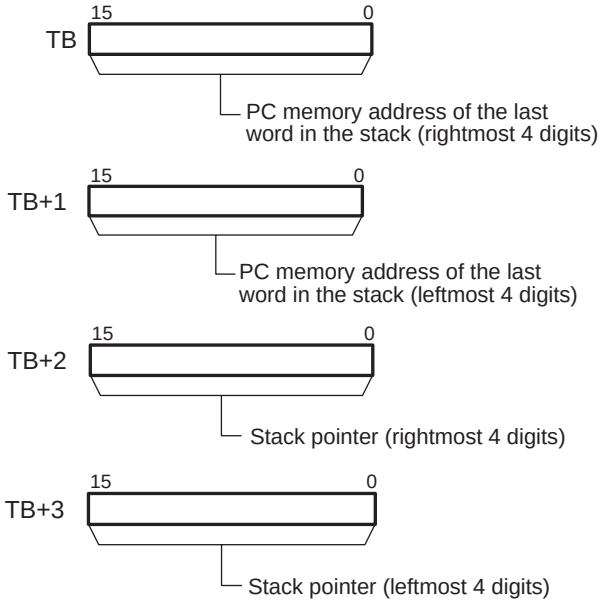
Applicable Program Areas

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

Operands

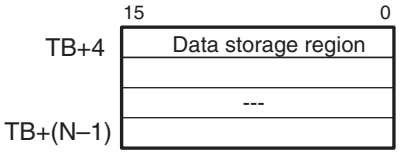
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next word to be overwritten by PUSH(632)).



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.



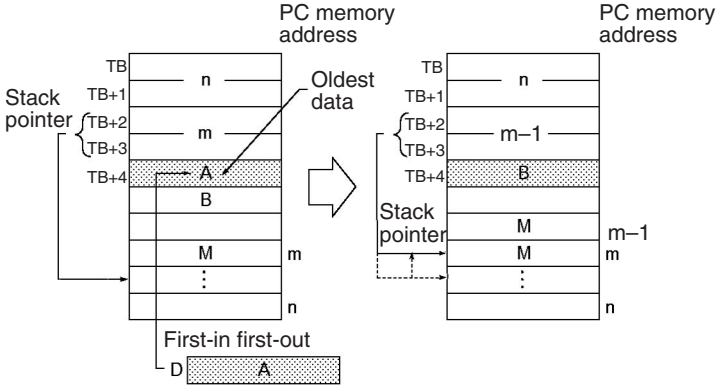
Operand Specifications

Area	TB	D
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	

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

Description

FIFO(633) reads the oldest word of data from the stack (TB+4) and outputs that data to D. Next, the stack pointer (TB+3 and TB+2) is decremented by one, all of the remaining data in the stack is shifted downward by one word, and the data read from TB+4 is deleted. The data at the end of the stack (the address that was indicated by the stack pointer) is left unchanged.



Use FIFO(633) in combination with PUSH(632). After PUSH(632) has been used to write data into a stack, FIFO(633) can be used to read data from the stack on a first-in first-out basis.

FIFO(633) reads the beginning data from the stack and deletes this data to move the next one forward.

Flags

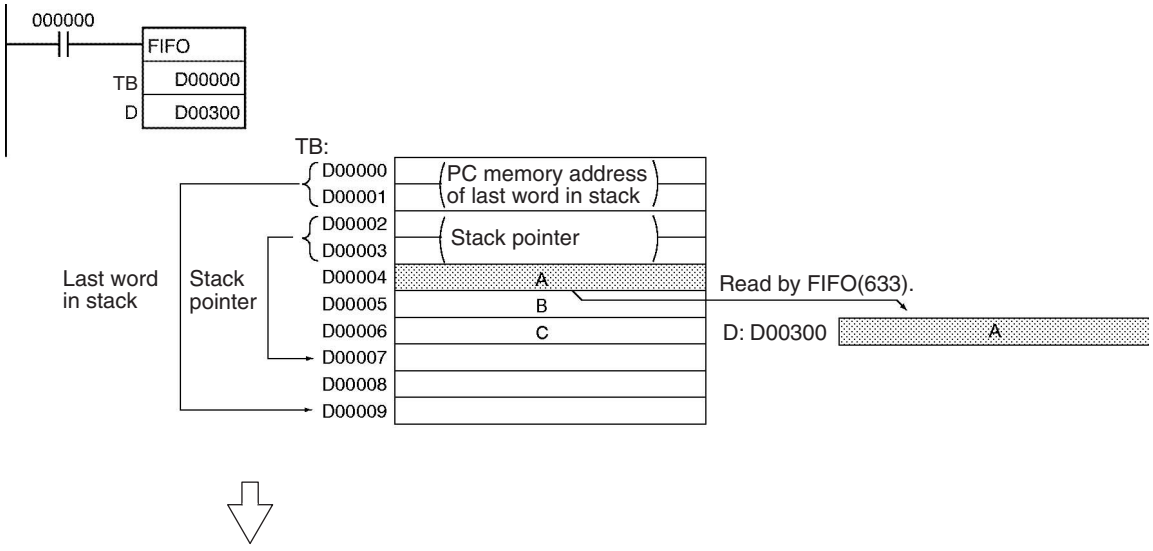
Name	Label	Operation
Error Flag	ER	ON if the contents of the stack pointer (TB+3 and TB+2) is less than or equal to the PLC memory address of first word in the data region of the stack (TB+4). (This is a stack underflow error.) OFF in all other cases.

Precautions

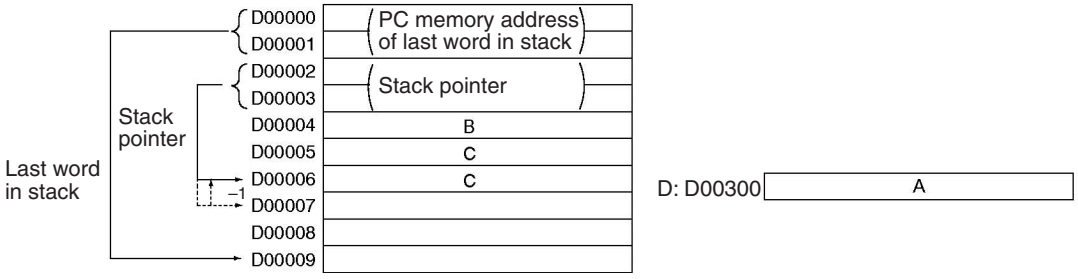
The stack must be defined in advance with SSET(630).

Examples

When CIO 000000 is ON in the following example, FIFO(633) reads the content of D00004 (TB+4 for the stack beginning at D00000) and writes that data to D00300.



After the data is written to D00300, the stack pointer is decremented by one and the remaining data is shifted down. (The content of D00005 is shifted to D00004 and the content of D00006 is shifted to D00005.)

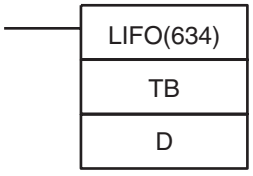


3-17-4 LAST IN FIRST OUT: LIFO(634)

Purpose

Reads the last word of data written to the specified stack (the newest data in the stack).

Ladder Symbol



TB: First stack address

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	LIFO(634)
	Executed Once for Upward Differentiation	@LIFO(634)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

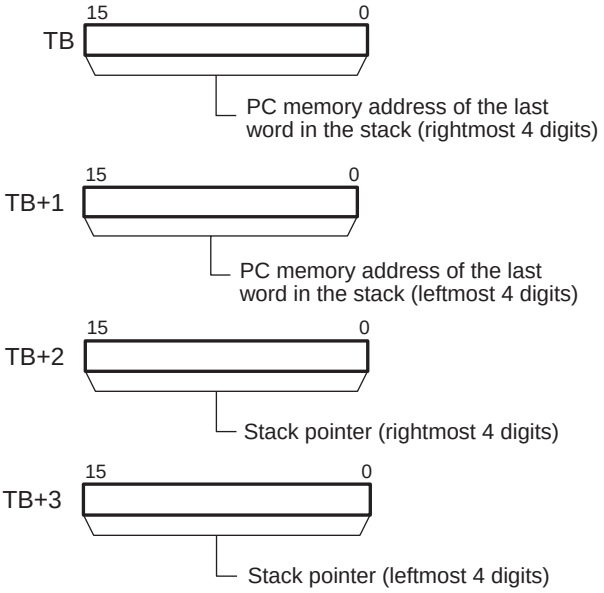
Applicable Program Areas

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

Operands

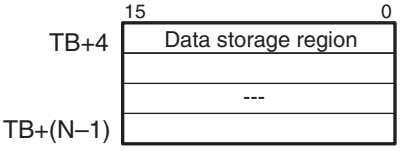
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next word to be overwritten by PUSH(632)).



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.



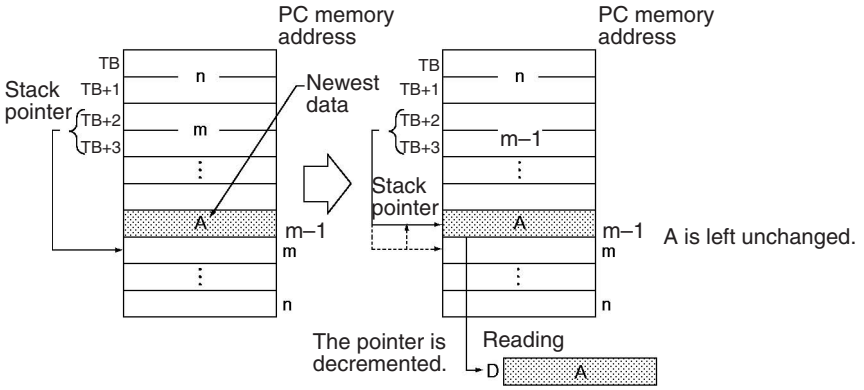
Operand Specifications

Area	TB	D
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	DR0 to DR15

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

Description

LIFO(634) reads the data from the address indicated by the stack pointer (the newest word of data in the stack), decrements the stack pointer by one, and outputs the data to D. The word that was read is left unchanged.



Use LIFO(634) in combination with PUSH(632). After PUSH(632) has been used to write data into a stack, LIFO(634) can be used to read data from the stack on a last-in first-out basis. After data is stored by PUSH(632), the stack pointer indicates the address next to the last data.

Flags

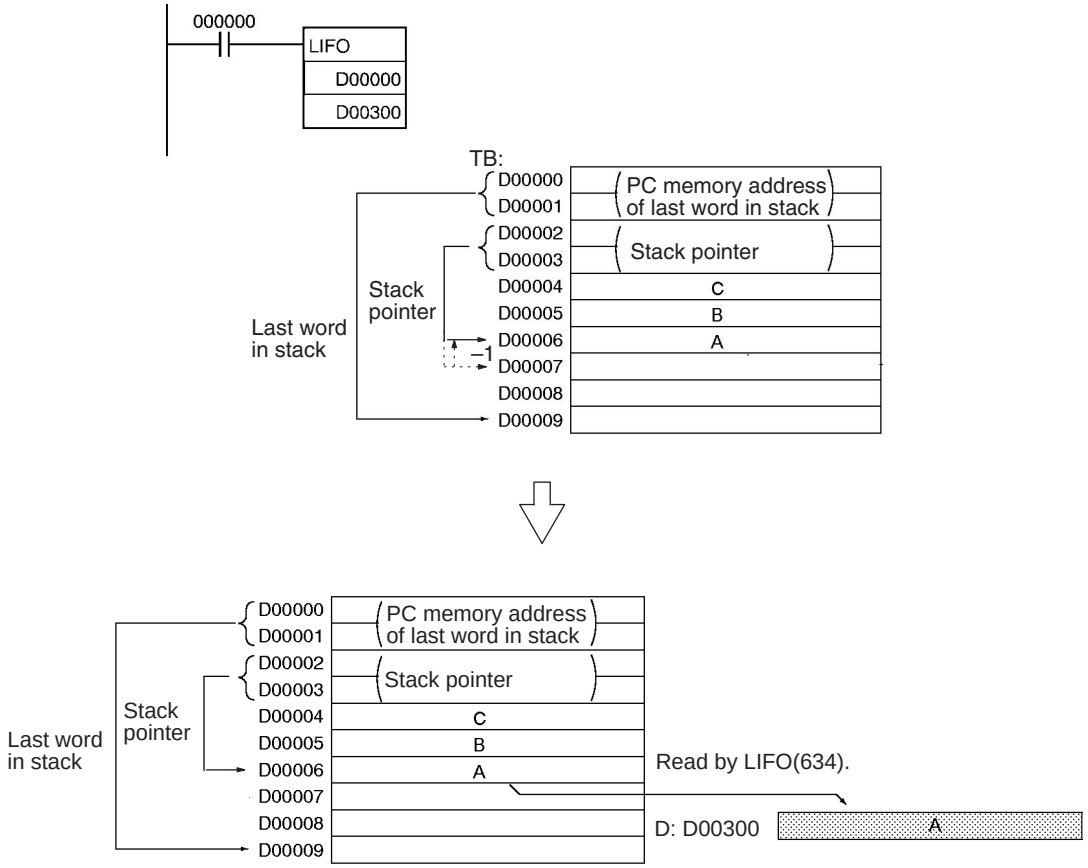
Name	Label	Operation
Error Flag	ER	ON if the contents of the stack pointer (TB+3 and TB+2) is less than or equal to the PLC memory address of first word in the data region of the stack (TB+4). (This is a stack underflow error.) OFF in all other cases.

Precautions

The stack must be defined in advance with SSET(630).

Examples

When CIO 000000 is ON in the following example, LIFO(634) reads the content of the word indicated by the stack pointer (D00006) and writes that data to D00300.



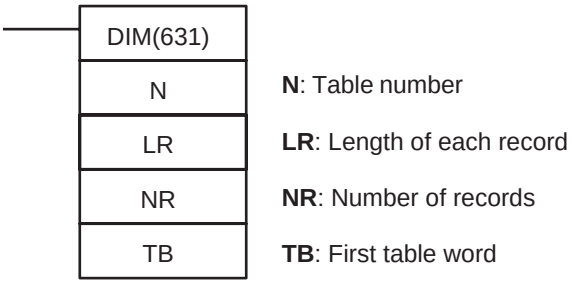
After the data is written to D00300, the stack pointer is decremented by one. The content of D00006 is left unchanged.

3-17-5 DIMENSION RECORD TABLE: DIM(631)

Purpose

Defines the specified I/O memory area as a record table by declaring the length of each record and the number of records. Up to 16 record tables can be defined.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	DIM(631)
	Executed Once for Upward Differentiation	@DIM(631)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Table number
Indicates the table number. N must be between 0 and15.

LR: Length of each record
Indicates the number of words in each record. LR must be 0001 to FFFF hexadecimal (1 to 65,535 words).

NR: Number of records
Indicates the number of records in the table. NR must be 0001 to FFFF hexadecimal (1 to 65,535 words).

TB: First table word
Indicates the first word of the table. All of the words in the table must be in the same data area. In other words TB and TB+LR×NR–1 must be in the same data area.

Operand Specifications

Area	N	LR	NR	TB
CIO Area	---	CIO 0000 to CIO 6143		
Work Area	---	W000 to W511		
Holding Bit Area	---	H000 to H511		
Auxiliary Bit Area	---	A000 to A959		A448 to A959
Timer Area	---	T0000 to T4095		
Counter Area	---	C0000 to C4095		
DM Area	---	D00000 to D32767		
EM Area without bank	---	E00000 to E32767		
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	0 to 15	#0001 to #FFFF (binary) or &1 to &65,535		---
Data Registers	---	DR0 to DR15		---
Index Registers	---	---		
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

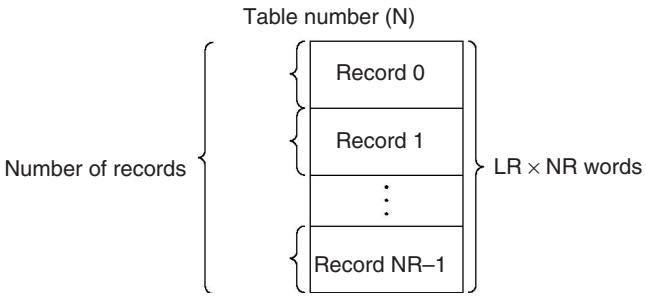
Description

DIM(631) registers the words from TB to TB+LR×NR–1 as table number N. Table number N has NR records and each record is LR words long. The data within this region cannot be changed once the region has been declared as records.

Use DIM(631) in combination with SETR(635) (SET RECORD NUMBER) or GETR(636) (GET RECORD NUMBER) to simplify the calculation of

addresses in data tables. Use DIM(631) to divide data into records and then use SETR(635) to store the first address of the desired record in an Index Register. The Index Register can then be used as a pointer in other instructions, such as read, write, search, or compare instructions.

As an example, if temperatures, pressures, or other set values are stored as records and the records for various models are combined into a table, it is easy to read the set values for each models for any particular conditions.



The two record-table instructions associated with DIM(631) are SETR(635) and GETR(636). SETR(635) sets the leading PLC memory address of the specified record number in the specified Index Register. GETR(636) outputs the record number of the record that includes the specified Index Register value (PLC memory address).

Flags

Name	Label	Operation
Error Flag	ER	ON if LR or NR is 0000. OFF in all other cases.

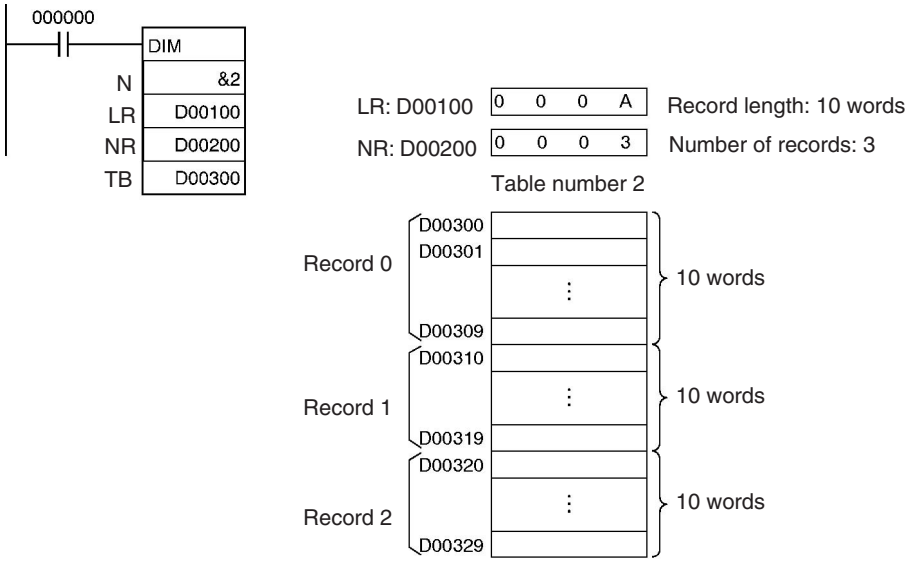
Precautions

Records in a registered table are identified by their record numbers, which range from 0 to NR–1.

Depending on the settings for the record length (LR) and number of records (NR), it is possible that a single table (from TB and TB+LR×NR–1) will overlap two data areas. Verify that no problems will arise before specifying a table that overlaps a data area boundary.

Examples

When CIO 000000 is ON in the following example, DIM(631) defines record table number 2 with three 10-word records. The table begins at D00300.

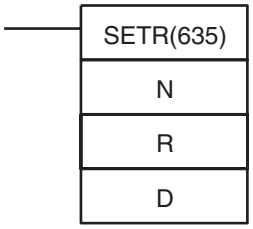


3-17-6 SET RECORD LOCATION: SETR(635)

Purpose

Writes the location of the specified record (the PLC memory address of the beginning of the record) in the specified Index Register.

Ladder Symbol



N: Table number

R: Record number

D: Destination Index Register

Variations

Variations		
	Executed Each Cycle for ON Condition	SETR(635)
	Executed Once for Upward Differentiation	@SETR(635)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Table number
Indicates the table number. N must be between 0 and 15.

R: Record number
Indicates the record number of the desired record. R must be 0000 to FFFE hexadecimal (0 to 65,534). Record numbers begin with 0, so the valid record numbers are 0 to NR–1 for a table with NR records.

D: Destination Index Register
Indicates the desired Index Register. D must be IR0 to IR15.

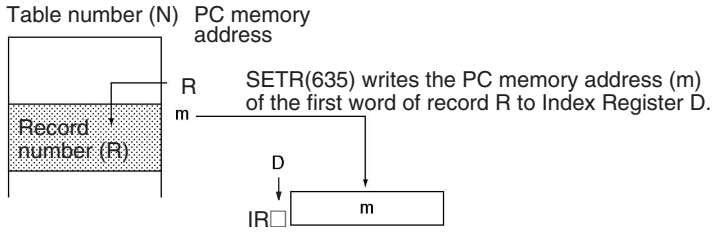
Operand Specifications

Area	N	R	D
CIO Area	---	CIO 0000 to CIO 6143	---
Work Area	---	W000 to W511	---
Holding Bit Area	---	H000 to H511	---
Auxiliary Bit Area	---	A000 to A959	---
Timer Area	---	T0000 to T4095	---
Counter Area	---	C0000 to C4095	---
DM Area	---	D00000 to D32767	---
EM Area without bank	---	E00000 to E32767	---
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)	---
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767	---
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767	---
Constants	0 to 15	#0000 to #FFFE (binary) or &0 to 65534	---
Data Registers	---	DR0 to DR15	---

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

Description

SETR(635) stores the PLC memory address of the first word of the specified record in the specified Index Register. The following diagram shows the basic operation of SETR(635).



Flags

Name	Label	Operation
Error Flag	ER	ON if the specified table number (N) has not been defined with DIM(631). ON if the specified record number (R) exceeds the highest record number in the table (NR–1). OFF in all other cases.

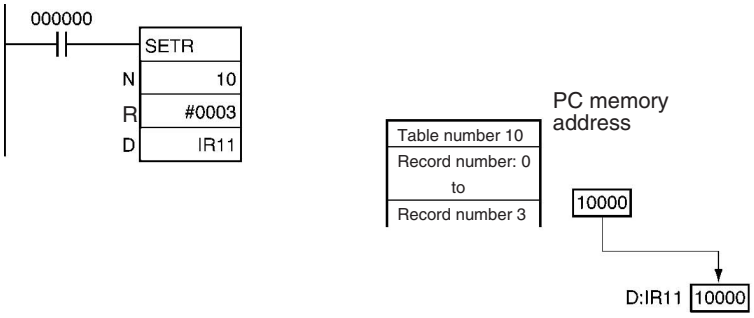
Precautions

The record table must be defined in advance with DIM(631).

Valid record numbers range from 0 to NR–1, where NR is the number of records specified when the table was defined with DIM(631).

Examples

When CIO 000000 is ON in the following example, SETR(635) finds the PLC memory address of the first word of record 3 of table number 10 and stores this address in Index Register IR11.

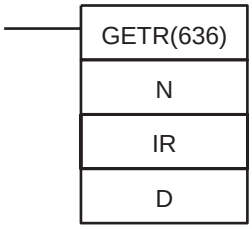


3-17-7 GET RECORD NUMBER: GETR(636)

Purpose

Returns the record number of the record at the PLC memory address contained in the specified Index Register.

Ladder Symbol



N: Table number

IR: Index Register

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	GETR(636)
	Executed Once for Upward Differentiation	@GETR(636)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Table number
Indicates the table number. N must be between 0 and 15.

IR: Index Register
Indicates the desired Index Register. D must be IR0 to IR15.

D: Destination word
Indicates the word where the record number will be written.

Operand Specifications

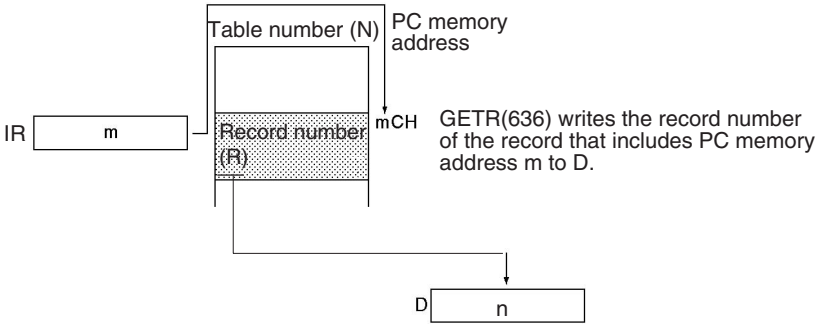
Area	N	IR	D
CIO Area	---		CIO 0000 to CIO 6143
Work Area	---		W000 to W511
Holding Bit Area	---		H000 to H511
Auxiliary Bit Area	---		A448 to A959
Timer Area	---		T0000 to T4095
Counter Area	---		C0000 to C4095
DM Area	---		D00000 to D32767
EM Area without bank	---		E00000 to E32767
EM Area with bank	---		En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---		@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---		*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	0 to 15	---	---
Data Registers	---		DR0 to DR15

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

Description

GETR(636) finds which record includes the PLC memory address contained in the specified Index Register and writes that record number in D. The PLC memory address contained in the Index Register does not have to be the first word in the record; it can be any word in the record.

The following diagram shows the basic operation of GETR(636).



Flags

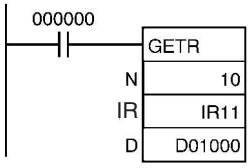
Name	Label	Operation
Error Flag	ER	ON if the PLC memory address in the specified Index Register is not within the specified table (N). ON if the specified table number (N) has not been defined with DIM(631). OFF in all other cases.

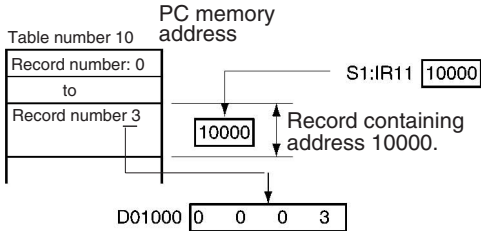
Precautions

The record table must be defined in advance with DIM(631) and the PLC memory address in the specified Index Register must be within the specified table.

Examples

When CIO 000000 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 D01000.



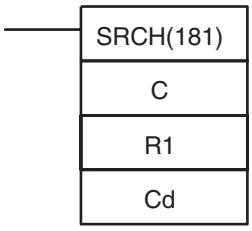


3-17-8 DATA SEARCH: SRCH(181)

Purpose

Searches for a word of data within a range of words.
In CS1D CPU Units for Single-CPU Systems and CS1-H, CJ1-H, and CJ1M CPU Units, this instruction can be run in the background. Refer to the *CS/CJ Series Programmable Controllers Programming Manual* for details on background execution.

Ladder Symbol



C: First control word

R1: First word in range

Cd: Comparison data

Variations

Variations		
	Executed Each Cycle for ON Condition	SRCH(181)
	Executed Once for Upward Differentiation	@SRCH(181)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

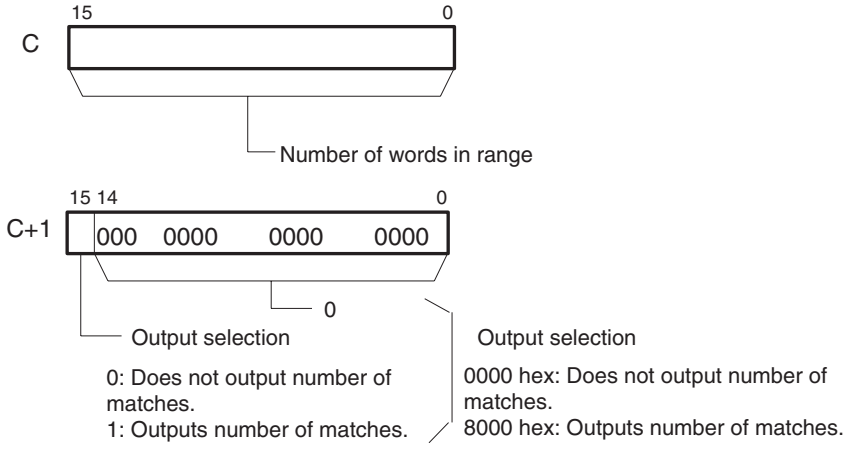
Applicable Program Areas

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

Operands

C and C+1: Control words

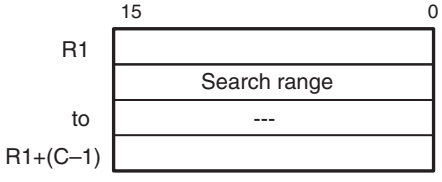
C specifies the number of words in the range and bit 15 of C+1 indicates whether or not to output the number of matches to DR00.



Note C and C+1 must be in the same data area.

R1: First word in range

R1 specifies the first word in the search range. The words from R1 to R1+(C–1) are searched for the desired data. (C is the number of words set in C.)



Note R1 and R1+C–1 must be in the same data area.

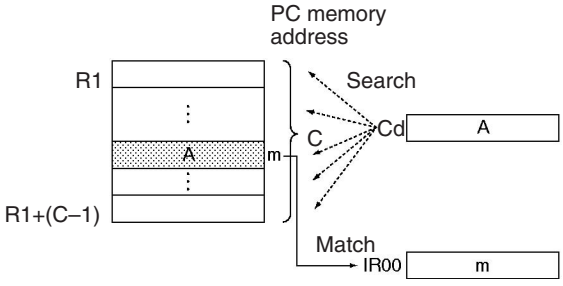
Operand Specifications

Area	C	R1	Cd
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143	
Work Area	W000 to W510	W000 to W511	
Holding Bit Area	H000 to H510	H000 to H511	
Auxiliary Bit Area	A000 to A958	A000 to A959	
Timer Area	T0000 to T4094	T0000 to T4095	
Counter Area	C0000 to C4094	C0000 to C4095	
DM Area	D00000 to D32766	D00000 to D32767	
EM Area without bank	E00000 to E32766	E00000 to E32767	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	Specified values only	---	#0000 to #FFFF (binary)
Data Registers	---		DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SRCH(181) searches the range of memory from R1 to R1+C–1 for words that contain the comparison data (Cd). If a match is found, SRCH(181) writes the PLC memory address of the word to IR00 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 IR00.)

When bit 15 of C+1 has been set to 1, SRCH(181) writes the number of matches to DR00. When bit 15 of C+1 is 0, DR00 is left unchanged.



SRCH(181) searches table data that contains one word in each record. For searching data that contains more than one word per record, use DIM(631), SETR(635), GETR(636), FOR(512)–NEXT(513), or BREAK(514) together with an Index Register (IR).

The status of the Equals Flag can be checked immediately after execution to determine whether or not there was a match.

Note SRCH(181) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Related Memory Area Words

Name	Address	Operation
IR00 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 IR00. (A595 contains the rightmost digits, and A596 contains the leftmost digits.)
DR00 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 DR00.
Equals Flag for Background Execution	A59801	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	A39510	This flag is turned ON if an error or illegal access occurs during background execution.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. OFF in all other cases.
Equals Flag	=	ON if one or more of the words in the search range contain the comparison 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.

Precautions

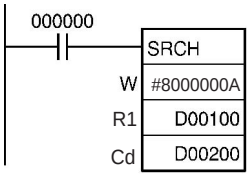
If no match is found, the contents of IR00 and DR00 are left unchanged.

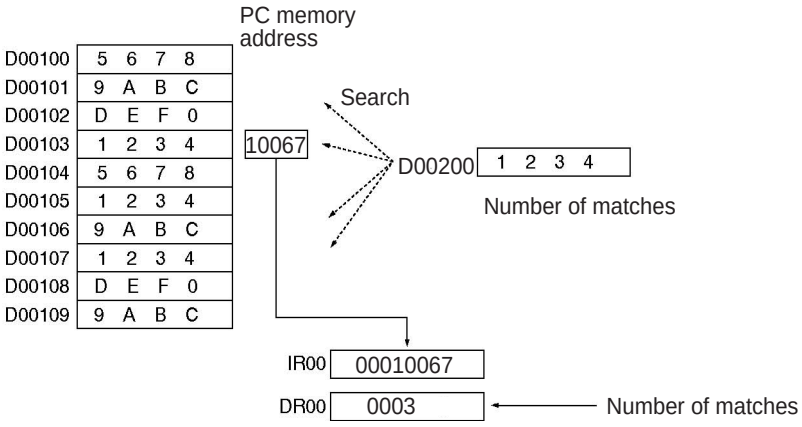
If background execution is enabled in the PLC Setup, the PLC memory address of the first word containing a match will be output to Auxiliary Area words A595 and A596 instead of IR00.

If background execution is enabled in the PLC Setup and control word C+1 is set to output the total number of matches to DR00 (C+1 = 8000 hex), the total number of matches will be output to Auxiliary Area word A597 instead of DR00.

Examples

When CIO 000000 is ON in the following example, SRCH(181) searches the 10-word range beginning at D00100 for words that have the same content as D00200. The PLC memory address of the first word containing a match is written to IR00 and the total number of matches is written to DR00.





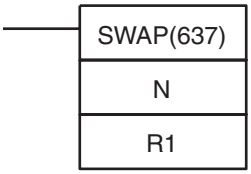
If the table length is specified as &10 (10 decimal) or A hexadecimal, the number of matches will not be output to the data register DR00.

3-17-9 SWAP BYTES: SWAP(637)

Purpose

Switches the leftmost and rightmost bytes in all of the words in the range.
In CS1D CPU Units for Single-CPU Systems and CS1-H, CJ1-H, and CJ1M CPU Units, this instruction can be run in the background. Refer to *CS/CJ Series Programmable Controllers Programming Manual* for details on background execution.

Ladder Symbol



N: Number of words

R1: First word in range

Variations

Variations		
	Executed Each Cycle for ON Condition	SWAP(637)
	Executed Once for Upward Differentiation	@SWAP(637)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

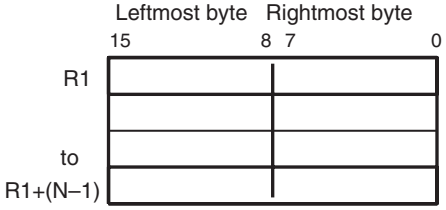
Applicable Program Areas

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

Operands

N: Number of words
N specifies the number of words in the range and must be 0001 to FFFF hexadecimal (or &1 to &65,535).

R1: First word in range
R1 specifies the first word in the range. R1 and R1+(N–1) must be in the same data area.

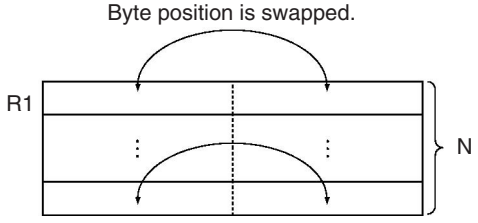


Operand Specifications

Area	N	R1
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0001 to #FFFF (binary) or &1 to &65,535	---
Data Registers	DR00 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

SWAP(637) switches the position of the two bytes in all of the words in the range of memory from R1 to R1+N–1. This instruction can be used to reverse the order of ASCII-code characters in each word.



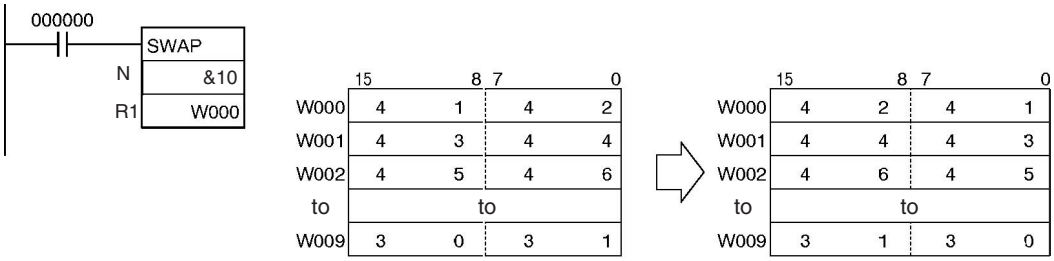
Note SWAP(637) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if the N is 0000. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

Examples

When CIO 000000 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 W000 to W009.



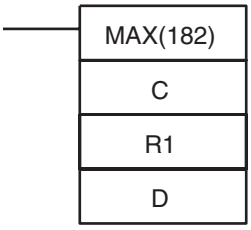
3-17-10 FIND MAXIMUM: MAX(182)

Purpose

Finds the maximum value in the range.

In CS1D CPU Units for Single-CPU Systems and CS1-H, CJ1-H, and CJ1M CPU Units, this instruction can be run in the background. Refer to *CS/CJ Series Programmable Controllers Programming Manual* for details on background execution.

Ladder Symbol



C: First control word

R1: First word in range

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MAX(182)
	Executed Once for Upward Differentiation	@MAX(182)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

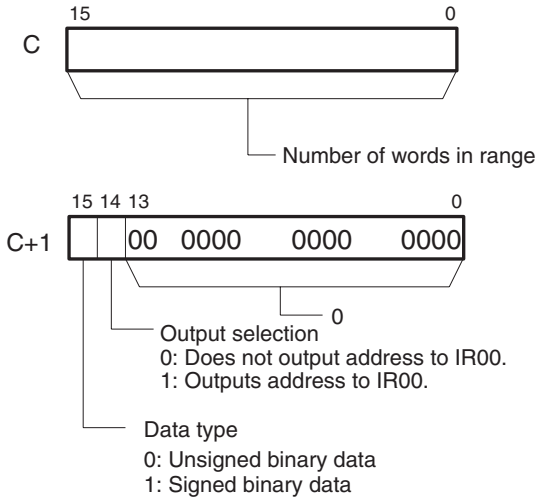
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C and C+1: Control words

C specifies the number of words in the range, bit 15 of C+1 indicates whether the data will be treated as signed binary or unsigned binary, and bit 14 of C+1 indicates whether or not to output the PLC memory address of the word that contains the maximum value to IR00.

Note C and C+1 must be in the same data area.

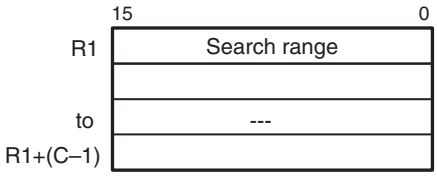


The following table shows the possible values of C.

C+1	Data type	Index Register output
0000	Unsigned binary	No
4000	Unsigned binary	Yes
8000	Signed binary	No
C000	Signed binary	Yes

R1: First word in range

R1 specifies the first word in the search range. The words from R1 to R1+(C–1) are searched for the maximum value. (C is the number of words specified in C.)



Note R1 and R1+(C–1) must be in the same data area.

Operand Specifications

Area	C	R1	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143	
Work Area	W000 to W510	W000 to W511	
Holding Bit Area	H000 to H510	H000 to H511	
Auxiliary Bit Area	A000 to A958	A000 to A959	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095	
Counter Area	C0000 to C4094	C0000 to C4095	
DM Area	D00000 to D32766	D00000 to D32767	
EM Area without bank	E00000 to E32766	E00000 to E32767	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)	

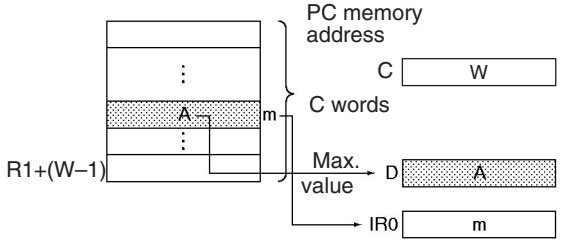
Area	C	R1	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	Specified values only	---	
Data Registers	---		DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

MAX(182) searches the range of memory from R1 to R1+C–1 for the maximum value in the range and outputs that maximum value to D.

When bit 14 of C+1 has been set to 1, MAX(182) writes the PLC memory address of the word containing the maximum value to IR00. (If two or more words within the range contain the maximum value, the address of the first word containing the maximum value is written to IR00.)

When bit 15 of C+1 has been set to 1, MAX(182) treats the data within the range as signed binary data.



Note MAX(182) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Related Memory Area Words

Name	Address	Operation
IR00 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 IR00. (A595 contains the rightmost digits, and A596 contains the leftmost digits.)
ER/AER Flag for Background Execution	A39510	This flag is turned ON if an error or illegal access occurs during background execution.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if the maximum value is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 is ON in the word containing the maximum value. OFF in all other cases.

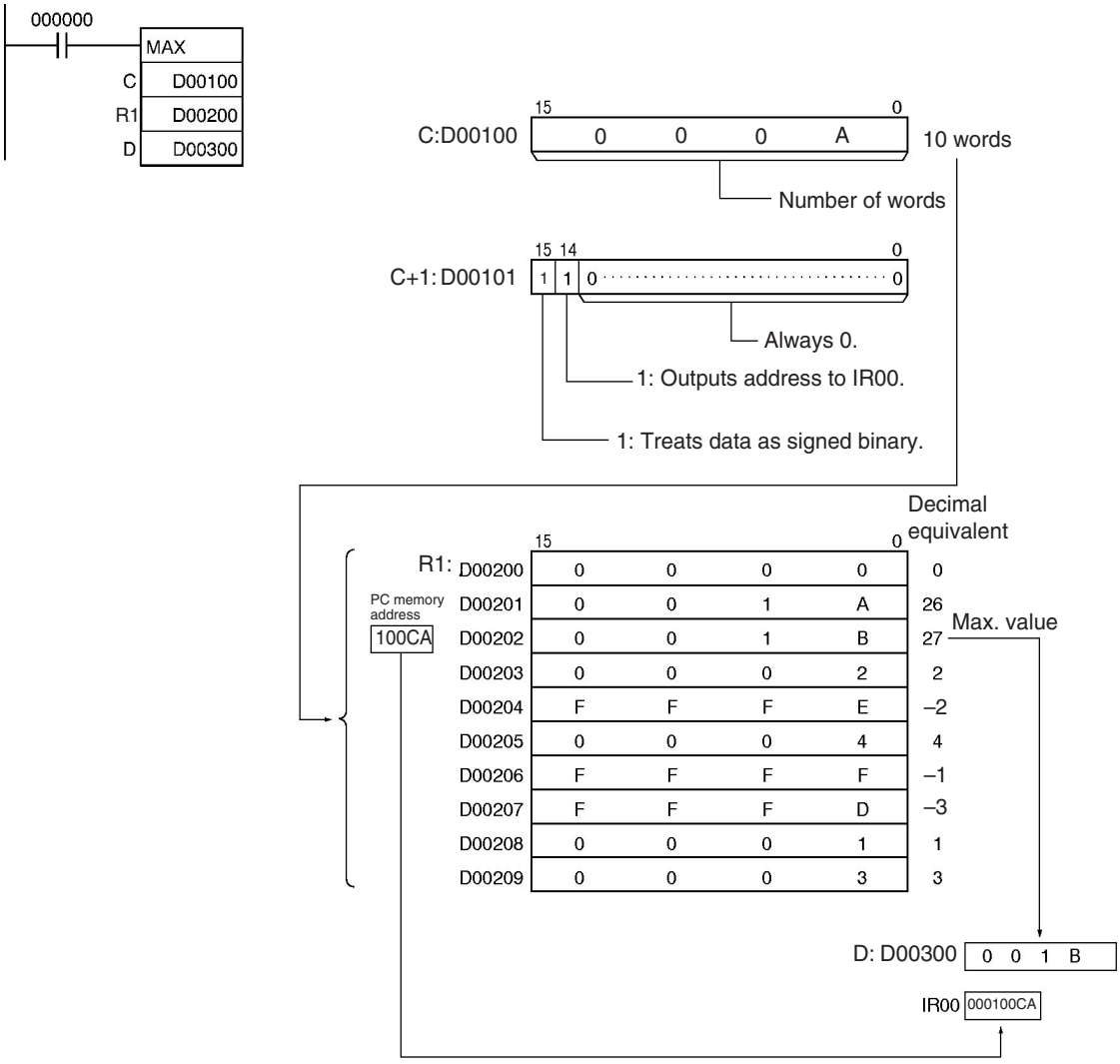
Precautions

When bit 15 of C+1 has been set to 1, the data within the range is treated as signed binary data and hexadecimal values 8000 to FFFF are considered negative. Thus, the results of the search will differ depending on the data-type setting.

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 IR00.

Examples

When CIO 000000 turns ON in the following example, MAX(182) searches the 10-word range beginning at D00200 for the maximum value. The maximum value is written to D00300 and the PLC memory address of the word containing the maximum value is written to IR00.



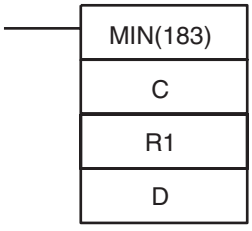
3-17-11 FIND MINIMUM: MIN(183)

Purpose

Finds the minimum value in the range.

In CS1D CPU Units for Single-CPU Systems and CS1-H, CJ1-H, and CJ1M CPU Units, this instruction can be run in the background. Refer to the *CS/CJ Series Programmable Controllers Programming Manual* for details on background execution.

Ladder Symbol



C: First control word

R1: First word in range

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MIN(183)
	Executed Once for Upward Differentiation	@MIN(183)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

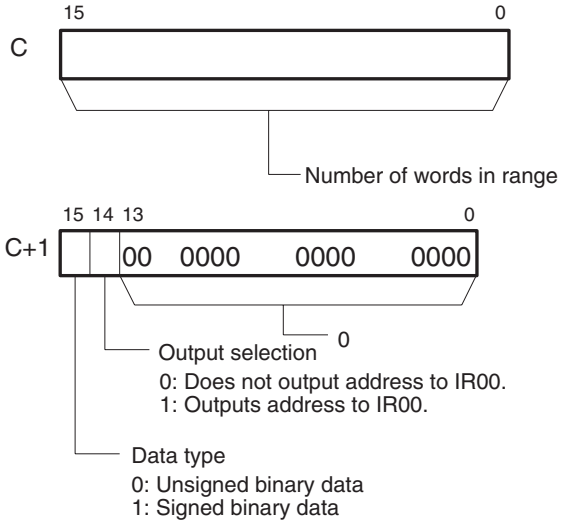
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C and C+1: Control words

C specifies the number of words in the range, bit 15 of C+1 indicates whether the data will be treated as signed binary or unsigned binary, and bit 14 of C+1 indicates whether or not to output the PLC memory address of the word that contains the minimum value to IR00.

Note C and C+1 must be in the same data area.

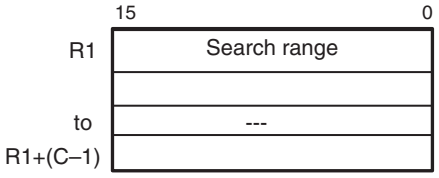


The following table shows the possible values of C.

C+1	Data type	Index Register output
0000	Unsigned binary	No
4000	Unsigned binary	Yes
8000	Signed binary	No
C000	Signed binary	Yes

R1: First word in range

R1 specifies the first word in the search range. The words from R1 to R1+(C–1) are searched for the minimum value. (C is the number of words specified in C.)



Note R1 and R1+C–1 must be in the same data area.

Operand Specifications

Area	C	R1	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143	
Work Area	W000 to W510	W000 to W511	
Holding Bit Area	H000 to H510	H000 to H511	

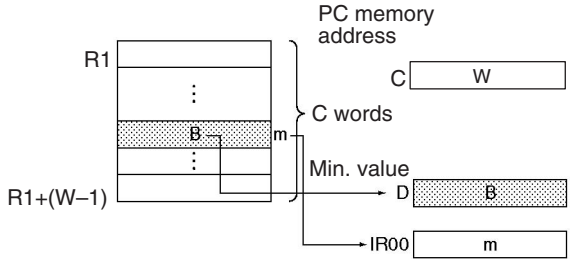
Area	C	R1	D
Auxiliary Bit Area	A000 to A958	A000 to A959	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095	
Counter Area	C0000 to C4094	C0000 to C4095	
DM Area	D00000 to D32766	D00000 to D32767	
EM Area without bank	E00000 to E32766	E00000 to E32767	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D0000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	Specified values only	---	
Data Registers	---		DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

MIN(183) searches the range of memory from R1 to R1+C–1 for the minimum value in the range and outputs that minimum value to D.

When bit 14 of C+1 has been set to 1, MIN(183) writes the PLC memory address of the word containing the minimum value to IR00. (If two or more words within the range contain the minimum value, the address of the first word containing the minimum value is written to IR00.)

When bit 15 of C+1 has been set to 1, MIN(183) treats the data within the range as signed binary data.



Note MIN(183) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Related Memory Area Words

Name	Address	Operation
IR00 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 IR00. (A595 contains the rightmost digits, and A596 contains the leftmost digits.)
ER/AER Flag for Background Execution	A39510	This flag is turned ON if an error or illegal access occurs during background execution.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if the minimum value is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 is ON in the word containing the minimum value. OFF in all other cases.

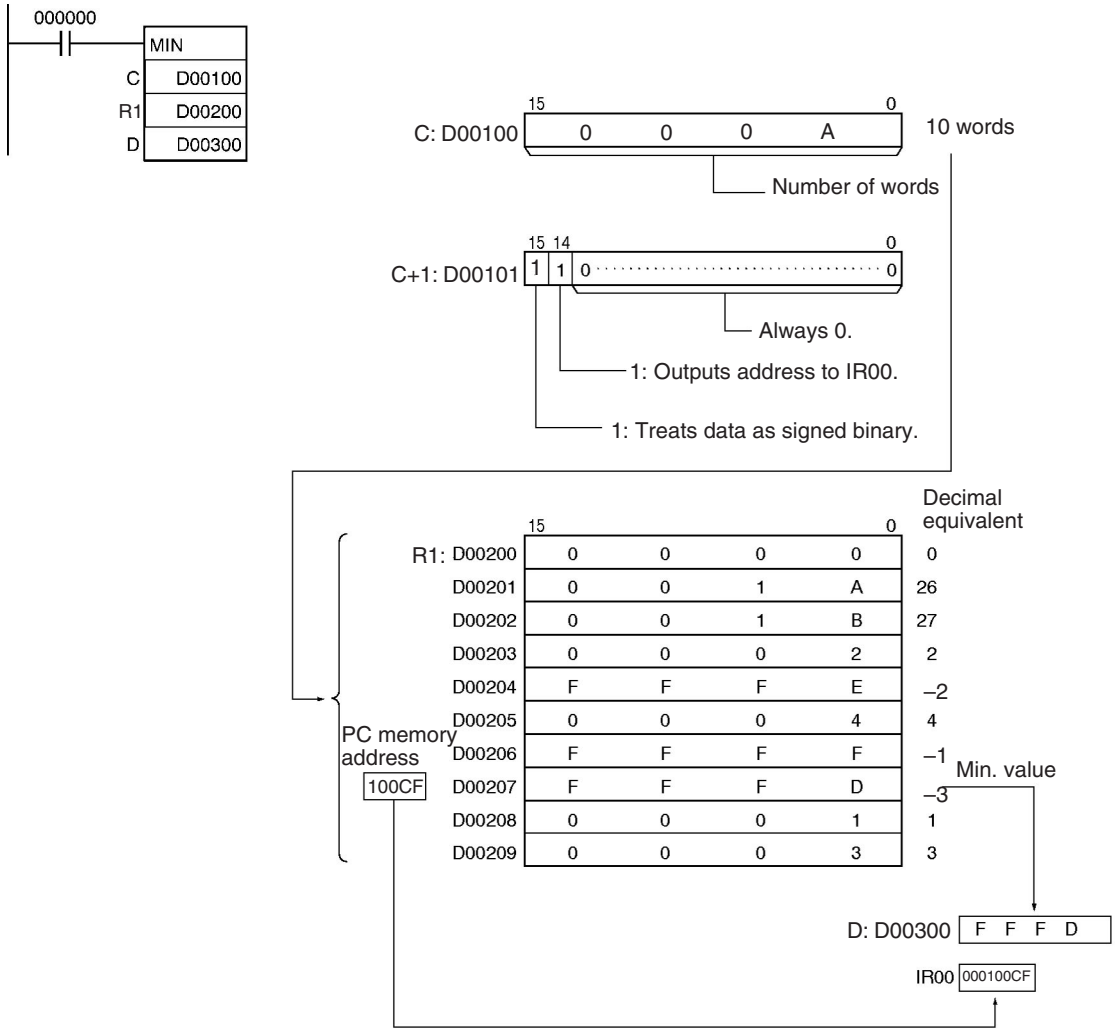
Precautions

When bit 15 of C+1 has been set to 1, the data within the range is treated as signed binary data and hexadecimal values 8000 to FFFF are considered negative. Thus, the results of the search will differ depending on the data-type setting.

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 IR00.

Examples

When CIO 000000 turns ON in the following example, MIN(183) searches the 10-word range beginning at D00200 for the minimum value. The minimum value is written to D00300 and the PLC memory address of the word containing the minimum value is written to IR00.

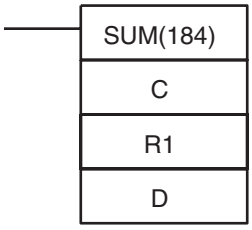


3-17-12 SUM: SUM(184)

Purpose

Adds the bytes or words in the range and outputs the result to two words.

Ladder Symbol



C: First control word

R1: First word in range

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	SUM(184)
	Executed Once for Upward Differentiation	@SUM(184)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

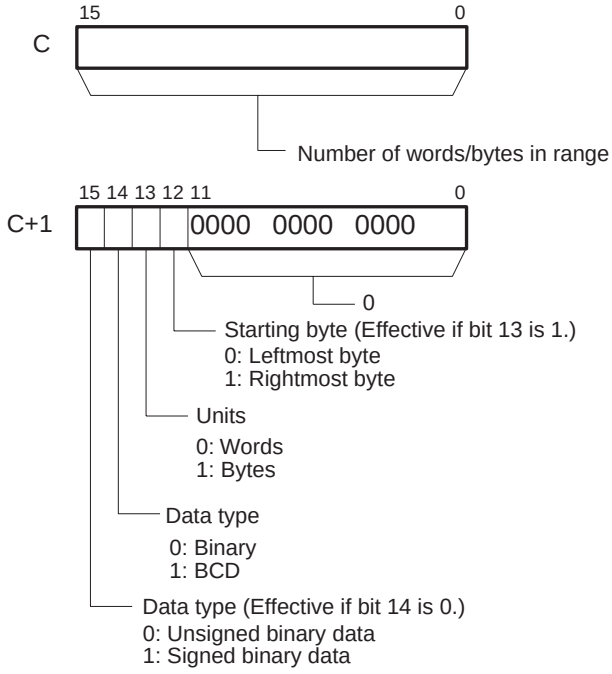
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C and C+1: Control words

C specifies the number of units (bytes or words) to be summed. (Bit 13 of C+1 determines whether bytes or words are being summed.)

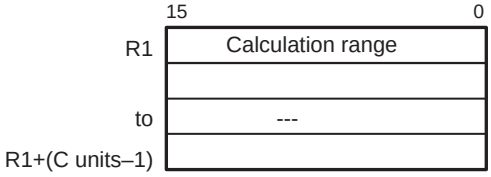
Bits 12 to 15 of C+1 indicate what type of data is being summed, as shown in the following diagram.



Note C and C+1 must be in the same data area.

R1: First word in range

R1 specifies the first word in the range. The length of the range depends on the number of units as well as the starting byte, if bytes are being added.



Note All of the words in the calculation range must be in the same data area.

D: First destination word

The result of the calculation is output to D+1 and D. The leftmost four digits are stored in D+1 and the rightmost four digits are stored in D.

Operand Specifications

Area	C	R1	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142
Work Area	W000 to W510	W000 to W511	W000 to W510
Holding Bit Area	H000 to H510	H000 to H511	H000 to H510
Auxiliary Bit Area	A000 to A958	A000 to A959	A448 to A958
Timer Area	T0000 to T4094	T0000 to T4095	T0000 to T4094
Counter Area	C0000 to C4094	C0000 to C4095	C0000 to C4094
DM Area	D00000 to D32766	D00000 to D32767	D00000 to D32766

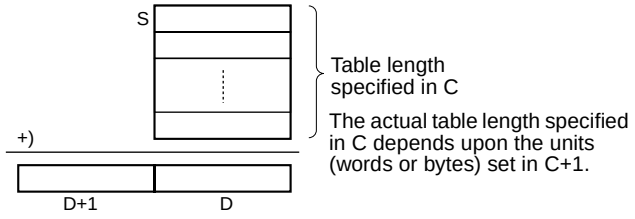
Area	C	R1	D
EM Area without bank	E00000 to E32766	E00000 to E32767	E00000 to E32766
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n= 0 to C)	En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	Specified values only	---	
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SUM(184) adds C units of data beginning with the data in R1 and outputs the result to D+1 and D. The settings in C+1 determine whether the units are words or bytes, whether the data is binary (signed or unsigned) or BCD, and whether to start with the right or left byte of R1 if bytes are being added.

When bit 14 of C+1 has been set to 0, SUM(184) treats the data as binary. In this case, bit 15 determines whether the data is signed (bit 15 = 1) or unsigned (bit 15 = 0).

When bit 13 of C+1 has been set to 1, SUM(184) adds bytes of data. In this case, bit 12 determines whether the calculation starts with the rightmost byte of R1 (bit 12 = 1) or the leftmost byte of R1 (bit 12 = 0).



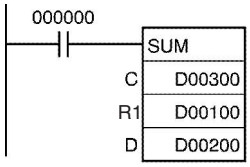
Note SUM(184) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

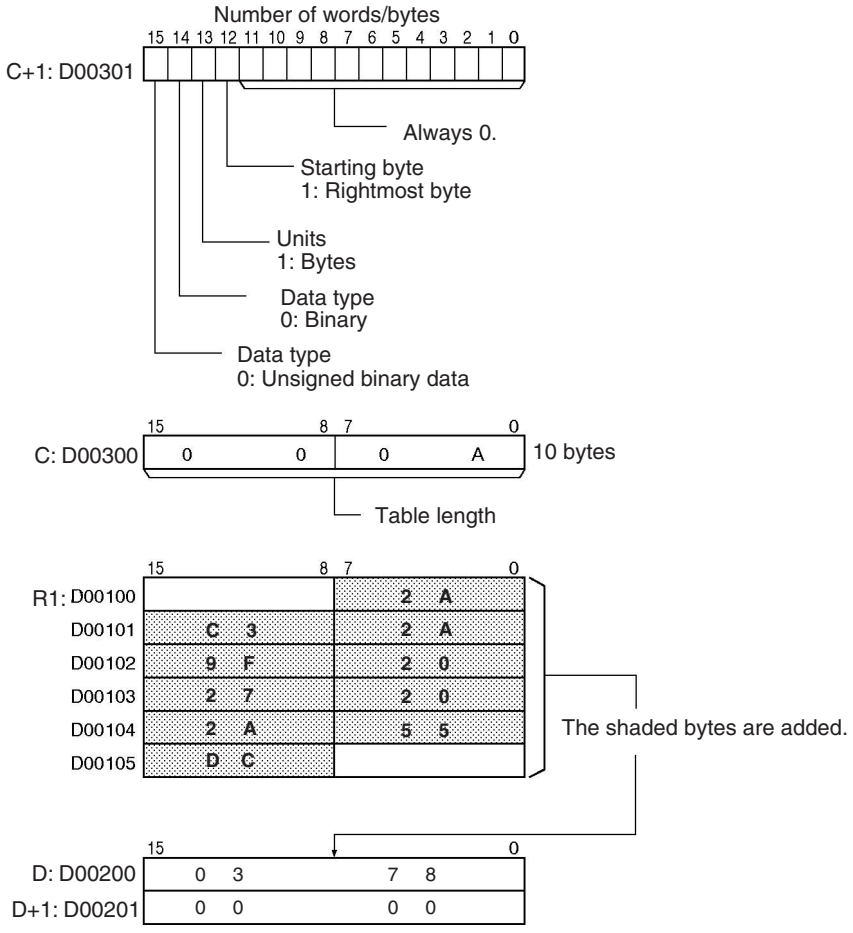
Flags

Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. ON if the BCD data has been specified, but the range contains binary data. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 is ON in the result. OFF in all other cases.

Examples

When CIO 000000 is ON in the following example, SUM(184) adds 10 bytes of unsigned binary data beginning with the rightmost byte of D00100 and writes the result to D00201 and D00200.



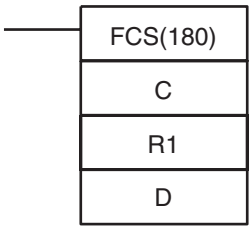


3-17-13 FRAME CHECKSUM: FCS(180)

Purpose

Calculates the FCS value for the specified range and outputs the result in ASCII.

Ladder Symbol



C: First control word

R1: First word in range

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FCS(180)
	Executed Once for Upward Differentiation	@FCS(180)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

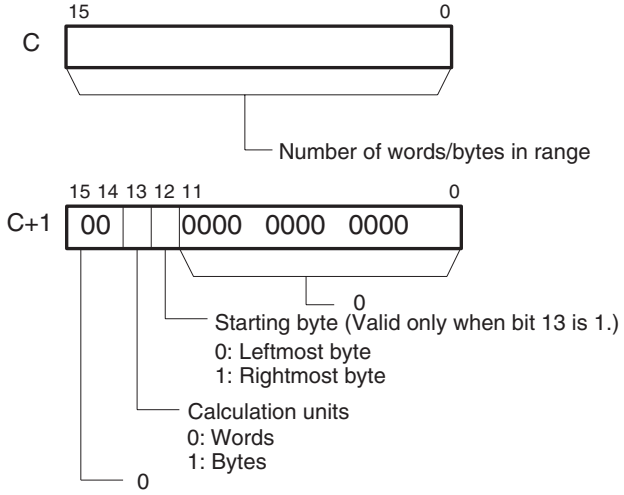
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C and C+1: Control words

C specifies the number of units (bytes or words) to be used in the FCS calculation. (Bit 13 of C+1 determines whether bytes or words are being used.)

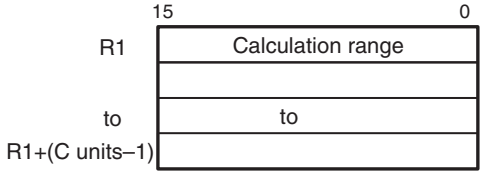
When bit 13 of C+1 has been set to 1, FCS(180) calculates the FCS value for bytes of data. In this case, bit 12 determines whether the calculation starts with the rightmost byte of R1 (bit 12 = 1) or the leftmost byte of R1 (bit 12 = 0).



Note C and C+1 must be in the same data area.

R1: First word in range

R1 specifies the first word in the range. The length of the range depends on the number of units as well as the starting byte, if bytes are being used in the calculation.



Note All of the words in the calculation range must be in the same data area.

D: First destination word

The result of the calculation is output to D if bytes have been selected.

The result of the calculation is output to D+1 and D if words have been selected. In this case, the leftmost four digits are stored in D+1 and the rightmost four digits are stored in D.

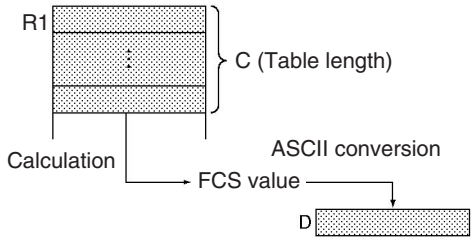
Operand Specifications

Area	C	R1	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143	
Work Area	W000 to W510	W000 to W511	
Holding Bit Area	H000 to H510	H000 to H511	
Auxiliary Bit Area	A000 to A958	A000 to A959	A448 to A959
Timer Area	T0000 to T4094	T0000 to T4095	
Counter Area	C0000 to C4094	C0000 to C4095	
DM Area	D00000 to D32766	D00000 to D32767	
EM Area without bank	E00000 to E32766	E00000 to E32767	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	En_0000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	Specified values only	---	
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

FCS(180) calculates the FCS value for C units of data beginning with the data in R1, converts the value to ASCII code, and outputs the result to D (for bytes) or D+1 and D (for words). The settings in C+1 determine whether the units are words or bytes, whether the data is binary (signed or unsigned) or BCD, and whether to start with the right or left byte of R1 if bytes are being added.

When bit 13 of C+1 has been set to 1, FCS(180) operates on bytes of data. In this case, bit 12 determines whether the calculation starts with the rightmost byte of R1 (bit 12 = 1) or the leftmost byte of R1 (bit 12 = 0).



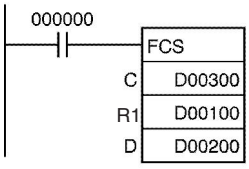
Note FCS(180) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

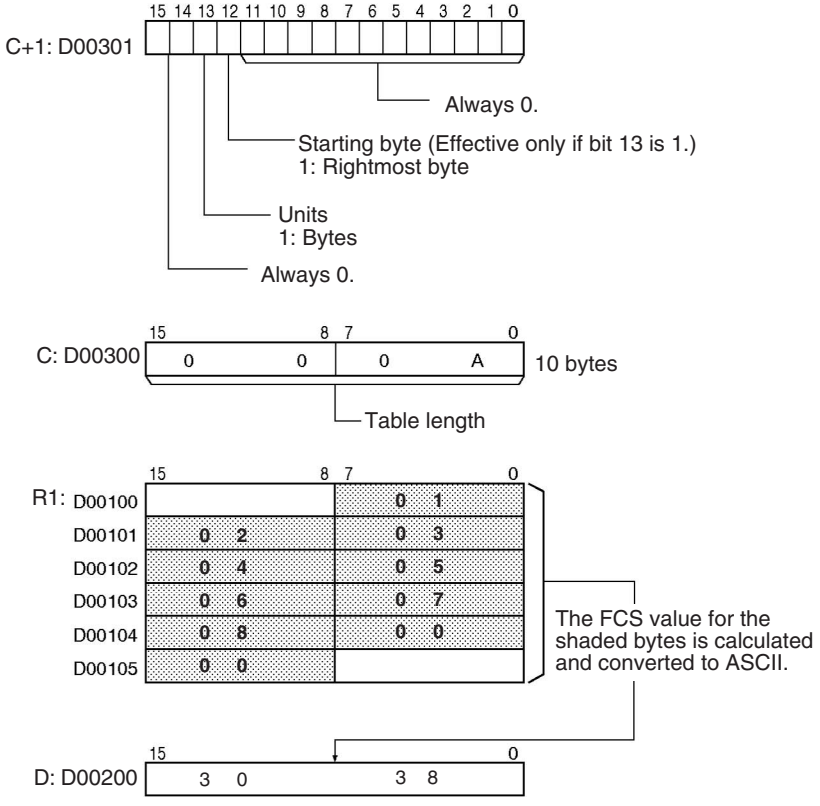
Flags

Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

Examples

When CIO 000000 is ON in the following example, FCS(180) calculates the FCS value for the 10 bytes of data beginning with the rightmost byte of D00100 and writes the result to D00200.





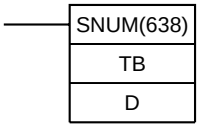
3-17-14 STACK SIZE READ: SNUM(638)

Purpose

Counts the amount of stack data (number of words) in the specified stack.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	SNUM(638)
	Executed Once for Upward Differentiation	@SNUM(638)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

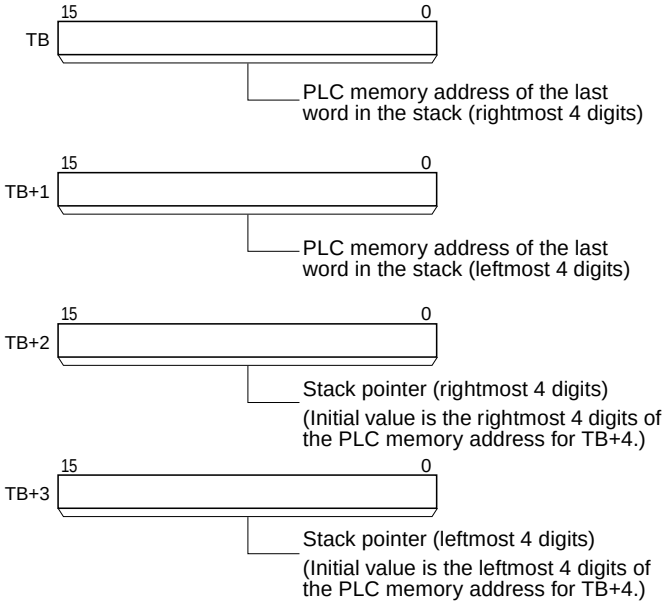
Applicable Program Areas

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

Operands

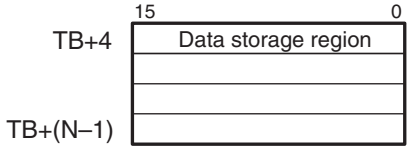
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.

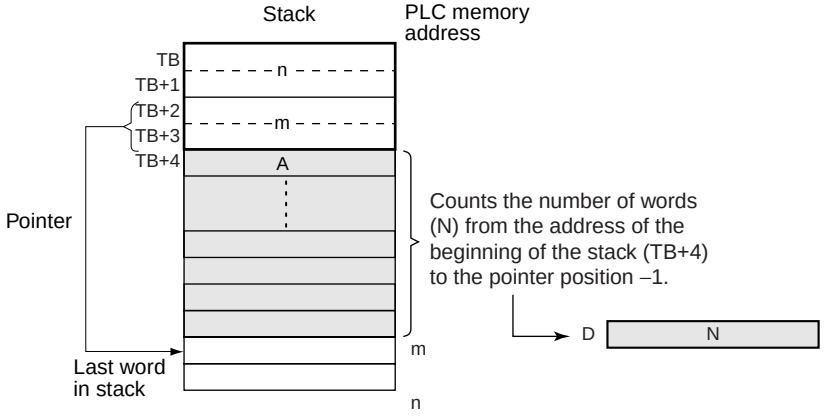


Operand Specifications

Area	TB	D
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

SNUM(638) counts the number of data words in the specified stack from the beginning of the data region at TB+4 to the address before the one indicated by the stack pointer (TB+3 and TB+2). SNUM(638) does not change the data in the stack or the stack pointer.



Flags

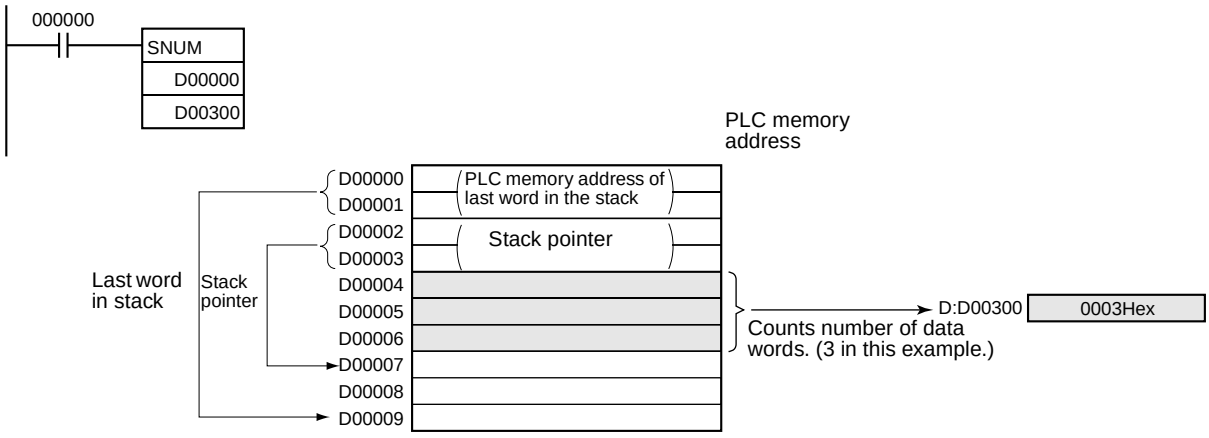
Name	Label	Operation
Error Flag	ER	ON if the number of words of data in the stack (the value output to D) is 0. OFF in all other cases.

Precautions

The stack must be defined in advance with SSET(630).

Examples

When CIO 000000 is ON in the following example, SNUM(638) counts the number of words from the beginning of the data region at D00004 to the stack pointer position - 1 (D00006) and outputs the result to D00300. (In this case, the stack pointer indicates D00007.)



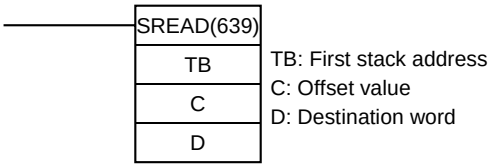
3-17-15 STACK DATA READ: SREAD(639)

Purpose

Reads the data from the specified data element in the stack. The offset value indicates the location of the desired data element (how many data elements before the current pointer position).

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	SREAD(639)
	Executed Once for Upward Differentiation	@SREAD(639)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

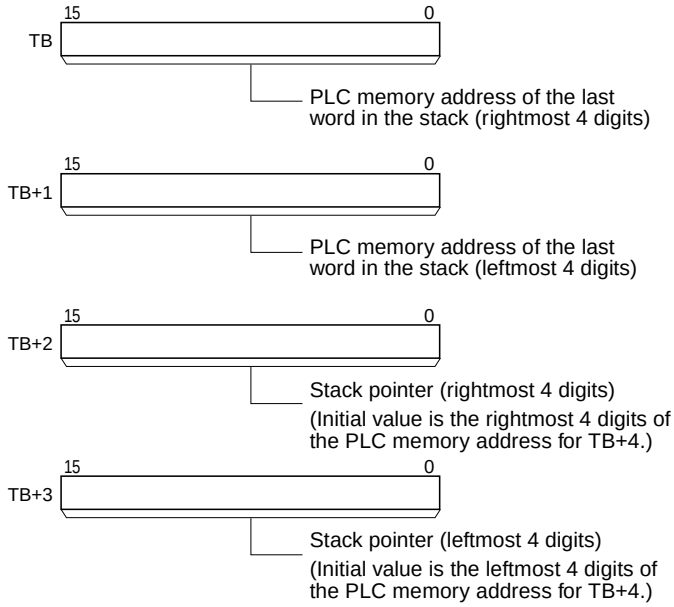
Applicable Program Areas

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

Operands

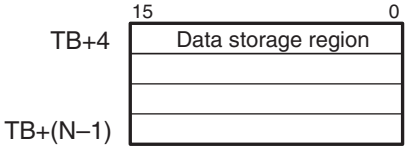
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.



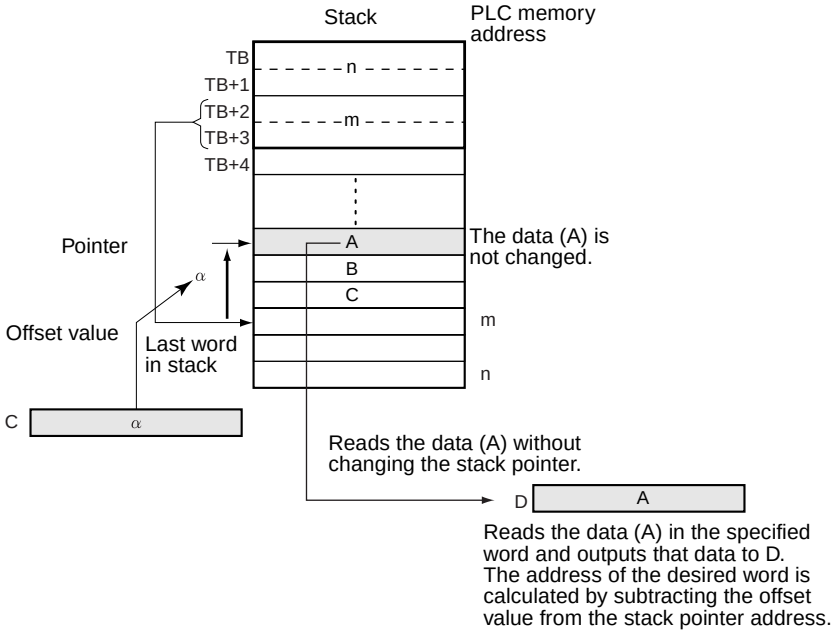
Operand Specifications

Area	TB	C	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A448 to A959	A000 to A959	A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0001 to #FFFB (Hexadecimal)	---
Data Registers	---	DR0 to DR15	

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

Description

SREAD(639) reads the data from the address specified by the stack pointer (TB+3 and TB+2) minus the offset value in C. SREAD(639) does not change the data in the stack or the stack pointer.



SREAD(639) can be used to read the data for an item currently on a conveyor. The position of the desired item is simply the number of items back (the offset value) from the most recent item added to the conveyor.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified read location is not within the stack area. ON if the offset value specified in C is 0 or greater than the maximum data region size (FFFB hex). OFF in all other cases.
Equals Flag	=	ON if the output data in D is 0000. OFF in all other cases.

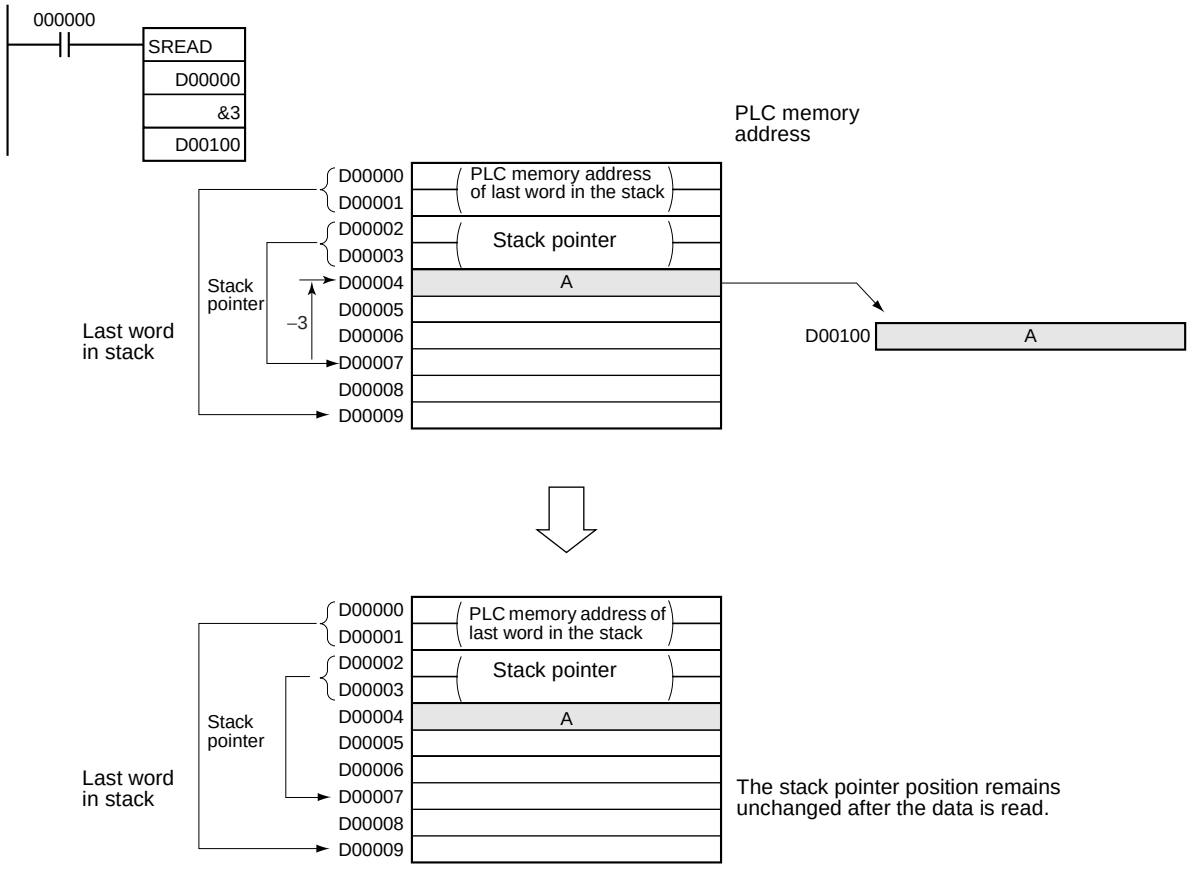
Precautions

The stack must be defined in advance with SSET(630).

The address in the stack pointer must be greater than the PLC memory address of the beginning of the data region (TB+4). An error will occur if the stack pointer is less than the PLC memory address of TB+4, i.e., if a stack underflow error occurs.

Examples

When CIO 000000 is ON in the following example, SREAD(639) reads the data in the specified word in the stack starting at D00000 and outputs the data to D00100. In this case, the stack pointer indicates D00007 and the offset value is 3, so the data is read from D00004.



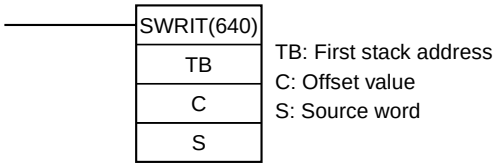
3-17-16 STACK DATA OVERWRITE: SWRIT(640)

Purpose

Writes the source data to the specified data element in the stack (overwriting the existing data). The offset value indicates the location of the desired data element (how many data elements before the current pointer position).

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	SWRIT(640)
	Executed Once for Upward Differentiation	@SWRIT(640)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

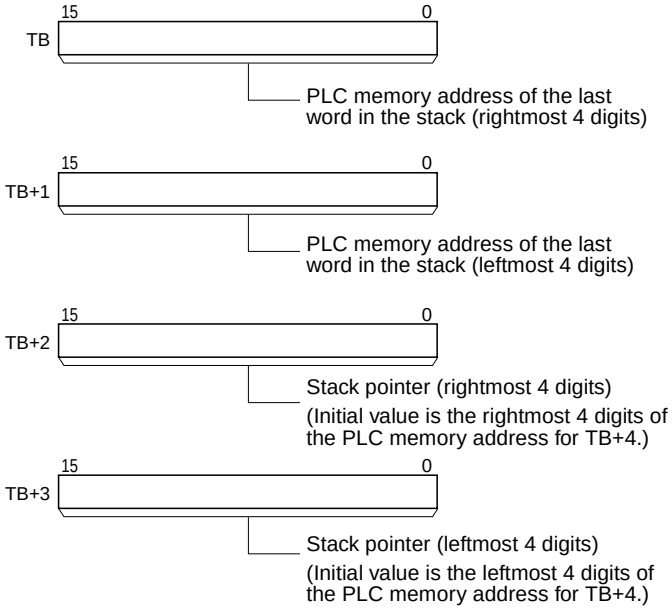
Applicable Program Areas

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

Operands

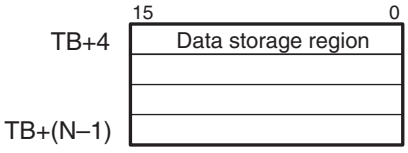
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.



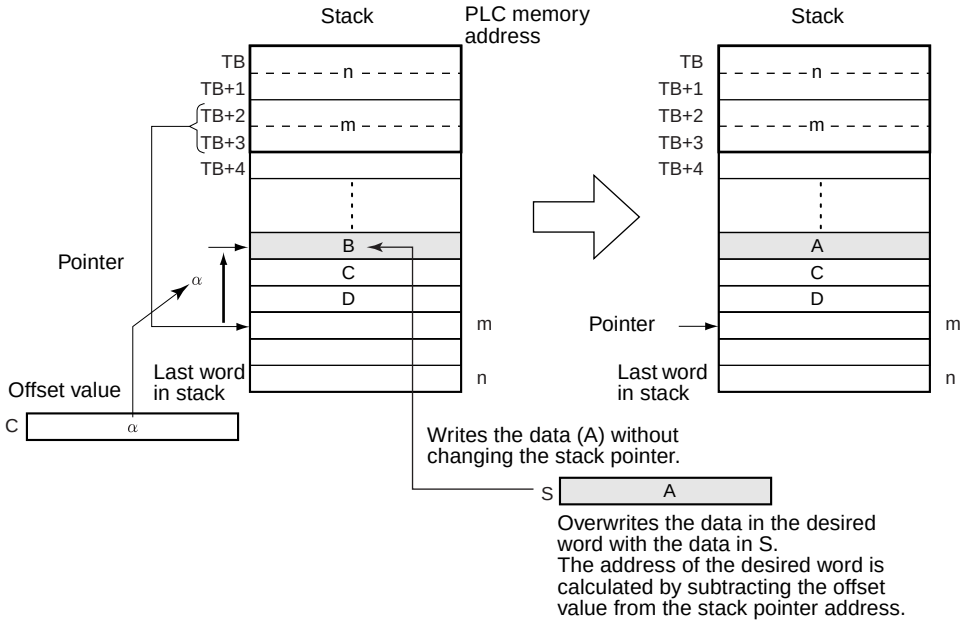
Operand Specifications

Area	TB	C	S
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A448 to A959	A000 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		

Area	TB	C	S
Constants	---	#0001 to #FFFB (Hexadecimal)	#0000 to #FFFF (Hexadecimal)
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SWRIT(640) overwrites the data in the desired word with the data specified in S. The location of the desired word is calculated by subtracting the offset value in C from the stack pointer (TB+3 and TB+2). SWRIT(640) does not change the stack pointer.



SWRIT(640) can be used to change the data for an item currently on a conveyor. The position of the desired item is simply the number of items back (the offset value) from the most recent item added to the conveyor.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified write location is not within the stack area. ON if the offset value specified in C is 0 or greater than the maximum data region size (FFFB hex). OFF in all other cases.

Precautions

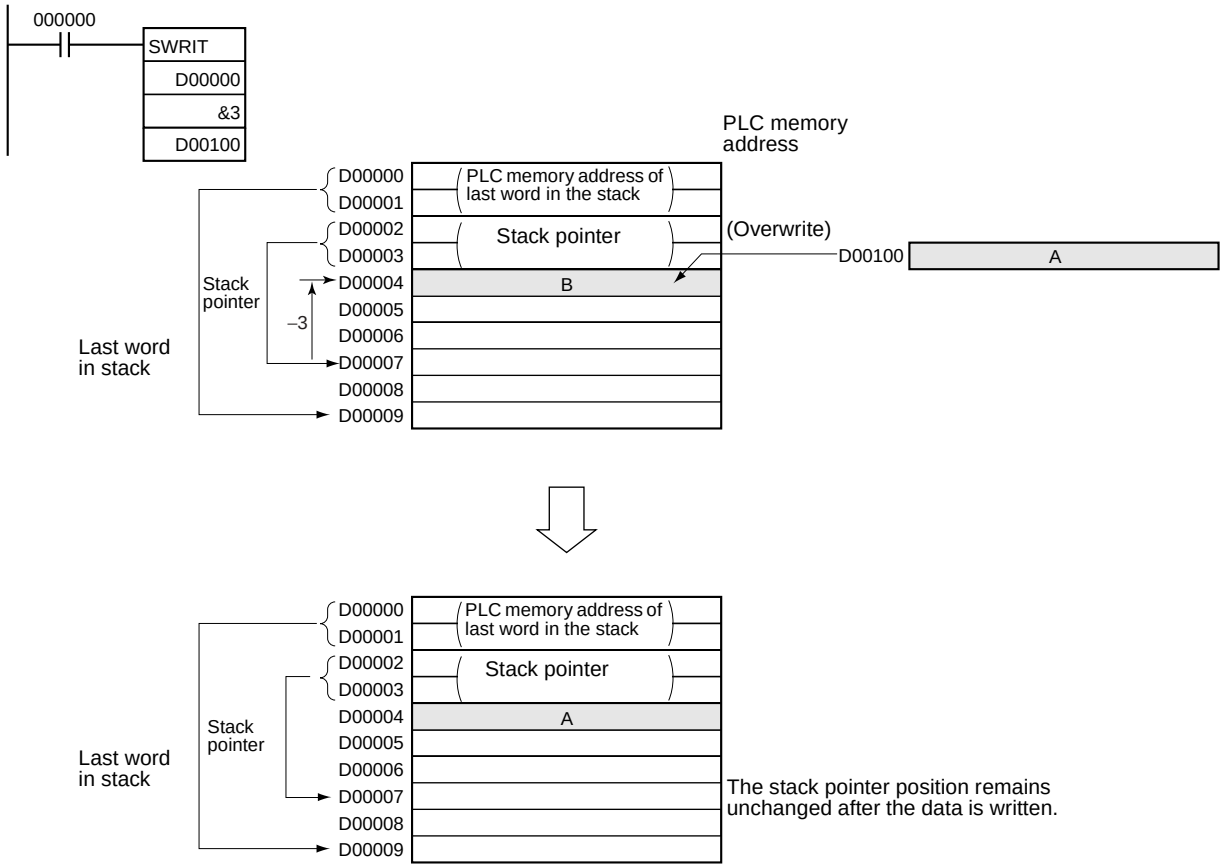
The stack must be defined in advance with SSET(630).

The address in the stack pointer must be greater than the PLC memory address of the beginning of the data region (TB+4). An error will occur if the stack pointer is less than the PLC memory address of TB+4, i.e., if a stack underflow error occurs.

Examples

When CIO 000000 is ON in the following example, SWRIT(640) writes the data in D00100 to the specified word in the stack starting at D00000. In this

case, the stack pointer indicates D00007 and the offset value is 3, so the data in D00004 is overwritten.



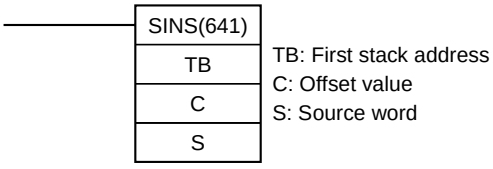
3-17-17 STACK DATA INSERT: SINS(641)

Purpose

Inserts the source data at the specified location in the stack and shifts the rest of the data in the stack downward. The offset value indicates the location of the desired data element (how many data elements before the current pointer position).

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	SINS(641)
	Executed Once for Upward Differentiation	@SINS(641)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

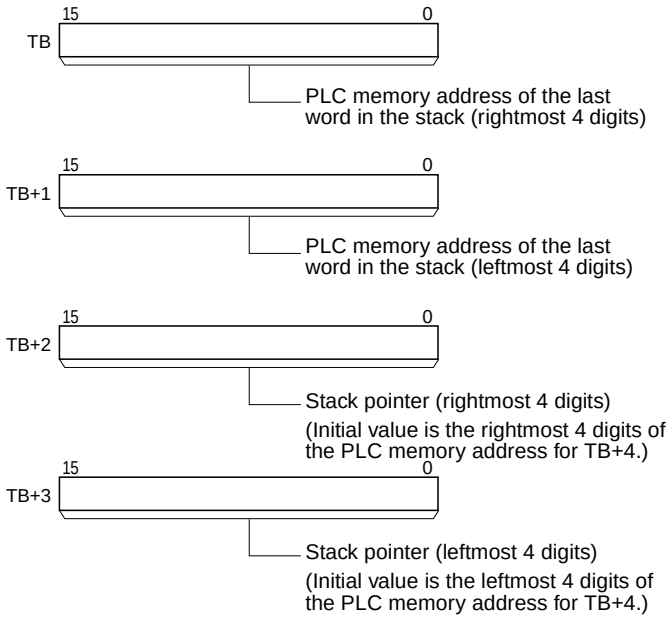
Applicable Program Areas

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

Operands

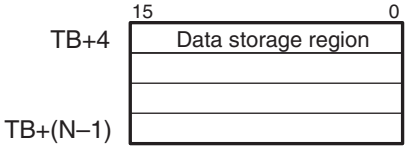
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.



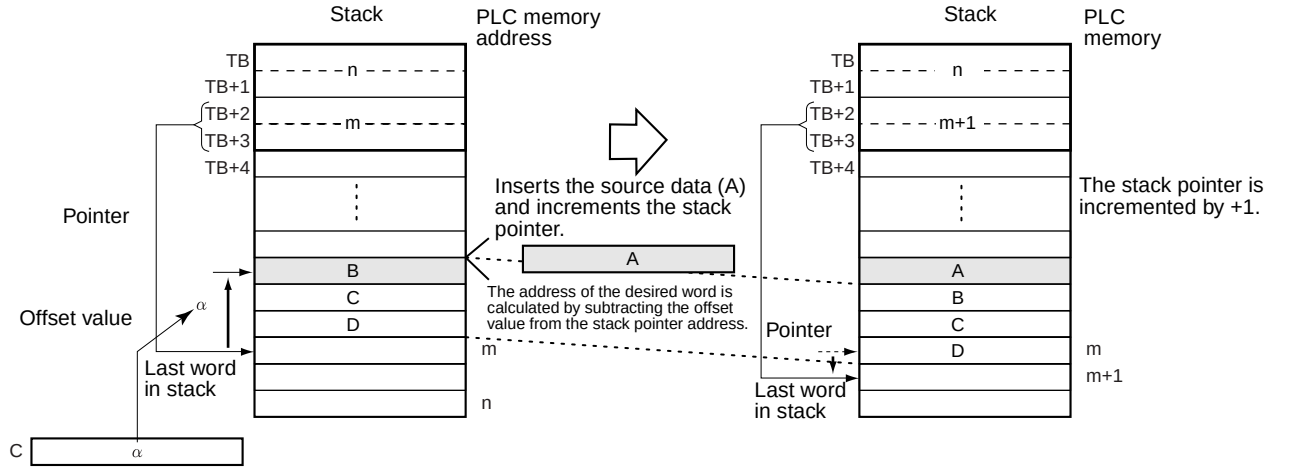
Operand Specifications

Area	TB	C	S
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A448 to A959	A000 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		

Area	TB	C	S
Constants	---	#0001 to #FFFB (Hexadecimal)	#0000 to #FFFF (Hexadecimal)
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SINS(641) inserts the source data at the desired address and shifts the existing data down one word. At the same time, SINS(641) increments the stack pointer (TB+3 and TB+2) by 1. The location of the desired address is calculated by subtracting the offset value in C from the stack pointer.



SINS(641) can be used to insert the data for an item that is inserted in the midst of items already on a conveyor. The position of the insertion point is simply the number of items back (the offset value) from the most recent item added to the conveyor.

Flags

Name	Label	Operation
Error Flag	ER	ON if the address indicated by the stack pointer (TB+3 and TB+2) is greater than the PLC memory address of last word in the data region of the stack. (This is a stack overflow error.) ON if the offset value specified is greater than the maximum data region size - 1 (FFFA hex). OFF in all other cases.

Precautions

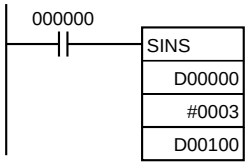
The stack must be defined in advance with SSET(630).

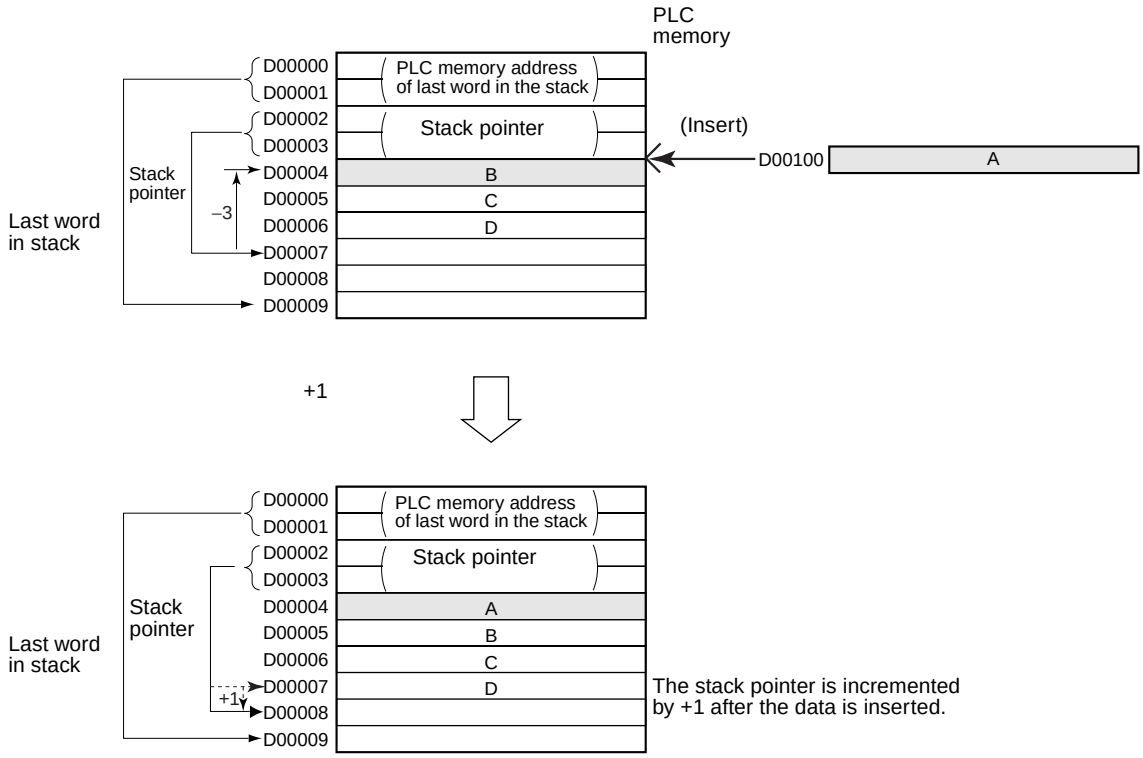
SINS(641) inserts one word of data into the stack, so there must be at least one available word at the end of the stack. If the stack is full, an error will occur and the source data will not be inserted.

If the address indicated by the stack pointer (TB+3 and TB+2) is already greater than the address of the last word in the stack (TB+1 and TB) when SINS(641) is executed, a stack overflow error will occur and the source data will not be inserted.

Examples

When CIO 000000 is ON in the following example, SINS(641) inserts the source data in D00100 at the specified address in the stack starting at D00000. In this case, the stack pointer indicates D00007 and the offset value is 3, so the source data is inserted in D00004. The existing data is shifted down one word and the data in D00007 is overwritten. At the same time the stack pointer will be incremented from D00007 to D00008.





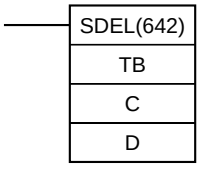
3-17-18 STACK DATA DELETE: SDEL(642)

Purpose

Deletes the data element at the specified location in the stack, outputs that data to the specified destination word, and shifts the remaining the data in the stack upward. The offset value indicates the location of the desired data element (how many data elements before the current pointer position).

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



TB: First stack address
C: Offset value
D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	SDEL(642)
	Executed Once for Upward Differentiation	@SDEL(642)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

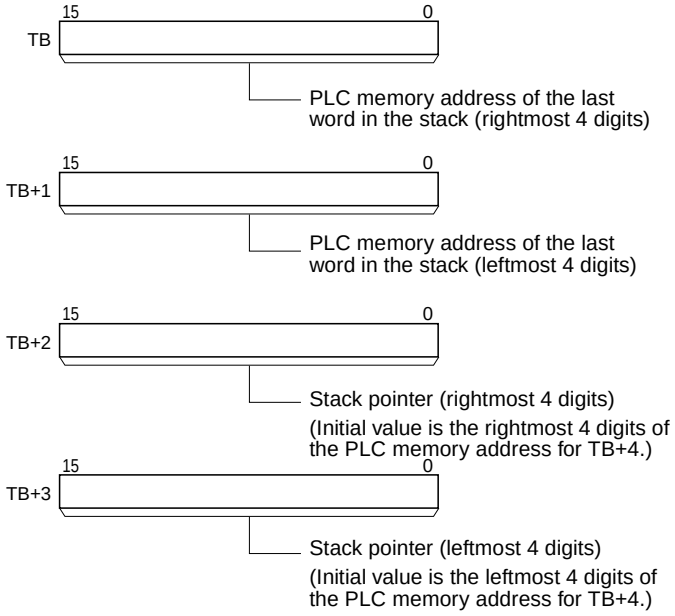
Applicable Program Areas

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

Operands

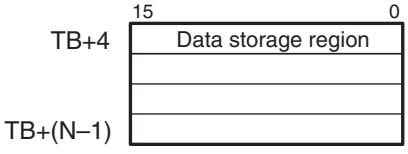
TB through TB+3: Stack control words

The first four words of the stack contain the PLC memory address of the last word in the stack and the stack pointer (the PLC memory address of the next available word in the stack.)



TB+4 through TB+(N–1): Data storage region

The remainder of the stack is used to store data.



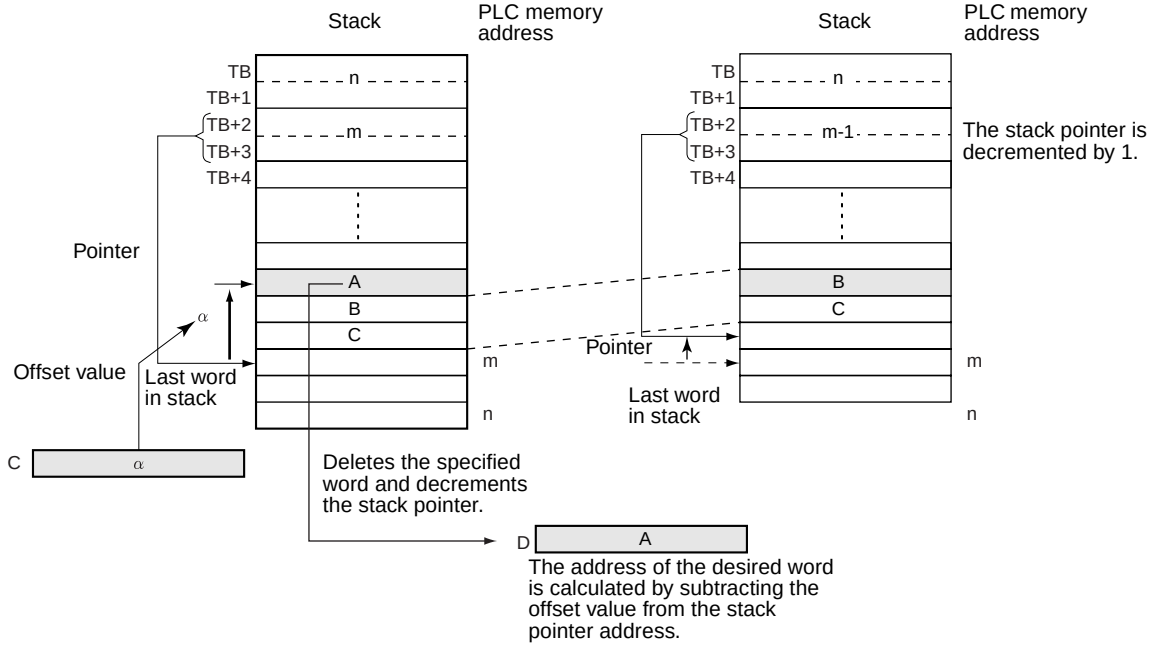
Operand Specifications

Area	TB	C	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A448 to A959	A000 to A959	A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		

Area	TB	C	D
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0001 to #FFFB (Hexadecimal)	---
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SDEL(642) deletes the data at the specified location in the stack, outputs that data to the specified destination word, and shifts the remaining the data in the stack upward. At the same time, SDEL(642) decrements the stack pointer (TB+3 and TB+2) by 1. The location of the desired address is calculated by subtracting the offset value in C from the stack pointer.



SDEL(642) can be used to delete the data for an item that is rejected from the items on a conveyor. The position of the deletion point is simply the number of items back (the offset value) from the most recent item added to the conveyor.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of the stack pointer (TB+3 and TB+2) is less than or equal to the PLC memory address of first word in the data region of the stack (TB+4). (This is a stack underflow error.) ON if the offset value specified in C is 0 or greater than the maximum data region size (FFFB hex). OFF in all other cases.
Equals Flag	=	ON if the output data in D is 0000. OFF in all other cases.

Precautions

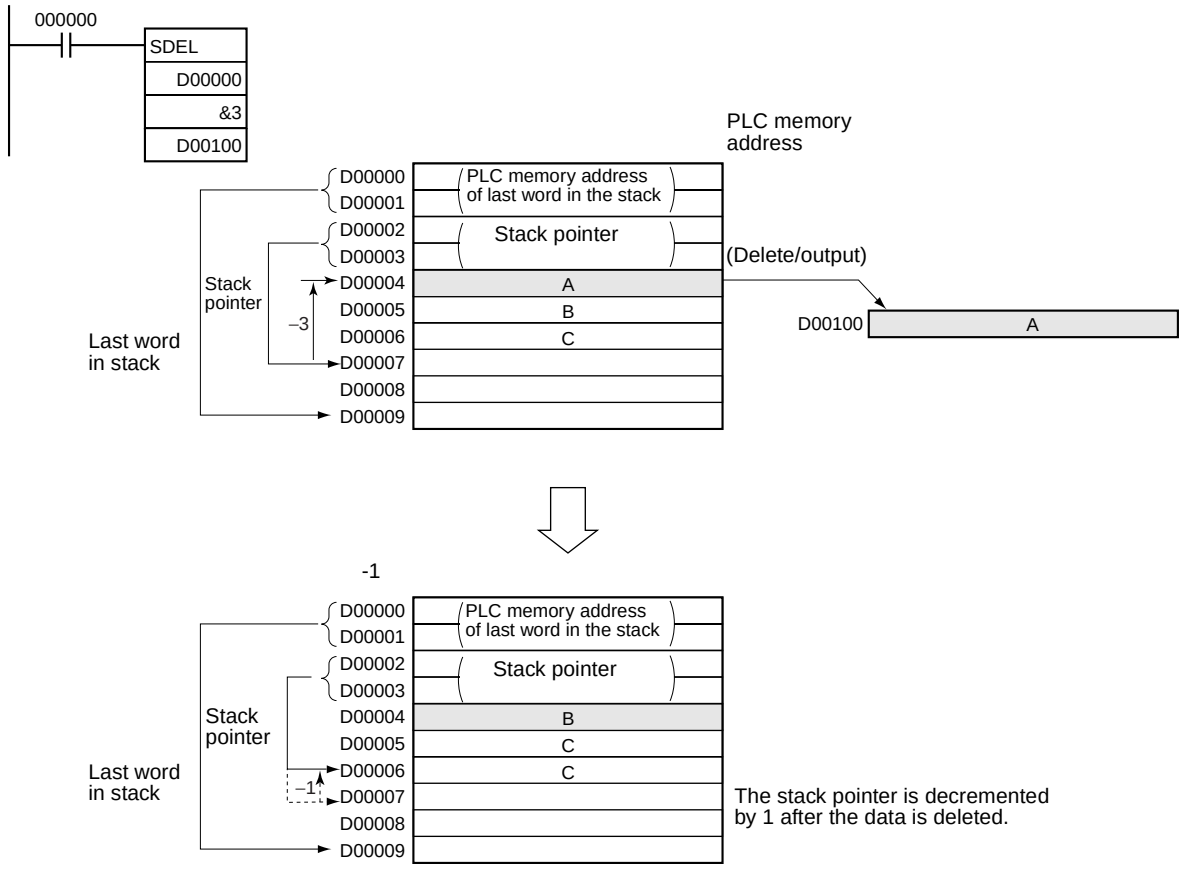
The stack must be defined in advance with SSET(630).

The address in the stack pointer must be greater than the PLC memory address of the beginning of the data region (TB+4). An error will occur if the stack pointer is less than the PLC memory address of TB+4, i.e., if a stack underflow error occurs.

Examples

When CIO 000000 is ON in the following example, SDEL(642) deletes the word at the specified address in the stack starting at D00000, outputs the deleted data to D00100, shifts the remaining data upward, and decrements the stack pointer.

In this case, the stack pointer indicates D00007 and the offset value is 3, so the data is deleted from D00004. The remaining data is shifted up one word and the stack pointer is decremented from D00007 to D00006.



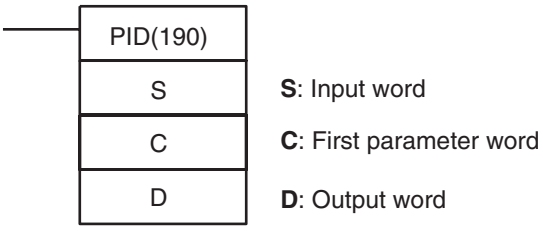
3-18 Data Control Instructions

3-18-1 PID CONTROL: PID(190)

Purpose

Executes PID control according to the specified parameters.

Ladder Symbol



Variations

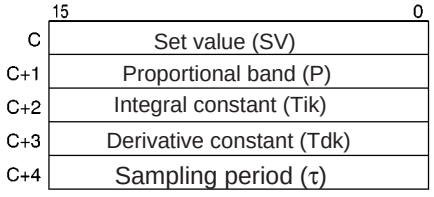
Variations		
	Executed Each Cycle for ON Condition	PID(190)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

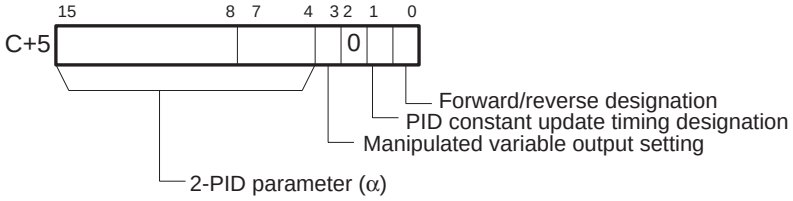
Applicable Program Areas

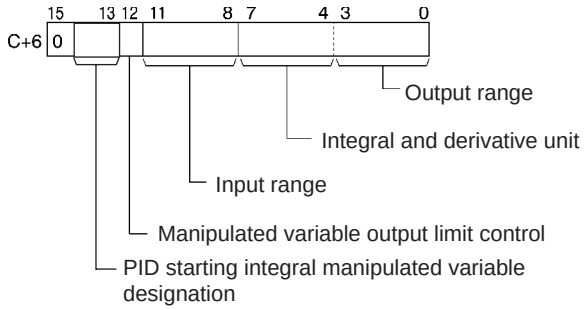
Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	Not allowed

Parameters

The following diagrams show the locations of the parameter data. For details on the parameters, refer to *PID Parameter Settings* in this section.







Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6105	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W473	W000 to W511
Holding Bit Area	H000 to H511	H000 to H473	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A921	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4057	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4057	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32729	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32729	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32729 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	DR0 to DR15	---	DR0 to DR15
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15		

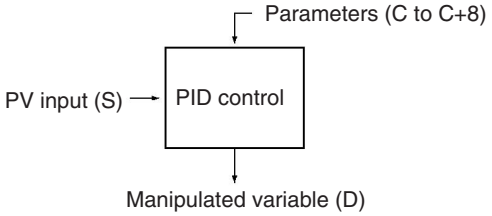
Description

When the execution condition is ON, PID(190) carries out target value filtered PID control with two degrees of freedom according to the parameters designated by C (set value, PID constant, etc.). It takes the specified input range of binary data from the contents of input word S and carries out the PID action according to the parameters that are set. The result is then stored as the manipulated variable in output word D.

The parameters are obtained when the execution condition turns from OFF to ON, and the Error Flag will turn ON if the settings are outside of the permissible range.

If the settings are within the permissible range, PID processing will be executed using the initial values. Bumpless operation is not performed at this time. It will be used for manipulated variables in subsequent PID processing execution. (Bumpless operation is processing that gradually and continuously changes the manipulated variable in order to avoid the adverse effects of sudden changes.)

When the execution condition turns ON, the PV for the specified sampling period is entered and processing is performed.



The number of valid input data bits within the 16 bits of the PV input (S) is designated by the input range setting in C+6, bits 08 to 11. For example, if 12 bits (4 hex) is designated for the input range, the range from 0000 hex to 0FFF hex will be enabled as the PV. (Values greater than 0FFF hex will be regarded as 0FFF hex.)

The set value range also depends on the input range.

Measured values (PV) and set values (SV) are in binary without sign, from 0000 hex to the maximum value of the input range.

The number of valid output data bits within the 16 bits of the manipulated variable output is designated by the output range setting in C+6, bits 00 to 03. For example, if 12 bits (4 hex) is designated for the output range, the range from 0000 hex to 0FFF hex will be output as the manipulated variable.

For proportional operation only, the manipulated variable output when the PV equals the SV can be designated as follows:

0: Output 0%
1: Output 50%.

The direction of proportional operation can be designated as either forward or reverse.

The upper and lower limits of the manipulated variable output can be designated.

The sampling period can be designated in units of 10 ms (0.01 to 99.99 s), but the actual PID action is determined by a combination of the sampling period and the time of PID(190) instruction execution (with each cycle).

The timing of enabling changes made to PID constants can be set to either 1) the beginning of PID instruction execution or 2) the beginning of PID instruction execution and each sampling period. Only the proportional band (P), integral constant (Tik), and derivative constant (Tdk) can be changed each sampling cycle (i.e., during PID instruction execution). The timing is set in bit 1 of C+5.

Note The setting in bit 1 of C+5 is supported only by CJ1, CS1-H, CJ1-H CPU Units and CS1 CPU Units with lot numbers of 001201□□□□ or later (manufactured December 1, 2000 or later).

Of the PID parameters (C to C+38), only the set value (SV) can be changed when the execution condition is ON. When changing other values, be sure to change the execution condition from OFF to ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if the C data is out of range. ON if the actual sampling period is more than twice the designated sampling period. OFF in all other cases.
Greater Than Flag	>	ON if the manipulated variable after the PID action exceeds the upper limit. OFF in all other cases.
Less Than Flag	<	ON if the manipulated variable after the PID action is below the lower limit. OFF in all other cases.
Carry Flag	CY	ON while PID control is being executed. OFF in all other cases.

Precautions

The same words cannot be used to store the PID parameters for more than one PID(190) instruction. Even if the same parameters are used, use different words to store the PID parameters for different PID(190) instructions.

PID(190) is executed as if the execution condition was a STOP-RUN signal. PID calculations are executed when the execution condition remains ON for the next cycle after C+9 to C+38 are initialized. Therefore, when using the Always ON Flag (ON) as an execution condition for PID(190), provide a separate process where C+9 to C+38 are initialized when operation is started.

If the C data is out of range, an error will occur and the Error Flag will turn ON.

If the actual sampling period is more than twice the designated sampling period, an error will occur and the Error Flag will turn ON. PID control will still be executed, however.

The Carry Flag turns ON while PID control is being executed.

The Greater Than Flag turns ON if the manipulated variable after the PID action exceeds the upper limit. At this time, the results are output at the upper limit.

The Less Than Flag turns ON if the manipulated variable after the PID action is below the lower limit. At this time, the results are output at the lower limit.

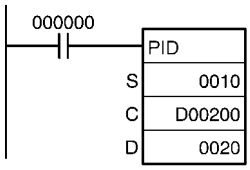
Within the PID parameters (C to C+38), the only value that can be changed while the input condition is ON is the set value for C. If any other value is changed, be sure to turn the input condition from OFF to ON to enable the new value.

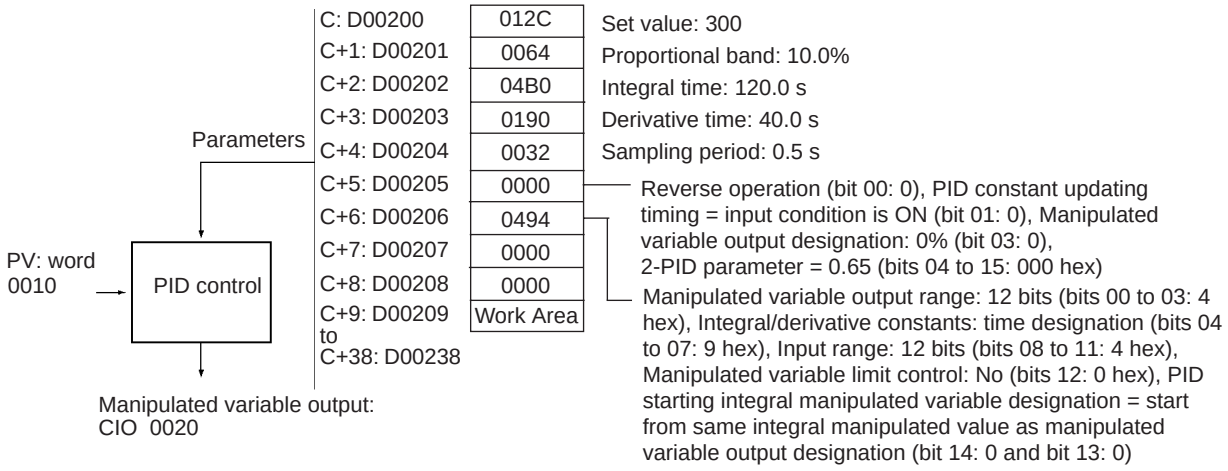
Example

At the rising edge of CIO 000000 (OFF to ON), the work area in D00209 to D00238 is initialized according to the parameters (shown below) set in D00200 to D00208. After the work area has been initialized, PID control is executed and the manipulated variable is output to CIO 0020.

When CIO 000000 is turned ON, PID control is executed at the sampling period intervals according to the parameters set in D00200 to D00208. The manipulated variable is output to CIO 0020.

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 CIP 000000 turns ON.





Note When CIO 000000 is OFF, operation can be the same as manual operation by writing to CIO 0020. 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.)

Input Values and Manipulated Variable Ranges

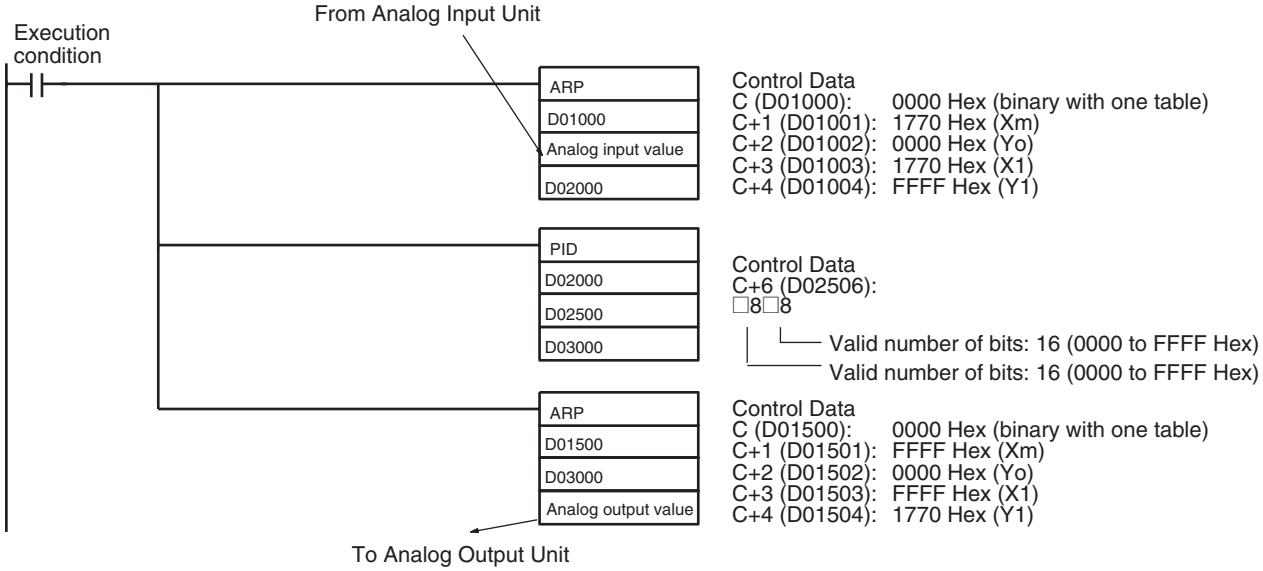
The number of valid input data bits for the measured value is designated by the input range setting in C+6, bits 08 to 11, and the number of valid output data bits for the manipulated variable output is designated by the output range setting in C+6, bits 0 to 3. These ranges are shown in the following table.

C+6, bits 08 to 11 or C+6, bits 00 to 03	Number of valid bits	Range
0	8	0000 to 00FF hex
1	9	0000 to 01FF hex
2	10	0000 to 03FF hex
3	11	0000 to 07FF hex
4	12	0000 to 0FFF hex
5	13	0000 to 1FFF hex
6	14	0000 to 3FFF hex
7	15	0000 to 7FFF hex
8	16	0000 to FFFF hex

If the range of data handled by an Analog Input Unit or Analog Output Unit cannot be set accurately by setting the number of valid bits, APR(069) (ARITHMETIC PROCESS) can be used to convert to the proper ranges before and after PID(190).

The following program section shows an example for a DRT1-AD04 Analog Input Unit and DRT1-DA02 Analog Output Unit operating as DeviceNet slaves. The data ranges for these two Units is 0000 to 1770 hex, which cannot be specified merely by setting the valid number of digits. APR(069) is thus used to convert the 0000 to 1770 hex range of the Analog Input Unit to 0000

to FFFF hex for input to PID(190) and then the manipulated variable output from PID(190) is converted back to the range 0000 to 1770 hex, again using APR(069), for output from the Analog Output Unit.



Performance Specifications

Item			Specifications
PID control method		---	Target value filter-type two-degrees-of-freedom PID method (forward/reverse)
Number of PID control loops		---	Unlimited (1 loop per instruction)
Sampling period		τ	0.01 to 99.99 s
PID constant	Proportional band	P	0.1 to 999.9%
	Integral constant	Tik	1 to 8191, 9999 (No integral action for sampling period multiple, 9999.)
	Derivative constant	Tdk	0 to 8191 (No derivative action for sampling period multiple, 0.)
Set value		SV	0 to 65535 (Valid up to maximum value of input range.)
Measured value		PV	0 to 65535 (Valid up to maximum value of input range.)
Manipulated variable		MV	0 to 65535 (Valid up to maximum value of output range.)

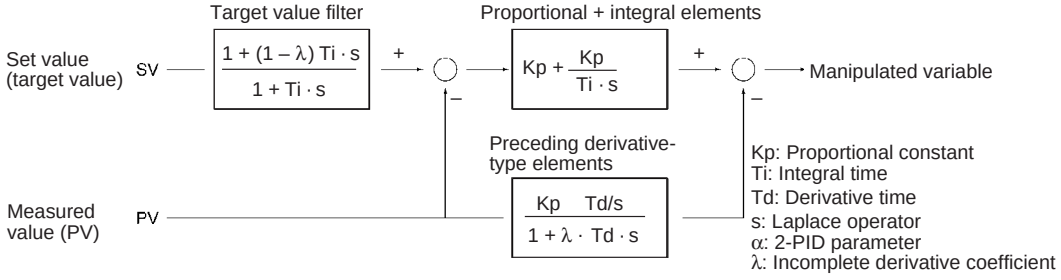
Calculation Method

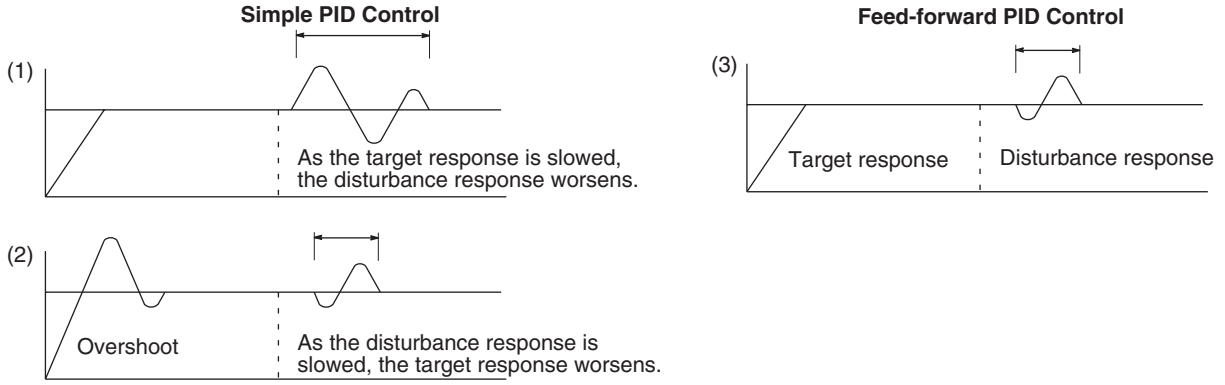
Calculations in PID control are performed by the target value filtered control with two degrees of freedom.

Block Diagram for Target Value PID with Two Degrees of Freedom

When overshooting is prevented with simple PID control, stabilization of disturbances is slowed (1). If stabilization of disturbances is speeded up, on the other hand, overshooting occurs and response toward the target value is slowed (2).

When target-value PID control with two degrees of freedom is used, on the other hand, there is no overshooting, and response toward the target value and stabilization of disturbances can both be speeded up (3).





PID Parameter Settings

Control data	Item	Contents	Setting range	Change with ON input condition
C	Set value (SV)	The target value of the process being controlled.	Binary data (of the same number of bits as specified for the input range)	Allowed
C+1	Proportional band	The parameter for P action expressing the proportional control range/total control range.	0001 to 270F hex (1 to 9999); (0.1% to 999.9%, in units of 0.1%)	Can be changed with input condition ON if bit 1 of C+5 is 1.
C+2	Tik Integral Constant	A constant expressing the strength of the integral action. As this value increases, the integral strength decreases.	0001 to 1FFF hex (1 to 8191); (9999 = Integral operation not executed) (See note 1.)	
C+3	Tdk Derivative Constant	A constant expressing the strength of the derivative action. As this value increases, the derivative strength decreases.	0001 to 1FFF hex (1 to 8191); (0000 = Derivative operation not executed) (See note 1.)	
C+4	Sampling period (τ)	Sets the period for executing the PID action.	0001 to 270F hex (1 to 9999); (0.01 to 99.99 s, in units of 10 ms)	Not allowed
Bits 04 to 15 of C+5	2-PID parameter (α)	The input filter coefficient. Normally use 0.65 (i.e., a setting of 000). The filter efficiency decreases as the coefficient approaches 0.	000 hex: α = 0.65 Setting from 100 to 163 hex means that the value of the rightmost two digits is set from α= 0.00 to α= 0.99. (See note 2.)	
Bit 03 of C+5	Manipulated variable output designation	Designates the manipulated variable output when the PV equals the SV. **Note** 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

Control data	Item	Contents	Setting range	Change with ON input condition
Bit 00 of C+5	PID forward/reverse designation	Determines the direction of the proportional action.	0: Reverse action 1: Forward action	Not allowed
Bits 13 to 14 of C+6	ID starting integral manipulated variable designation (unit version 4.0 or later only)	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 5: 13 bits 1: 9 bits 6: 14 bits 2: 10 bits 7: 15 bits 3: 11 bits 8: 16 bits 4: 12 bits	
Bits 04 to 07 of C+6	Integral and derivative unit	Determines the unit for expressing the integral and derivative constants.	1: Sampling period multiple 9: Time (unit: 100 ms)	
Bits 00 to 03 of C+6	Output range	The number of output data bits.	0: 8 bits 5: 13 bits 1: 9 bits 6: 14 bits 2: 10 bits 7: 15 bits 3: 11 bits 8: 16 bits 4: 12 bits	
C+7	Manipulated variable output lower limit	The lower limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)	
C+8	Manipulated variable output upper limit	The upper limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)	

Note

1. When the unit is designated as 1, the range is from 1 to 8,191 times the period. When the unit is designated as 9, the range is from 0.1 to 819.1 s. When 9 is designated, set the integral and derivative times to within a range of 1 to 8,191 times the sampling period.
2. Setting the 2-PID parameter (α) to 000 yields 0.65, the normal value.
3. When the manipulated variable output limit control is enabled (i.e., set to "1"), set the values as follows:

   0000 ≤ MV output lower limit ≤ MV output upper limit ≤ Max. value of output range

Sampling Period and Cycle Time

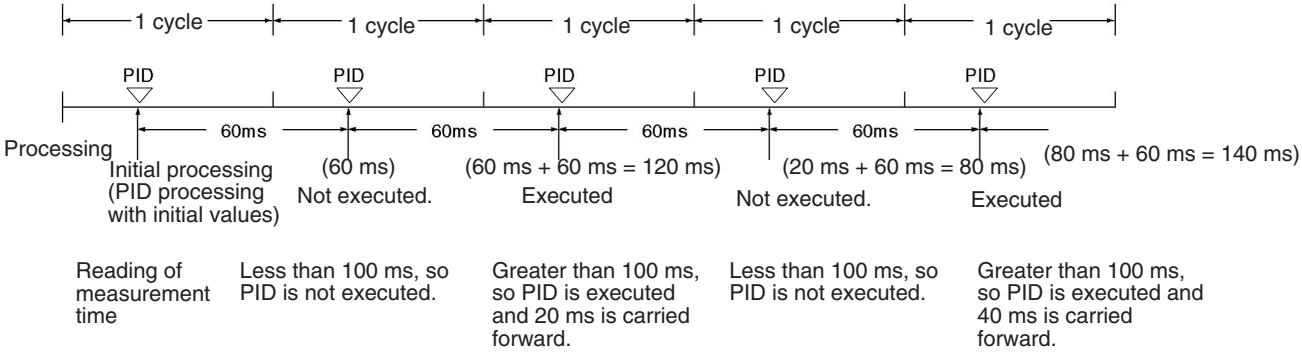
The sampling period can be designated in units of 10 ms (0.01 to 99.99 s), but the actual PID action is determined by a combination of the sampling period and the time of PID instruction execution (with each cycle). The relationship between the sampling period and the cycle time is as follows:

- If the sampling period is less than the cycle time, PID control is executed with each cycle and not with each sampling period.

- If the sampling period is greater than or equal to the cycle time, PID control is not executed with each cycle, but PID(190) is executed when the cumulative value of the cycle time (the time between PID instructions) is greater than or equal to the sampling period. The surplus portion of the cumulative value (i.e., the cycle time's cumulative value minus the sampling period) is carried forward to the next cumulative value.

  For example, suppose that the sampling period is 100 ms and that the cycle time is consistently 60 ms. For the first cycle after the initial execution, PID(190) will not be executed because 60 ms is less than 100 ms. For the second cycle, 60 ms + 60 ms is greater than 100 ms, so PID(190) will be executed. The surplus of 20 ms (i.e., 120 ms – 100 ms = 20 ms) will be carried forward.

  For the third cycle, the surplus 20 ms is added to 60 ms. Because the sum of 80 ms is less than 100 ms, PID(190) will not be executed. For the fourth cycle, the 80 ms is added to 60 ms. Because the sum of 140 ms is greater than 100 ms, PID(190) will be executed and the surplus of 40 ms (i.e., 120 ms – 100 ms = 20 ms) will be carried forward. This procedure is repeated for subsequent cycles.



Control Actions

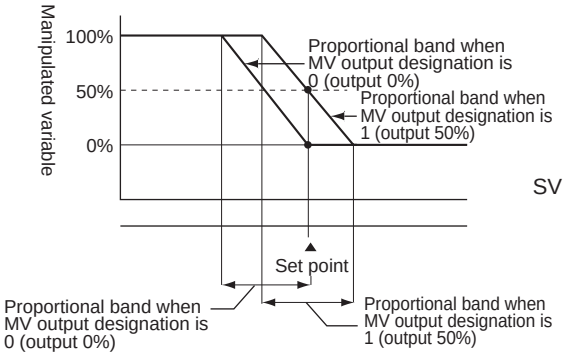
Proportional Action (P)

Proportional action is an operation in which a proportional band is established with respect to the set value (SV), and within that band the manipulated variable (MV) is made proportional to the deviation. An example for reverse operation is shown in the following illustration.

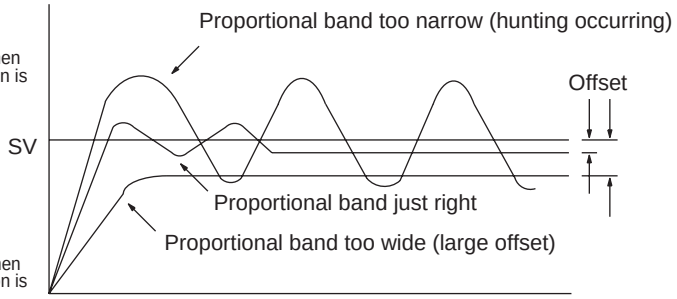
If the proportional action is used and the present value (PV) becomes smaller than the proportional band, the manipulated variable (MV) is 100% (i.e., the maximum value). Within the proportional band, the MV is made proportional to the deviation (the difference between from SV and PV) and gradually decreased until the SV and PV match (i.e., until the deviation is 0), at which time the MV will be at the minimum value of 0% (or 50%, depending on the setting of the manipulated variable output designation parameter). The MV will also be 0% when the PV is larger than the SV.

The proportional band is expressed as a percentage of the total input range. The smaller the proportional band, the larger the proportional constant and the stronger the corrective action will be. With proportional action an offset (residual deviation) generally occurs, but the offset can be reduced by making the proportional band smaller. If it is made too small, however, hunting will occur.

Proportional Action (Reverse Action)



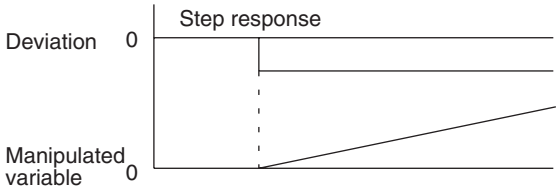
Adjusting the Proportional Band



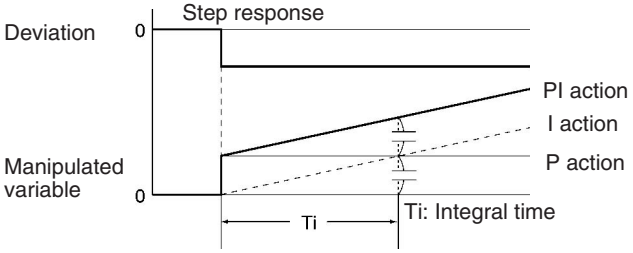
Integral Action (I)

Combining integral action with proportional action reduces the offset according to the time that has passed, so that the PV will match the SV. The strength of the integral action is indicated by the integral time, which is the time required for the manipulated variable of the integral action to reach the same level as the manipulated variable of the proportional action with respect to the step deviation, as shown in the following illustration. The shorter the integral time, the stronger the correction by the integral action will be. If the integral time is too short, the correction will be too strong and will cause hunting to occur.

Integral Action



Pi Action and Integral Time

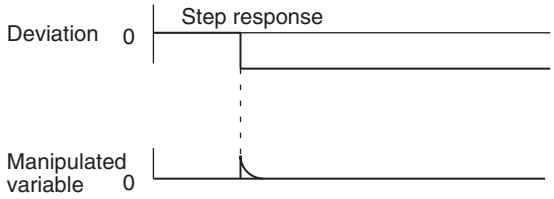


Derivative Action (D)

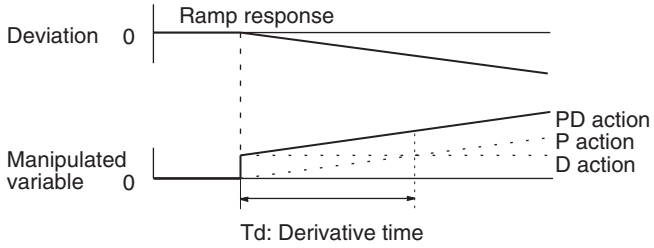
Proportional action and integral action both make corrections with respect to the control results, so there is inevitably a response delay. Derivative action compensates for that drawback. In response to a sudden disturbance it delivers a large manipulated variable and rapidly restores the original status. 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



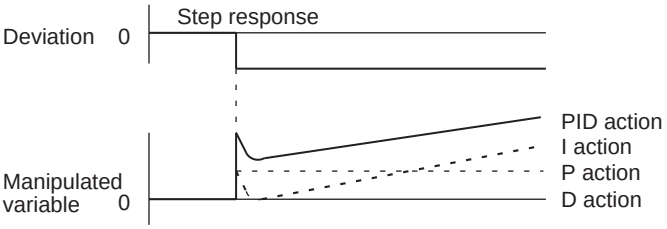
PD Action and Derivative Time



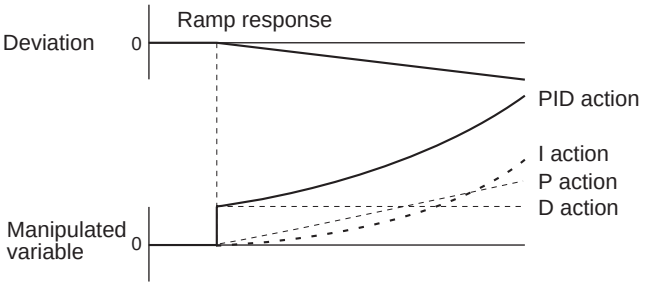
PID Action

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



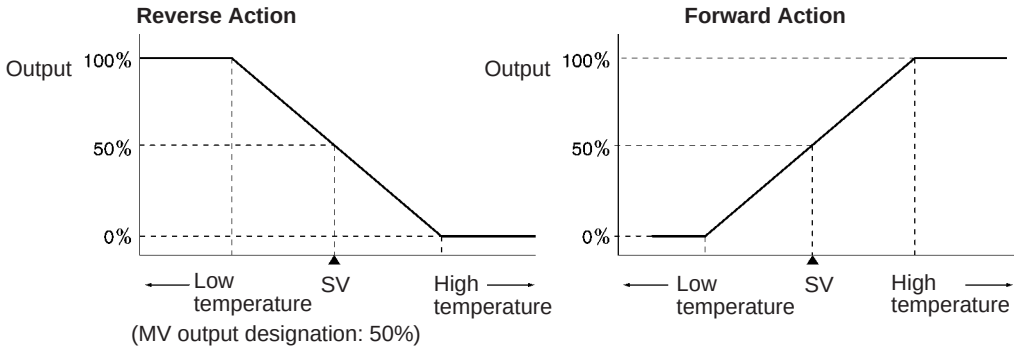
Ramp Response of PID Control Action Output



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.

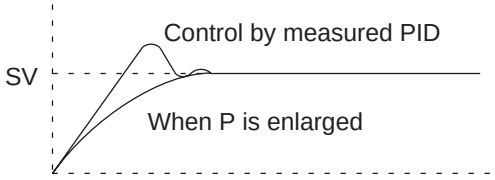
- Forward action: MV is increased when the PV is larger than the SV.
- Reverse action: MV is increased when the PV is smaller than the SV.



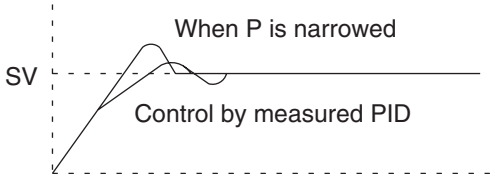
Adjusting PID Parameters

The general relationship between PID parameters and control status is shown below.

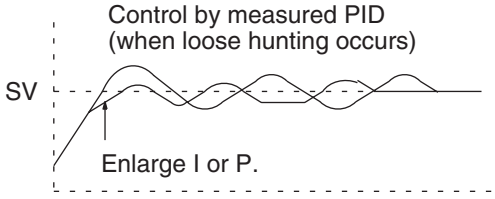
- When it is not a problem if a certain amount of time is required for stabilization (settlement time), but it is important not to cause overshooting, then enlarge the proportional band.



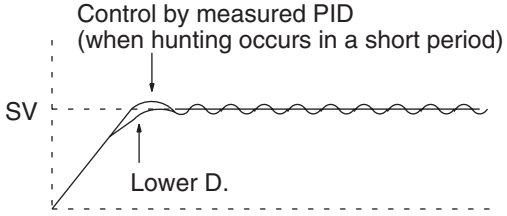
- When overshooting is not a problem but it is desirable to quickly stabilize control, then narrow the proportional band. If the proportional band is narrowed too much, however, then hunting may occur.



- When there is broad hunting, or when operation is tied up by overshooting and undershooting, it is probably because integral action is too strong. The hunting will be reduced if the integral time is increased or the proportional band is enlarged.



- If the period is short and hunting occurs, it may be that the control system response is quick and the derivative action is too strong. In that case, set the derivative action lower.



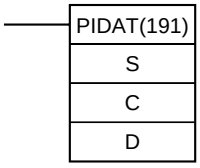
3-18-2 PID CONTROL WITH AUTOTUNING: PIDAT(191)

Purpose

Executes PID control according to the specified parameters. The PID constants can be autotuned.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: Input word
C: First parameter word
D: Output word

Variations

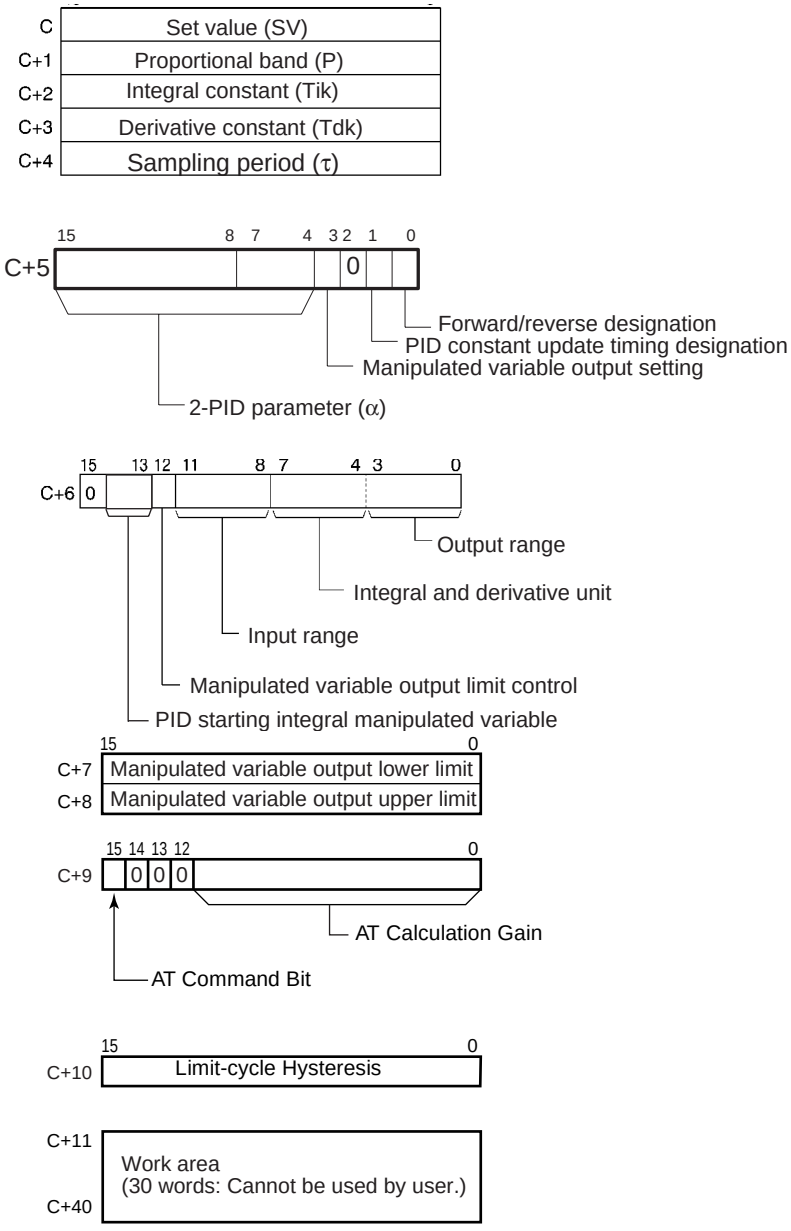
Variations	**Executed Each Cycle for ON Condition**	PIDAT(191)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Parameters

The following diagrams show the locations of the parameter data. For details on the parameters, refer to *PID Parameter Settings* in this section.



Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6105	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W473	W000 to W511
Holding Bit Area	H000 to H511	H000 to H473	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A921	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4057	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4057	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32729	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32729	E00000 to E32767

Area	S	C	D
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32729 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	DR0 to DR15	---	DR0 to DR15
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15		

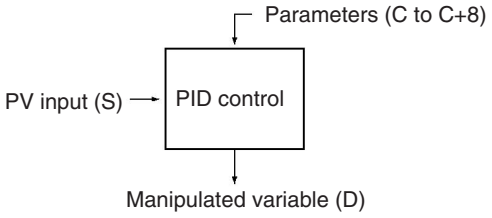
Description

When the execution condition is ON, PIDAT(191) carries out target value filtered PID control with two degrees of freedom according to the parameters designated by C (set value, PID constant, etc.). It takes the specified input range of binary data from the contents of input word S and carries out the PID action according to the parameters that are set. The result is then stored as the manipulated variable in output word D.

The parameter settings are read when the execution condition turns from OFF to ON, and the Error Flag will turn ON if the settings are outside of the permissible range.

If the settings are within the permissible range, PID processing will be executed using the initial values. Bumpless operation is not performed at this time. It will be used for manipulated variables in subsequent PID processing execution. (Bumpless operation is processing that gradually and continuously changes the manipulated variable in order to avoid the adverse effects of sudden changes.)

When the execution condition turns ON, the PV for the specified sampling period is entered and processing is performed.



Autotuning

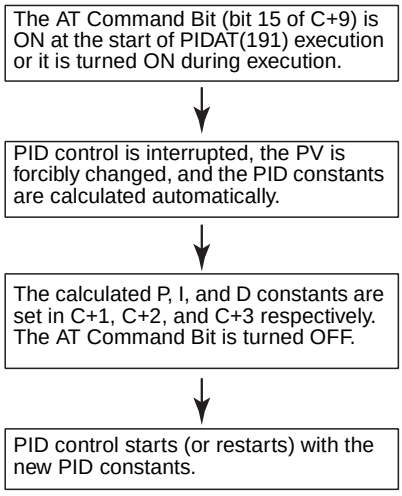
The status of the AT Command Bit (bit 15 of C+9) is checked every cycle. If this control bit is turned ON in a given cycle, PIDAT(191) will begin autotuning the PID constants. (The changes in the SV will not be reflected while autotuning is being performed.)

The limit-cycle method is used for autotuning. PIDAT(191) forcibly changes the manipulated variable (max. manipulated variable ↔ min. manipulated variable) and monitors the characteristics of the controlled system. The PID constants are calculated based on the characteristics that were observed,

and the new P, I, and D constants are stored automatically in C+1, C+2, and C+3. At this point, the AT Command Bit (bit 15 of C+9) is turned OFF and PID control resumes with the new PID constants in C+1, C+2, and C+3.

- If the AT Command Bit is ON when PIDAT(191) execution begins, autotuning will be performed first and then PID control will start with the calculated PID constants.
- If the AT Command Bit is turned ON during PIDAT(191) execution, PIDAT(191) interrupts the PID control being performed with the user-set PID constants, performs autotuning, and then resumes PID control with the calculated PID constants.

The following flowchart shows the autotuning procedure:



Note

1. If autotuning is interrupted by turning OFF the AT Command Bit during autotuning, PID control will start with the PID constants that were being used before autotuning began.
2. Also, if an AT execution error occurs, PID control will start with the PID constants that were being used before autotuning began.

In both cases described in notes 1 and 2, the PID constants will be enabled if they were already calculated when autotuning was interrupted.

PID Control

The number of valid input data bits within the 16 bits of the PV input (S) is designated by the input range setting in C+6, bits 08 to 11. For example, if 12 bits (4 hex) is designated for the input range, the range from 0000 hex to 0FFF hex will be enabled as the PV. (Values greater than 0FFF hex will be regarded as 0FFF hex.)

The set value range also depends on the input range.

Measured values (PV) and set values (SV) are in binary without sign, from 0000 hex to the maximum value of the input range.

The number of valid output data bits within the 16 bits of the manipulated variable output is designated by the output range setting in C+6, bits 00 to 03. For example, if 12 bits (4 hex) is designated for the output range, the range from 0000 hex to 0FFF hex will be output as the manipulated variable.

For proportional operation only, the manipulated variable output when the PV equals the SV can be designated as follows:

0: Output 0%
1: Output 50%.

The direction of proportional operation can be designated as either forward or reverse.

The upper and lower limits of the manipulated variable output can be designated.

The sampling period can be designated in units of 10 ms (0.01 to 99.99 s), but the actual PID action is determined by a combination of the sampling period and the time of PIDAT(191) instruction execution (with each cycle).

The timing of enabling changes made to PID constants can be set to either 1) the beginning of PIDAT(191) instruction execution or 2) the beginning of PID instruction execution and each sampling period. Only the proportional band (P), integral constant (Tik), and derivative constant (Tdk) can be changed each sampling cycle (i.e., during PID instruction execution). The timing is set in bit 1 of C+5.

The same words cannot be used to store the PID parameters for more than one PIDAT(191) instruction. Even if the same parameters are used, use different words to store the PID parameters for different PIDAT(191) instructions.

When changing the PID constants manually, set the PID constant change enable setting (bit 1 of C+5) to 1 so that the values in C+1, C+2, and C+3 are refreshed each sampling period in the PID calculation. This setting also allows the PID constants to be adjusted manually after autotuning.

Of the PID parameters (C to C+38), only the following parameters can be changed when the execution condition is ON. When any other values have been changed, be sure to change the execution condition from OFF to ON to enable the new settings.

- Set value (SV) in C
  (Can be changed during PID control only. An SV change during autotuning will not be reflected.)
- PID constant change enable setting (bit 1 of C+5)
- P, I, and D constants in C+1, C+2, and C+3
  (Changes to these constants will be reflected each sampling period only if the PID constant change enable setting (bit 1 of C+5) is set to 1.)
- AT Command Bit (bit 15 of C+9)
- AT Calculation Gain (bits 0 to 14 of C+9) and Limit-cycle Hysteresis (C+10) (These values are read when autotuning starts.)

Note The PIDAT(191) instruction is the same as the PID(190) instruction with the added autotuning (AT) function, so the PID control operations are identical. Refer to *3-18-1 PID CONTROL: PID(190)* for details on PID control operations and examples.

Flags

Name	Label	Operation
Error Flag	ER	ON if the C data is out of range. ON if the actual sampling period is more than twice the designated sampling period. ON if an error occurred during autotuning. OFF in all other cases.
Greater Than Flag	>	ON if the manipulated variable after the PID action exceeds the upper limit. OFF in all other cases.

Name	Label	Operation
Less Than Flag	<	ON if the manipulated variable after the PID action is below the lower limit. OFF in all other cases.
Carry Flag	CY	ON while PID control is being executed. OFF in all other cases.

Precautions

PIDAT(191) is executed as if the execution condition was a STOP-RUN signal. PID calculations are executed when the execution condition remains ON for the next cycle after C+11 to C+40 are initialized. Therefore, when using the Always ON Flag (ON) as an execution condition for PIDAT(191), provide a separate process where C+11 to C+40 are initialized when operation is started.

If the C data is out of range, an error will occur and the Error Flag will turn ON.

If an error occurred during autotuning, the Error Flag will turn ON.

If the actual sampling period is more than twice the designated sampling period, an error will occur and the Error Flag will turn ON. PID control will still be executed, however.

The Carry Flag turns ON while PID control is being executed.

The Greater Than Flag turns ON if the manipulated variable after the PID action exceeds the upper limit. At this time, the results are output at the upper limit.

The Less Than Flag turns ON if the manipulated variable after the PID action is below the lower limit. At this time, the results are output at the lower limit.

PID Parameter Settings

Control data	Item	Contents	Setting range	Change with ON input condition
C	Set value (SV)	The target value of the process being controlled.	Binary data (of the same number of bits as specified for the input range)	Allowed
C+1	Proportional band	The parameter for P action expressing the proportional control range/total control range.	0001 to 270F hex (1 to 9999); (0.1% to 999.9%, in units of 0.1%)	Can be changed with input condition ON if bit 1 of C+5 is 1.
C+2	Tik Integral Constant	A constant expressing the strength of the integral action. As this value increases, the integral strength decreases.	0001 to 1FFF hex (1 to 8191); (9999 = Integral operation not executed) (See note 1.)	
C+3	Tdk Derivative Constant	A constant expressing the strength of the derivative action. As this value increases, the derivative strength decreases.	0001 to 1FFF hex (1 to 8191); (0000 = Derivative operation not executed) (See note 1.)	
C+4	Sampling period (τ)	Sets the period for executing the PID action.	0001 to 270F hex (1 to 9999); (0.01 to 99.99 s, in units of 10 ms)	Not allowed
Bits 04 to 15 of C+5	2-PID parameter (α)	The input filter coefficient. Normally use 0.65 (i.e., a setting of 000). The filter efficiency decreases as the coefficient approaches 0.	000 hex: α = 0.65 Setting from 100 to 163 hex means that the value of the rightmost two digits is set from α= 0.00 to α= 0.99. (See note 2.)	
Bit 03 of C+5	Manipulated variable output designation	Designates the manipulated variable output for when the PV equals the SV.	0: Output 0% 1: Output 50%	

Control data	Item	Contents	Setting range	Change with ON input condition
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 (unit version 4.0 or later only)	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 5: 13 bits 1: 9 bits 6: 14 bits 2: 10 bits 7: 15 bits 3: 11 bits 8: 16 bits 4: 12 bits	
Bits 04 to 07 of C+6	Integral and derivative unit	Determines the unit for expressing the integral and derivative constants.	1: Sampling period multiple 9: Time (unit: 100 ms)	
Bits 00 to 03 of C+6	Output range	The number of output data bits. (The number of output bits is automatically the same as the number of input bits.)	0: 8 bits 5: 13 bits 1: 9 bits 6: 14 bits 2: 10 bits 7: 15 bits 3: 11 bits 8: 16 bits 4: 12 bits	
C+7	Manipulated variable output lower limit	The lower limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)	
C+8	Manipulated variable output upper limit	The upper limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)	

Control data	Item	Contents	Setting range	Change with ON input condition
Bit 15 of C+9	AT Command Bit	This control bit starts autotuning. • Set the AT Command Bit to 1 to perform autotuning. (Autotuning can be started while PIDAT(191) is being executed.) • This bit is turned OFF automatically when autotuning is completed. Autotuning will be interrupted if the AT Command Bit is turned OFF manually. In this case, the PID constants will be enabled if they were already calculated when autotuning was interrupted.	As a Control Bit: • 0 → 1: Executes autotuning. • 1 → 0: Interrupts autotuning. (PID(191) turns the bit OFF automatically when autotuning is completed. As a Flag: 0: Autotuning is not being executed. 1: Autotuning is being executed.	Allowed
Bits 00 to 11 of C+9	AT Calculation Gain	Set this parameter to adjust the contribution of the PID calculation results to the stored values. Normally, leave this parameter set to its default (0000). • Increase the value when emphasizing stability. • Decrease the value when emphasizing responsiveness.	0000 hex: 1.00 (Default) 0001 to 03E8 hex (1 to 1000); (0.01 to 10.00, in units of 0.01)	Allowed (These parameters are read when autotuning starts.)
C+10	Limit-cycle Hysteresis	Sets the hysteresis when the limit cycle is generated. The default setting for reverse operation turns ON the MV with a hysteresis of SV–20%. Increase this setting if a proper limit cycle cannot be generated because the PV is unstable. However, the AT accuracy will decline if the Limit-cycle Hysteresis is higher than necessary.	0000 hex: 0.20% (Default) 0001 to 03E8 hex: 0.01 to 10.00% in units of 0.01% FFFF hex: 0.00% **Note** The percentage is with respect to the input range.	

Note

1. When the unit is designated as 1, the range is from 1 to 8,191 times the period. When the unit is designated as 9, the range is from 0.1 to 819.1 s. When 9 is designated, set the integral and derivative times to within a range of 1 to 8,191 times the sampling period.
2. Setting the 2-PID parameter (α) to 000 yields 0.65, the normal value.

   When the manipulated variable output limit control is enabled (i.e., set to "1"), set the values as follows:

   0000 ≤ MV output lower limit ≤ MV output upper limit ≤ Max. value of output range

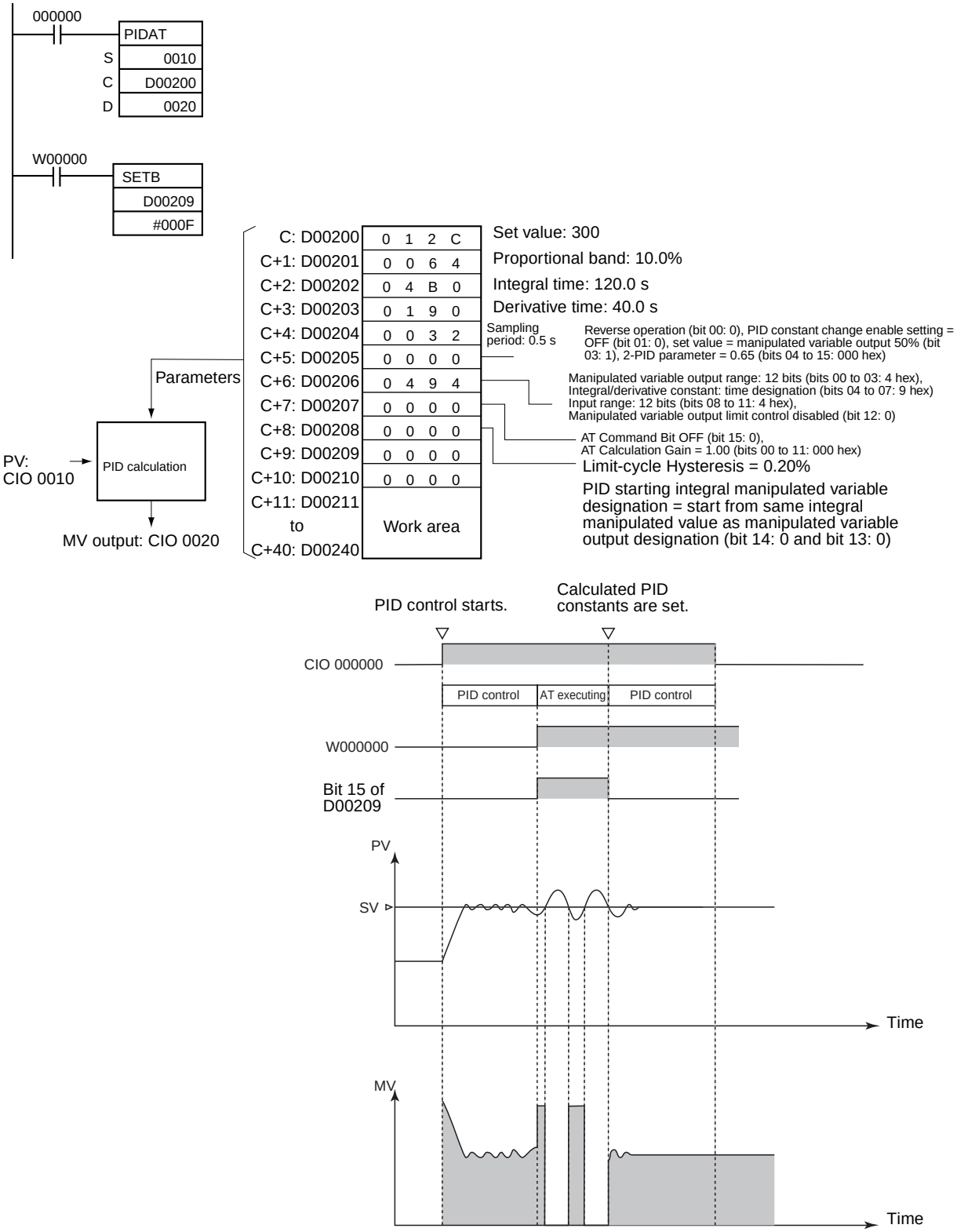
Example 1: Interrupting PID Control to Perform Autotuning

At the rising edge of CIO 000000 (OFF to ON), the work area in D00211 to D00240 is initialized according to the parameters (shown below) set in D00200 to D00208. After the work area has been initialized, PID control is executed and the manipulated variable is output to CIO 0020.

While CIO 000000 is ON, PID control is executed at the sampling period intervals according to the parameters set in D00200 to D00210. The manipulated variable is output to CIO 0020.

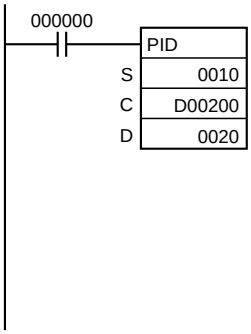
The PID constants used in PID calculations will not be changed even if the proportional band (P), integral constant (Tik), or derivative constant is changed after CIO 000000 turns ON.

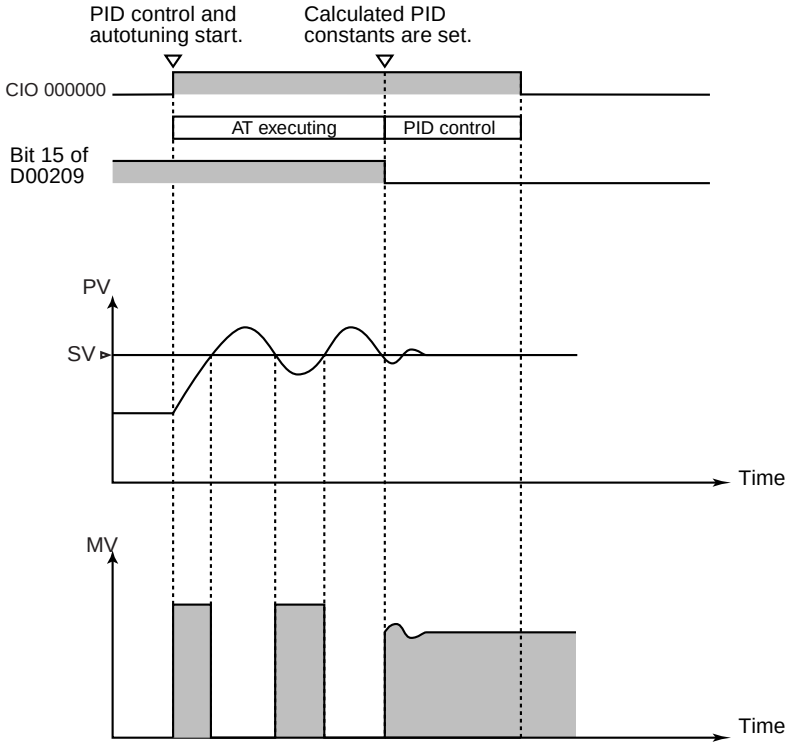
At the rising edge of W 000000 (OFF to ON), SETB(532) turns ON bit 15 of D00209 (C+9) and starts autotuning. When autotuning is completed, the calculated P, I, and D constants are written to C+1, C+2, and C+3. PID control is then restarted with the new PID constants.



Example 2: Starting PIDAT(191) with Autotuning

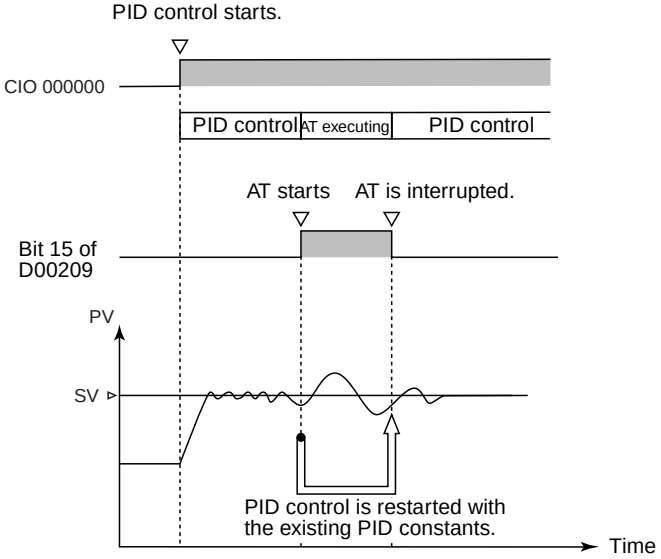
At the rising edge of CIO 000000 (OFF to ON), autotuning will be performed first if bit 15 of D00209 (C+9) is ON. When autotuning is completed, the calculated P, I, and D constants are written to C+1, C+2, and C+3. PID control is then started with the calculated PID constants.





Example 3: Interrupting Autotuning Before Completion

Autotuning can be interrupted by turning bit 15 of D00209 (C+9) from ON to OFF. PID control will be restarted with the P, I, and D constants that were in effect before autotuning was started.

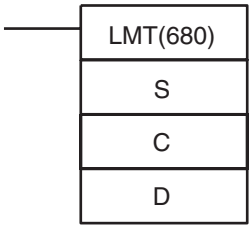


3-18-3 LIMIT CONTROL: LMT(680)

Purpose

Controls output data according to whether or not input data is within upper and lower limits.

Ladder Symbol



S: Input word

C: First limit word

D: Output word

Variations

Variations		
	Executed Each Cycle for ON Condition	LMT(680)
	Executed Once for Upward Differentiation	@LMT(680)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W510	W000 to W511
Holding Bit Area	H000 to H511	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to 959	A000 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	DR0 to DR15	---	DR0 to DR15

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

Description

When the execution condition is ON, LMT(680) controls output data according to whether or not the specified input data (signed 16-bit binary) is within the upper and lower limits. The contents of words C and C+1 are as follows:

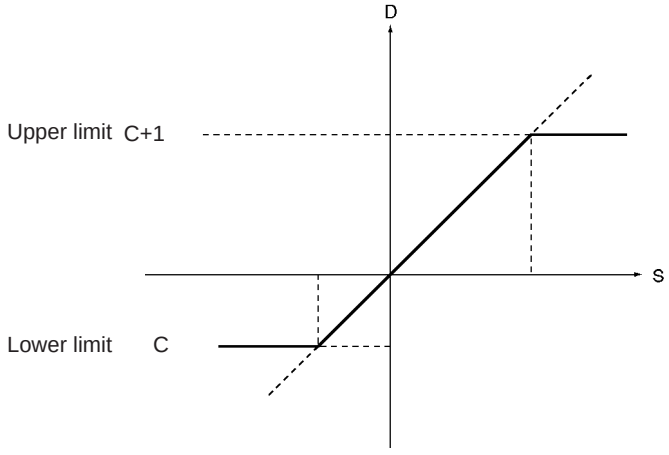
C	Lower limit data (minimum output data)
C+1	Upper limit data (maximum output data)

C and C+1 must have the same area classification.

If the input data (S) is less than the lower limit (C), the lower limit data will be output to D and the Less Than Flag will turn ON.

If the input data (S) is greater than the upper limit (C+1), the upper limit data will be output to D and the Greater Than Flag will turn ON.

If the input data (S) is greater than or equal to the lower limit (C) and less than or equal to the upper limit (C+1), the input data (S) will be output to D.



Flags

Name	Label	Operation
Error Flag	ER	ON if the upper limit is less than the lower limit. OFF in all other cases.
Greater Than Flag	>	ON if the input data (S) is greater than the upper limit. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Less Than Flag	<	ON if the input data (S) is less than the lower limit. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the result is “1.” OFF in all other cases.

Precautions

If the upper limit is less than the lower limit, an error will occur and the Error Flag will turn ON.

If the input data (S) is greater than the upper limit, the Greater Than Flag will turn ON.

If the output word D is 0000 hex, the Equals Flag will turn ON.

If the input data (S) is less than the lower limit, the Less Than Flag will turn ON.

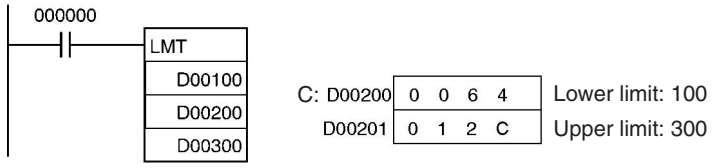
If the status of the leftmost bit of the output word D is “1,” the Negative Flag will turn ON.

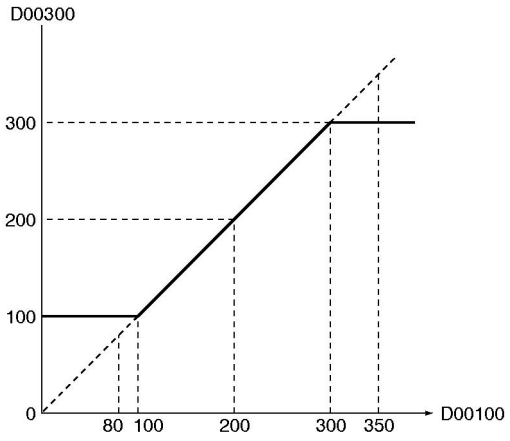
Example

If D00100 is 0050 hex (80), then 0064 hex (100) will be output to D00300 because 80 is less than the lower limit of 100.

If D00100 is 00C8 hex (200), then 0064 hex (100) will be output to D00300 because 200 is within the upper and lower limits.

If D00100 is 012C hex (300), then 015E hex (350) will be output to D00300 because 350 is greater than the upper limit of 300.



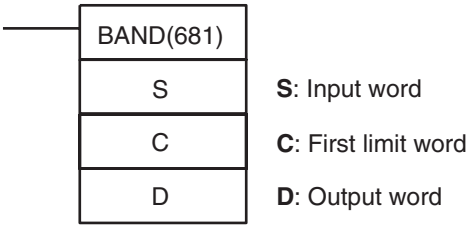


3-18-4 DEAD BAND CONTROL: BAND(681)

Purpose

Controls output data according to whether or not input data is within the lower and upper limits of the range (dead band range.)

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	BAND(681)
	Executed Once for Upward Differentiation	@BAND(681)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W510	W000 to W511
Holding Bit Area	H000 to H511	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

When the execution condition is ON, BAND(681) controls output data according to whether or not the specified input data (signed 16-bit binary) is within the upper and lower limits (dead band). The contents of words C and C+1 are as follows:

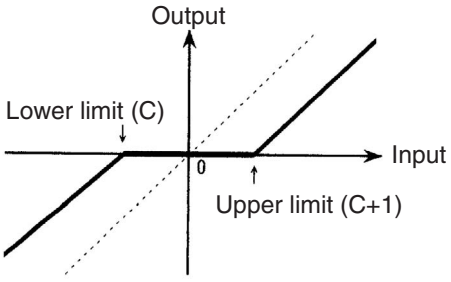
C	Lower limit data (dead band lower limit)
C+1	Upper limit data (dead band upper limit)

C and C+1 must have the same area classification.

If the input data (S) is greater than or equal to the lower limit (C) and less than or equal to the upper limit (C+1), 0000 (hex) will be output to D and the Equals Flag will turn ON.

If the input data (S) is less than the lower limit (C), the difference between the input data minus the lower limit data will be output to D and the Less Than Flag will turn ON.

If the input data (S) is greater than the upper limit (C+1), the difference between the input data minus the upper limit data will be output to D and the Greater Than Flag will turn ON.



If the output data is smaller than the 8000 (hex) or if is greater than 7FFF, the sign will be reversed. For example, for a lower limit of 0100 (hex) and input data of 8000 (hex), the output data will be as follows:
8000 (hex) [–32768] – 0100 (hex) [256] = 7F00 (hex) [32512]

Flags

Name	Label	Operation
Error Flag	ER	ON if the upper limit is less than the lower limit. OFF in all other cases.
Greater Than Flag	>	ON if the input data (S) is greater than the upper limit. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Less Than Flag	<	ON if the input data (S) is less than the lower limit. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the result is “1.” OFF in all other cases.

Precautions

If the upper limit is less than the lower limit, an error will occur and the Error Flag will turn ON.

If the input data (S) is greater than the upper limit, the Greater Than Flag will turn ON.

If the output word D is 0000 hex, the Equals Flag will turn ON.

If the input data (S) is less than the lower limit, the Less Than Flag will turn ON.

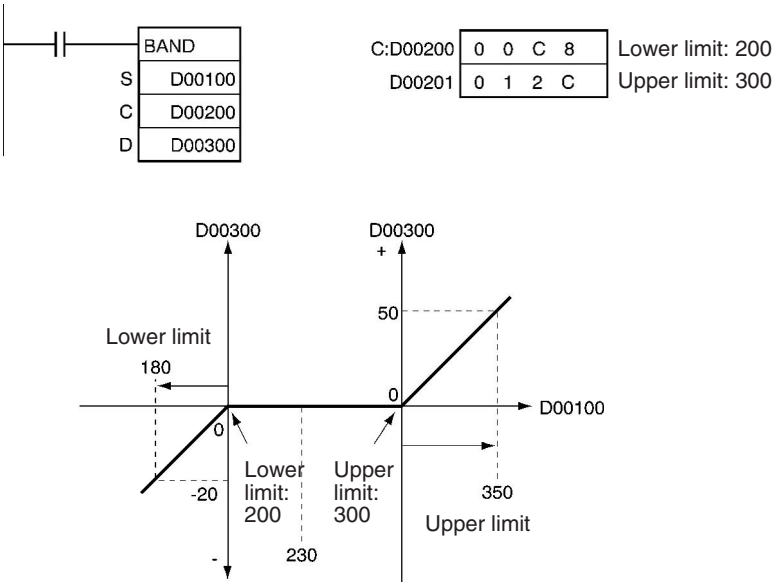
If the status of the leftmost bit of the output word D is “1,” the Negative Flag will turn ON.

Example

If D00100 is 00B4 hex (180), then 180–200=FFEC hex (–20) will be output to D00300 because 180 is less than the lower limit of 200.

If D00100 is 00E6 hex (230), then 0 will be output to D00300 because 230 is within the upper and lower limits.

If D00100 is 015E hex (350), then 350–300=0032 hex (50) will be output to D00300 because 350 is greater than the upper limit of 300.

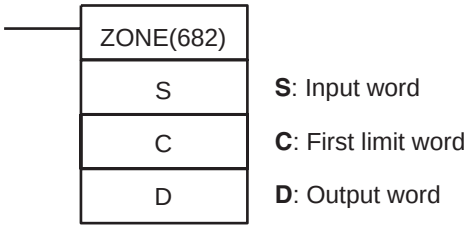


3-18-5 DEAD ZONE CONTROL: ZONE(682)

Purpose

Adds the specified bias to input data and outputs the result.

Ladder Symbol



S: Input word

C: First limit word

D: Output word

Variations

Variations	**Executed Each Cycle for ON Condition**	ZONE(682)
	Executed Once for Upward Differentiation	@ZONE(682)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W510	W000 to W511
Holding Bit Area	H000 to H511	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32766	E00000 to E32767

Area	S	C	D
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

When the execution condition is ON, ZONE(682) adds the specified bias to the specified input data (signed 16-bit binary) and places the result in a specified word. The contents of words C and C+1 are as follows:

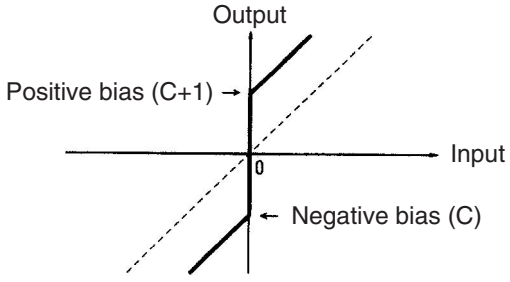
C	Negative bias
C+1	Positive bias

C and C+1 must have the same area classification.

If the input data (S) is less than zero, the input data plus the negative bias will be output to D and the Less Than Flag will turn ON.

If the input data (S) is greater than zero, the input data plus the positive bias will be output to D and the Greater Than Flag will turn ON.

If the input data (S) is equal to zero, 0000 will be output to D and the Equals Flag will turn ON.



If the output data is smaller than the 8000 (hex) or if is greater than 7FFF, the sign will be reversed. For example, for a negative bias value of FF00 (hex) and input data of 8000 (hex), the output data will be as follows:
8000 (hex) [–32768] – FF00 (hex) [–256] = 7F00 (hex) [32512]

Flags

Name	Label	Operation
Error Flag	ER	ON if the upper limit is less than the lower limit. OFF in all other cases.
Greater Than Flag	>	ON if the input data (S) is greater than the upper limit. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Less Than Flag	<	ON if the input data (S) is less than the lower limit. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the result is “1.” OFF in all other cases.

Precautions

If the upper limit is less than the lower limit, an error will occur and the Error Flag will turn ON.

If the input data (S) is greater than the upper limit, the Greater Than Flag will turn ON.

If the output word D is 0000 hex, the Equals Flag will turn ON.

If the input data (S) is less than the lower limit, the Less Than Flag will turn ON.

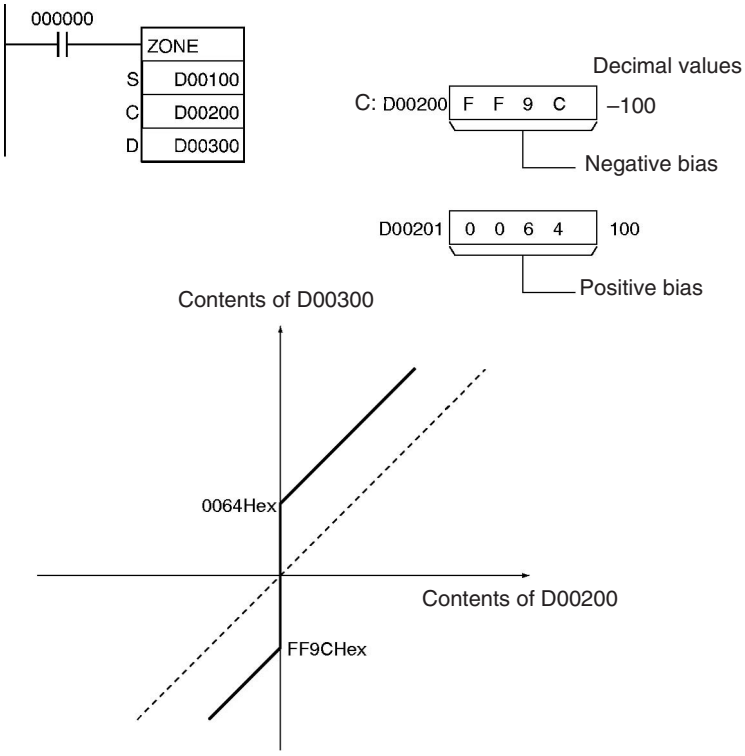
If the status of the leftmost bit of the output word D is “1,” the Negative Flag will turn ON.

Example

When CIO 000000 is ON, a bias of –100 will be applied to the value of D00100 if that value is less than 0, and the resulting value will be stored in D00300.

If the value of D00100 is 0, then 0000 hex will be stored in D00300.

If the value of D00100 is greater than 0, then a bias of +100 will be applied and the resulting value will be stored in D00300.



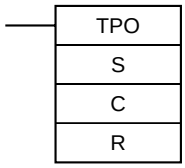
3-18-6 TIME-PROPORTIONAL OUTPUT: TPO(685)

Purpose

Inputs the duty ratio or manipulated variable from the specified word, converts the duty ratio to a time-proportional output based on the specified parameters, and outputs the result from the specified output.

This instruction is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



S: Input word

C: First parameter word

R: Pulse output bit

Variations

Variations	**Executed Each Cycle for ON Condition**	TPO(685)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

S: **Input Word**

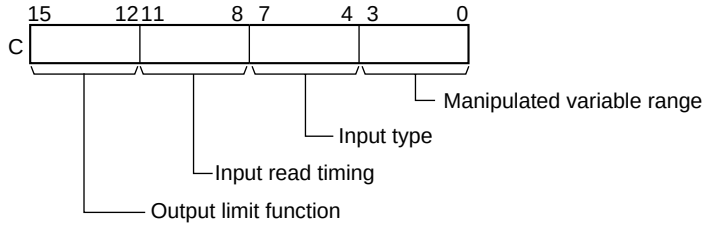
Specifies the input word containing the input duty ratio or manipulated variable. Bits 04 to 07 of C specify the input type, i.e., whether the input word contains an input duty ratio or manipulated variable. (Set these bits to 0 hex to specify a input duty ratio or to 1 hex to specify a manipulated variable.)

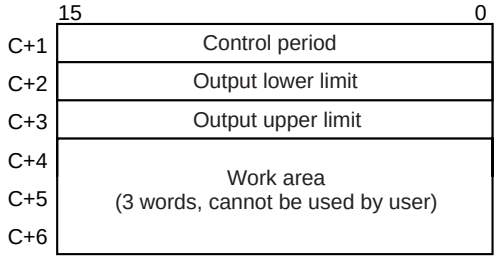
- Input duty ratio: 0000 to 2710 hex (0.00% to 100.00%)
- Input manipulated variable (See note.): 0000 to FFFF hex (0 to 65,535 max.) (Bits 00 to 03 of C specify the manipulated variable range, i.e., the number of valid bits in the manipulated variable. Specify the same number of bits as specified for the output range setting in PID(190).)

Note If S is a manipulated variable, specify the word containing the manipulated variable output from a PID(190) or PIDAT(191) instruction.

C to C+6: Parameters

The following diagram shows the locations of the parameter data. For details on the parameters, refer to *Parameter Settings* in this section.





Note: For details, see the description of each parameter.

R: Pulse Output Bit

Specifies the destination output bit for the pulse output.

Normally, specify an output bit allocated to a Transistor Output Unit and connect a solid state relay to the Transistor Output Unit.

Operand Specifications

Area	S	C	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6137	CIO 000000 to CIO 614315
Work Area	W000 to W511	W000 to W505	W00000 to W51115
Holding Bit Area	H000 to H511	H000 to H505	H00000 to H51115
Auxiliary Bit Area	A000 to 959	A000 to A953	A44800 to A95915
Timer Area	T0000 to T4095	T0000 to T4089	---
Counter Area	C0000 to C4095	C0000 to C4089	---
DM Area	D00000 to D32767	D00000 to D32761	---
EM Area without bank	E00000 to E32767	E00000 to E32761	---
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32761 (n = 0 to C)	---
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		---
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		---
Constants	#0000 to #FFFF (binary)	---	---
Data Registers	DR0 to DR15	---	---

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

Description

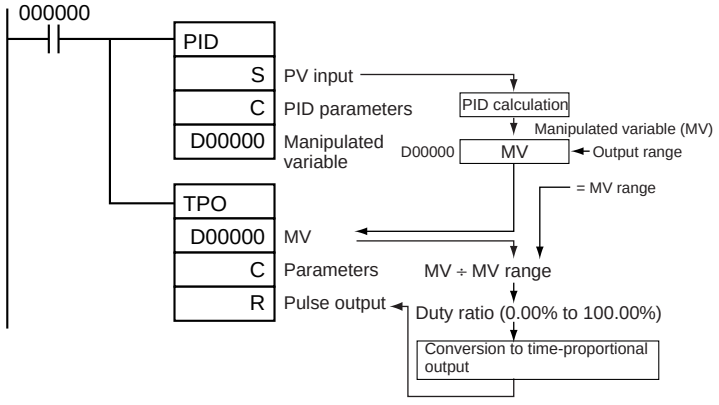
Receives a duty ratio or manipulated variable input from the word address specified by S, converts the duty ratio to a time-proportional output (see note) based on the parameters specified in words C to C+3, and outputs a pulse output to the bit specified by R.

Note A time-proportional output is changed proportionally based on the ON/OFF ratio in input word S. The period in which the ON and OFF status changes is known as the control period and is set in parameter word C+1.
Example: When the control period is 1 s and the input value is 50%, the bit is ON for 0.5 s and OFF for 0.5 s. When the control period is 1 s and the input value is 80%, the bit is ON for 0.8 s and OFF for 0.2 s.

Generally, TPO(685) is used together with PID(190) or PIDAT(191) and the PID instruction's manipulated variable result word (D) is specified as the input word (S) for the TPO(685) instruction. Also, an output bit allocated to a Transistor Output Unit is generally specified as R and a solid state relay is connected to the Transistor Output Unit to perform time-proportional control of a heater (proportional control of the ON/OFF ratio).

Combining TPO(685) with a PID Control Instruction

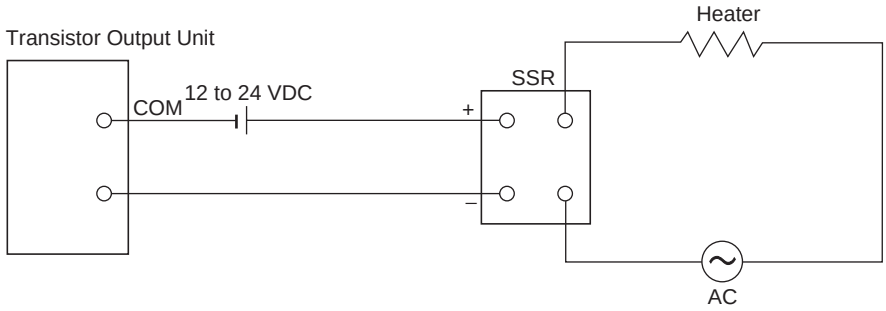
When combining TPO(685) with a PID control instruction, the manipulated variable input is divided by the manipulated variable range to calculate the duty ratio, that duty ratio is converted to a time-proportional output, and pulses are output.



In this case, set the same value for the PID Control instruction's output range and the TPO(685) instruction's manipulated variable range. For example, when the PID Control instruction's output range and the TPO(685) instruction's manipulated variable range are both set to 12 bits (0000 to 0FFF hex), the duty ratio is calculated by dividing the manipulated variable from the PID Control instruction by 0FFF hex and TPO(685) converts that duty ratio to a time-proportional output.

External Wiring Example

Connect the Transistor Output Unit to a solid state relay (SSR) as shown in the following diagram.



Parameter Settings

Control data		Item	Contents	Setting range	Change with ON input condition
Word	Bits				
C	00 to 03	Manipulated variable range	Specifies the number of input data bits.	0 hex: 8 bits 5 hex: 13 bits 1 hex: 9 bits 6 hex: 14 bits 2 hex: 10 bits 7 hex: 15 bits 3 hex: 11 bits 8 hex: 16 bits 4 hex: 12 bits	Allowed
	04 to 07	Input type	Specifies whether S contains a duty ratio or manipulated variable.	0 hex: Duty ratio Setting range for S: 0000 to 2710 hex (0.00 to 100.00%) 1 hex: Manipulated variable Setting range for S: 0000 to FFFF hex (0 to 65,535) (The maximum setting depends on the MV range set with bits 00 to 03 of C.)	Allowed
	08 to 11	Input read timing	Specifies the input read timing.	0 hex: Use the beginning value of the control period 1 hex: Use lower value 2 hex: Use higher value 3 hex: Continuous adjustment	Allowed
	12 to 15	Output limit control	Specifies whether the output limit function is enabled or disabled.	0 hex: Disabled 1 hex: Enabled (See note.)	Allowed
C+1	00 to 15	Control period	Control period (Time period in which the ON/OFF changes are made.)	0064 to 270F hex (1.00 to 99.99 s) Note: For example, 1.00 s is set as 0064 hex, and not 0001 hex.	Allowed
C +2	00 to 15	Output lower limit	Specifies the lower limit when the output limit is enabled.	0000 to 2710 hex (0 to 100.00%)	Allowed
C +3	00 to 15	Output upper limit	Specifies the upper limit when the output limit is enabled.	0000 to 2710 hex (0 to 100.00%)	Allowed
C+4	00 to 15	Work area	This work area is used by the system. It cannot be used by the user.	Cannot be used.	---
C+5	00 to 15				
C+6	00 to 15				

Note When the output limit control function is enabled, set the lower and upper limits as follows: 0000 hex ≤ lower limit ≤ upper limit ≤ 2710 hex.

Execution

- The instruction is executed while the input condition is ON.
- When instruction execution starts, the output bit (R) is turned ON/OFF according to the duty ratio.

- The parameters (in C to C+3) are read in real time each time that the instruction is executed. When changing the parameters, change all of them at the same time so that different sets of parameters are not mixed.
- The output (R) is turned ON/OFF when the instruction is executed and the accuracy of the output's ON/OFF timing is 10 ms max.
- Execution of the instruction stops when the input condition goes OFF. At that time, the elapsed time value will be reset and the control period will be initialized.
- The input type setting (bits 04 to 07 of C) determines whether the input word (S) contains a duty ratio or manipulated variable. When S contains the manipulated variable, the duty ratio is calculated by dividing the manipulated variable input by the manipulated variable range (bits 00 to 03 of C).

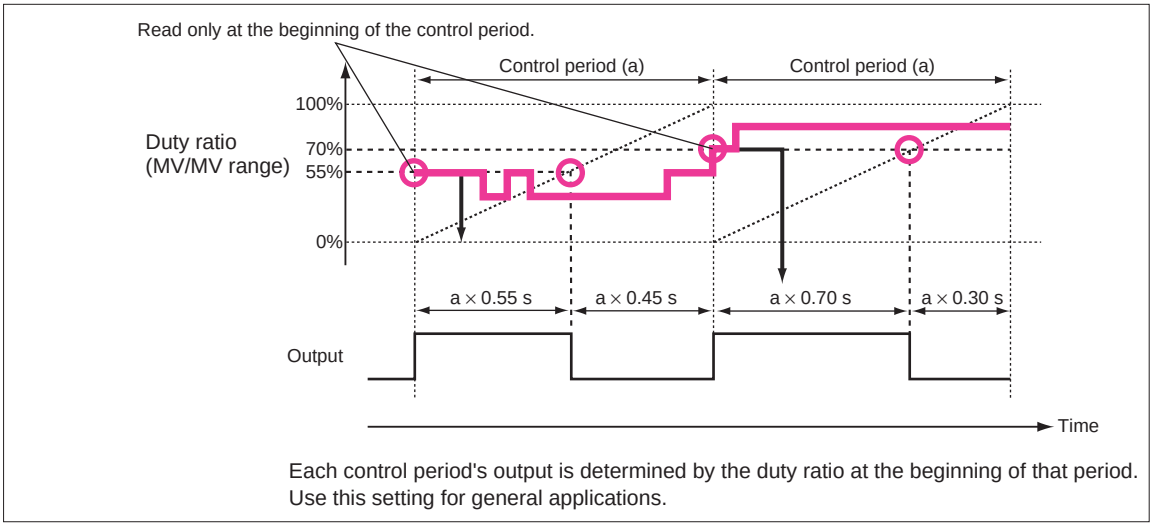
Input Read Timing Setting (C bits 08 to 11)

The input read timing setting (bits 08 to 11 of C) specifies when the input word (S) is read, as shown in the following table:

Input read timing	Description
0: Use the beginning value of the control period	The duty ratio input is read at the beginning of the control period and the ratio cannot be changed during the control period.
1: Use lower value	If the duty ratio input falls below the duty ratio at the beginning of the control period, the lower value will take precedence and the output ON time will be reduced accordingly.
2: Use higher value	If the duty ratio input rises above the duty ratio at the beginning of the control period, the higher value will take precedence and the output ON time will be increased accordingly.
3: Continuous adjustment	The duty ratio will be read in real time each time the instruction is executed and the ON/OFF operation will be repeated within the control period.

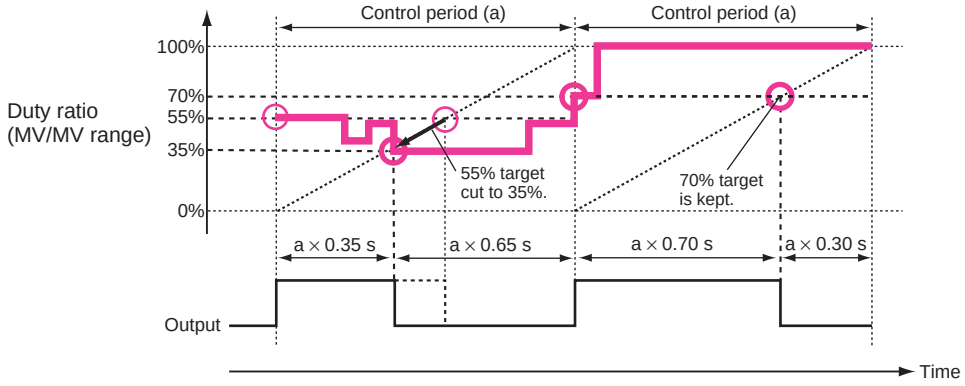
The following diagrams show the operation of each input read timing setting.

- Input time setting = 0 (Use the beginning value of the control period.)



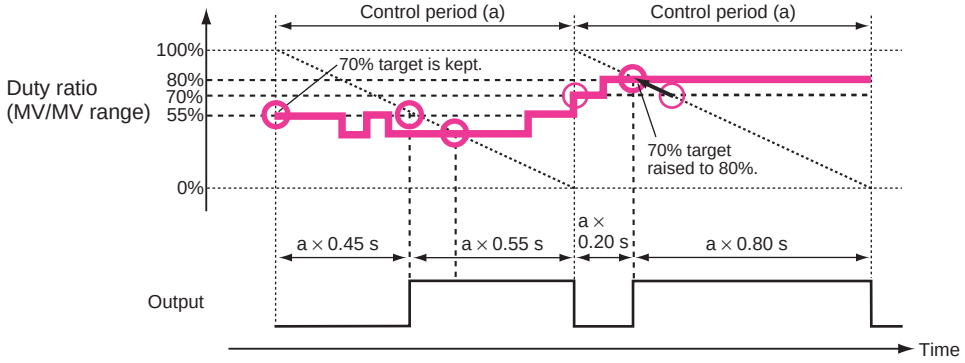
Each control period's output is determined by the duty ratio at the beginning of that period. Use this setting for general applications.

• Input time setting = 1 (Use lower value.)



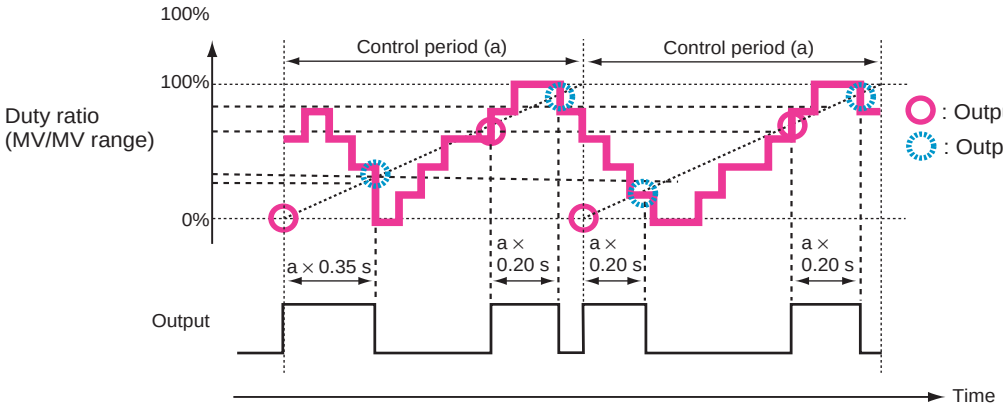
If the duty ratio falls below the initial value early enough, the duty ratio will be adjusted and the output will be turned OFF sooner.
Use this setting for applications such as avoiding overshooting when using time-proportional control to control heating and using a relatively long control period.

• Input time setting = 2 (Use higher value.)



If the duty ratio rises above the initial value early enough, the duty ratio will be adjusted and the output will be turned ON sooner. (With this setting the output's ON/OFF order is reversed and the output goes from OFF to ON.)
Use this setting for applications such as avoiding undershooting when using time-proportional control to control cooling and using relatively long control period.

• Input time setting = 3 (Continuous adjustment)



Changes in the duty ratio are monitored in real time. If the duty ratio falls below the initial value early enough, the duty ratio will be adjusted and the output will be turned OFF sooner. If the duty ratio rises again after that, the ratio will be adjusted again and the output will be turned ON. This process is repeated continuously.
Use this setting to improve responsiveness when the control period is relatively long and the duty ratio changes quickly. This setting is also appropriate for lighting or power applications that require precise control.

Flags

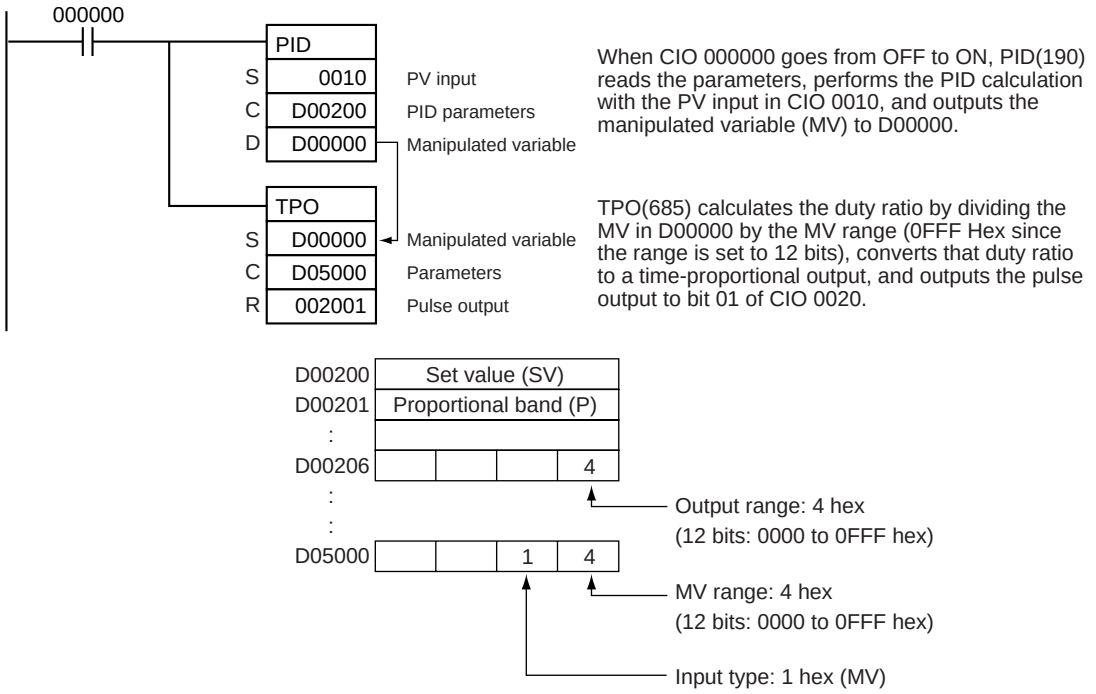
Name	Label	Operation
Error Flag	ER	ON if the input data in S is out of range. (The input data setting range depends on the input type setting.) ON if the C data is out of range. (The manipulated variable range will cause an error only when the input type is set to manipulated variable.) ON if the control period in C+1 is out of range. ON if the output limit function is enabled but the output lower limit (C+2) or output upper limit (C+3) is out of range. ON if the output limit function is enabled but the output lower limit (C+2) is less than or equal to the output upper limit (C+3). OFF in all other cases.

Example

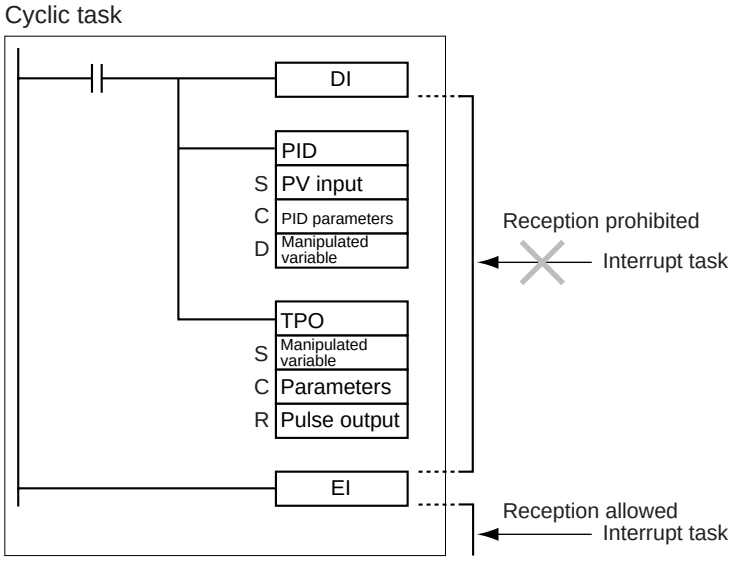
Example 1: Combining TPO(685) with PID(190)

When CIO 000000 is ON, TPO(685) takes the manipulated variable output from PID(190) (contained in D00000), 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 002001.

In this case, CIO 0020 is allocated to a Transistor Output Unit and bit CIO 002001 is connected to a solid state relay for heater control.



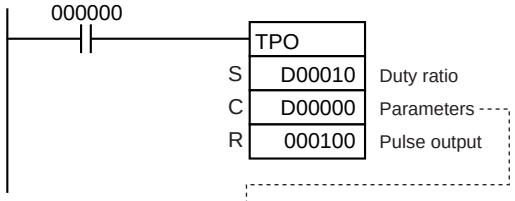
Note When using TPO(685) in combination with PID(190) in a cyclic task and also using an interrupt task, temporarily disable interrupts by executing DI(693) (DISABLE INTERRUPTS) ahead PID(190) and TPO(685). If interrupts are not disabled and an interrupt occurs between the PID(190) and TPO(685), the control period may be shifted.



Example 2: Using TPO(685) Alone

When CIO 000000 is ON, TPO(685) takes the duty ratio in D00010, converts the duty ratio to a time-proportional output, and outputs the pulses to CIO 000100.

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 D00010, converts that duty ratio to a time-proportional output, and outputs the pulse output to bit 00 of CIO 0001.

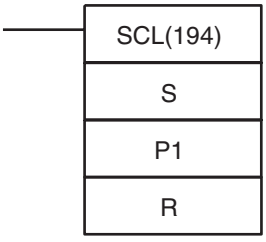
D00000	1	1	0	0	Duty ratio input, read initial value, and enable output limit function.
D00001	0	0	6	4	Control period = 1.00 s
D00002	0	7	D	0	Output lower limit = 20.00%
D00003	1	F	4	0	Output upper limit = 80.00%
D00004	Do not set.				
D00005	Do not set.				
D00006	Do not set.				
:					
:					
D00010	0 to 2710 hex				0 to 100.00%

3-18-7 SCALING: SCL(194)

Purpose

Converts unsigned binary data into unsigned BCD data according to the specified linear function.

Ladder Symbol



S: Source word

P1: First parameter word

R: Result word

Variations

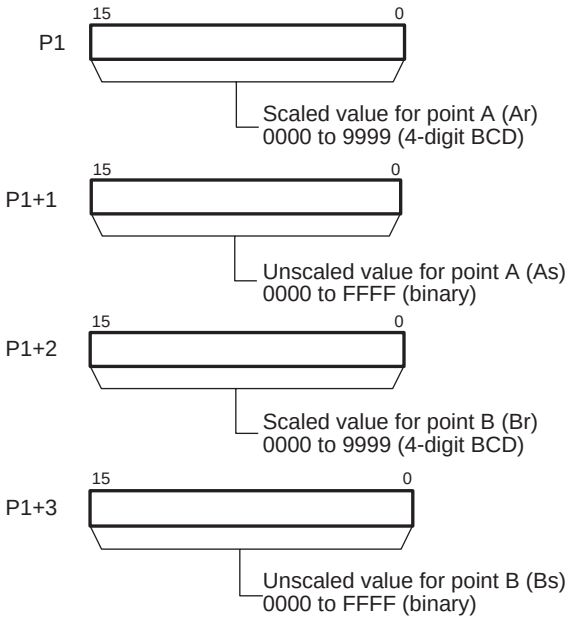
Variations		
	Executed Each Cycle for ON Condition	SCL(194)
	Executed Once for Upward Differentiation	@SCL(194)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

The contents of the four words starting with the first parameter word (P1) are shown in the following diagram.



Note P1 to P1+3 must be in the same area.

Operand Specifications

Area	S	P1	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6140	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W508	W000 to W511
Holding Bit Area	H000 to H511	H000 to H508	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A956	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4092	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4092	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32764	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32764	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32764 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	DR0 to DR15	---	DR0 to DR15

Area	S	P1	R
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SCL(194) is used to convert the unsigned binary data contained in the source word S into unsigned BCD data and place the result in the result word R according to the linear function defined by points (As, Ad) and (Bs, Bd). The address of the first word containing the coordinates of points (As, Ar) and (Bs, Br) is specified for the first parameter word P1. These points define by 2 values (As and Bs) before scaling and 2 values (Ar and Br) after scaling.

The following equations are used for the conversion.

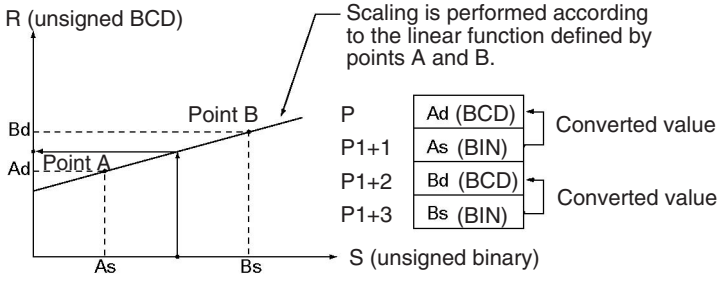
$$R = Bd - \frac{(Bd - Ad)}{\text{BCD conversion of } (Bs - As)} \times \text{BCD conversion of } (Bs - S)$$

The slope of the line is as follows:

$$R = Bd - \frac{(Bd - Ad)}{\text{BCD conversion of } (Bs - As)}$$

Points A and B can define a line with either a positive or negative slope. Using a negative slope enables reverse scaling.

The result will be rounded to the nearest integer. If the result is less than 0000, 0000 will be output as the result. If the result is greater than 9999, 9999 will be output.



SCL(194) can be used to scale the results of analog signal conversion values from Analog Input Units according to user-defined scale parameters. For example, if a 1 to 5-V input to an Analog Input Unit is input to memory as 0000 to 0FA0 hexadecimal, the value in memory can be scaled to 50 to 200°C using SCL(194).

SCL(194) converts unsigned binary to unsigned BCD. To convert a negative value, it will be necessary to first add the maximum negative value in the program before using SCL(194) (see example).

SCL(194) cannot output a negative value to the result word, R. If the result is a negative value, 0000 will be output to R.

Flags

Name	Label	Operation
Error Flag	ER	ON if the contents of C (Ar) or C+1 (Br) is not BCD. ON if the contents of C+1 (As) and C+3 (Bs) are equal. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.

Precautions

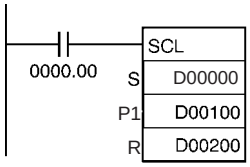
An error will occur and the Error Flag will turn ON if the values for Ar (C) and Br (C+2) are not in BCD, or if the values for As (C+1) and Bs (C+3) are equal.

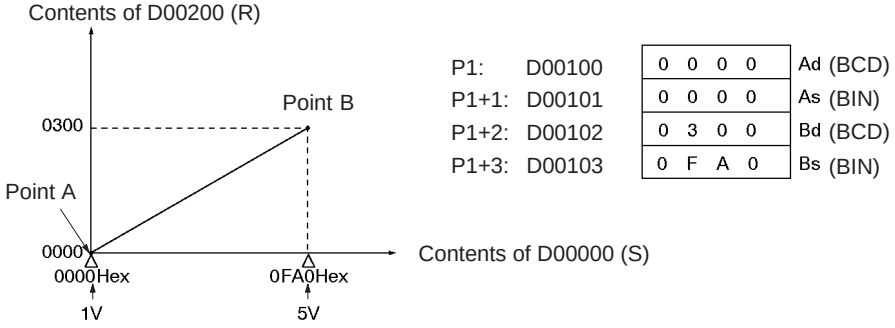
The Equals Flag will turn ON when the contents of the result word D is 0000.

Examples

In the following example, it is assume that an analog signal from 1 to 5 V is converted and input to D00000 as 0000 to 0FA0 hexadecimal. SCL(194) is used to convert (scale) the value in CIO 0200 to a value between 0000 and 0300 BCD.

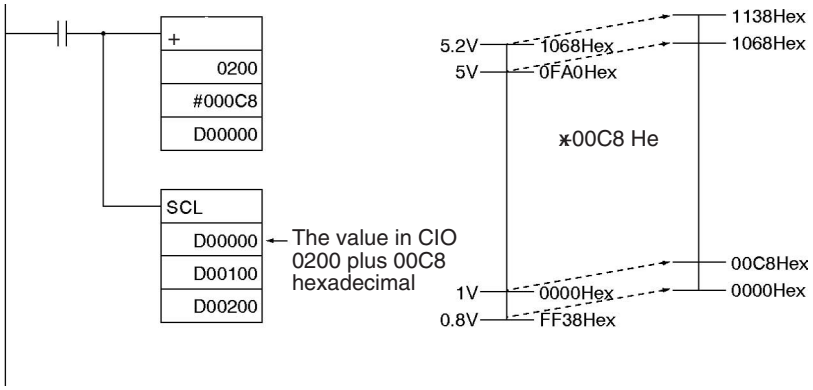
When CIO 000000 is ON, the contents of D00000 is scaled using the linear function defined by point A (0000, 0000) and point B (0FA0, 0300). The coordinates of these points are contained in D00100 to D00103, and the result is output to D00200.

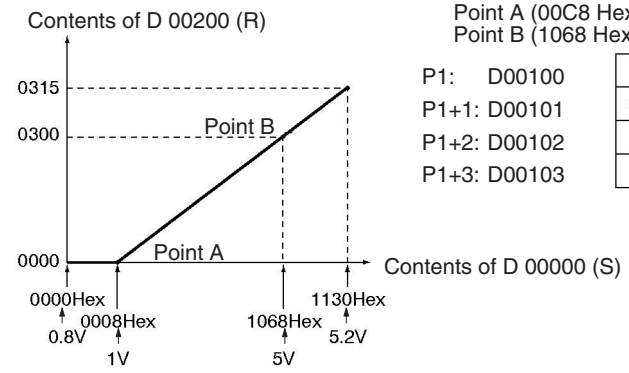




Negative Values

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.





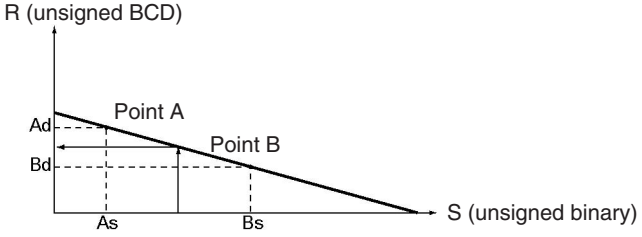
Point A (00C8 Hex → 0000 (BCD))
Point B (1068 Hex → 0300 (BCD))

P1: D00100	0	0	0	0	Ad (BCD)
P1+1: D00101	0	0	C	8	As (BIN)
P1+2: D00102	0	3	0	0	Bd (BCD)
P1+3: D00103	1	0	6	8	Bs (BIN)

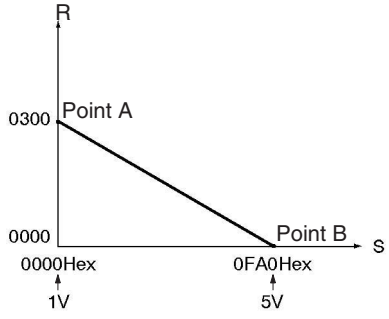
In this example, values from 0000 to 00C8 hexadecimal will be converted to negative values. SCL(194), however, can output only unsigned BCD values from 0000 to 9999, so 0000 BCD will be output whenever the contents of D00000 is between 0000 and 00C8 hexadecimal.

Reverse Scaling

Reverse scaling can also be used by setting As < Bs and Ar > Br. The following relationship will result.



Reverse scaling can be used, for example, to convert (reverse scale) 1 to 5 V (0000 to 0FA0 hexadecimal) to 0300 to 0000, respectively, as shown in the following diagram.

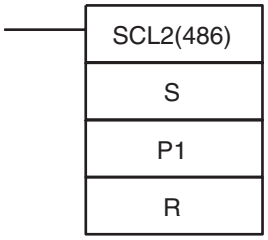


3-18-8 SCALING 2: SCL2(486)

Purpose

Converts signed binary data into signed BCD data according to the specified linear function. An offset can be input in defining the linear function.

Ladder Symbol



S: Source word

P1: First parameter word

R: Result word

Variations

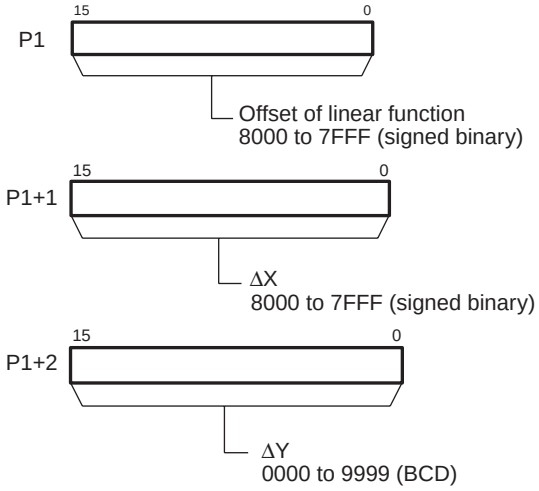
Variations		
	Executed Each Cycle for ON Condition	SCL2(486)
	Executed Once for Upward Differentiation	@SCL2(486)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

The contents of the three words starting with the first parameter word (P1) are shown in the following diagram.



Note P1 to P1+2 must be in the same area.

Operand Specifications

Area	S	P1	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6141	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W509	W000 to W511
Holding Bit Area	H000 to H511	H000 to H509	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A957	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4093	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4093	C0000 to C4095

Area	S	P1	R
DM Area	D00000 to D32767	D00000 to D32765	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32765	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32765 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SCL2(486) is used to convert the signed binary data contained in the source word S into signed BCD data (the BCD data contains the absolute value and the Carry Flag shows the sign) and place the result in the result word R according to the linear function defined by the slope (ΔX, ΔY) and an offset. The address of the first word containing ΔX, ΔY, and the offset is specified for the first parameter word P1. The sign of the result is indicated by the status of the Carry Flag (ON: negative, OFF: positive).

The following equations are used for the conversion.

$$R = \frac{\Delta Y}{\text{BCD conversion of } \Delta X} \times ((\text{BCD conversion of S}) - (\text{BCD conversion of offset})$$

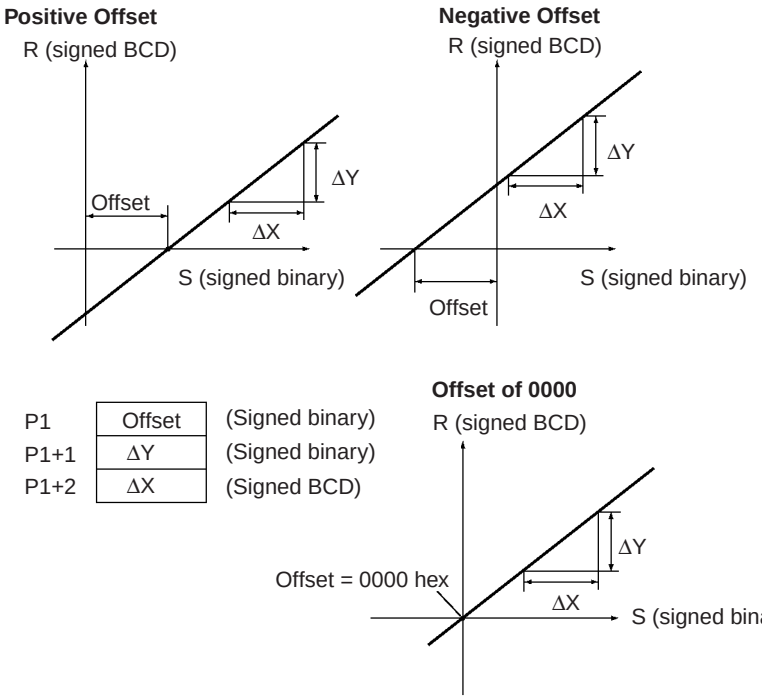
The slope of the line is $\Delta Y/\Delta X$.

The offset and slope can be a positive value, 0, or a negative value. Using a negative slope enables reverse scaling.

The result will be rounded to the nearest integer.

The result in R will be the absolute BCD conversion value and the sign will be indicated by the Carry Flag. The result can thus be between –9999 and 9999.

If the result is less than –9999, –9999 will be output as the result. If the result is greater than 9999, 9999 will be output.



SCL2(486) can be used to scale the results of analog signal conversion values from Analog Input Units according to user-defined scale parameters. For example, if a 1 to 5-V input to an Analog Input Unit is input to memory as 0000 to 0FA0 hexadecimal, the value in memory can be scaled to –100 to 200°C using SCL2(486).

SCL2(486) converts signed binary to signed BCD. Negative values can thus be handled directly for S. The result of scaling in R and the Carry Flag can also be used to output negative values for the scaling result.

Flags

Name	Label	Operation
Error Flag	ER	ON if the contents of C+1 (ΔX) is 0000. ON if the contents of C+2 (ΔY) is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Carry Flag	CY	ON if the result is negative. OFF if the result is zero or positive.

Precautions

An error will occur and the Error Flag will turn ON if the value for ΔX (C+1) is 0000 or if the value for ΔY (C+2) is not BCD.

The Equals Flag will turn ON when the contents of the result word D is 0000.

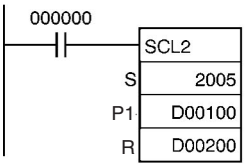
The Carry Flag will turn ON if the value placed in the result word is negative.

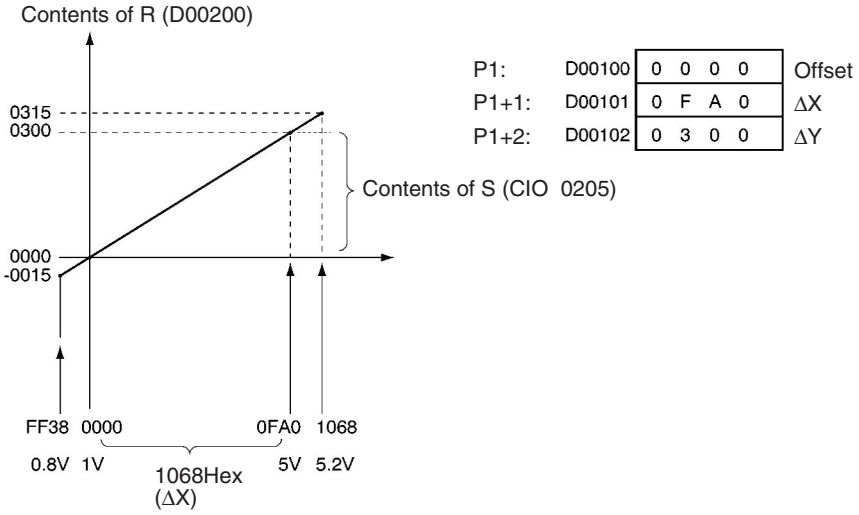
Examples

Scaling 1 to 5-V Analog Input to 0 to 300

In the following example, it is assumed that an analog signal from 1 to 5 V is converted and input to CIO 0205 as 0000 to 0FA0 hexadecimal. SCL2(486) is used to convert (scale) the value in CIO 0205 to a value between 0000 and 0300 BCD.

When CIO 000000 is ON, the contents of CIO 0205 is scaled using the linear function defined by ΔX (0FA0), ΔY (0300), and the offset (0). These values are contained in D00100 to D00102, and the result is output to D00200.

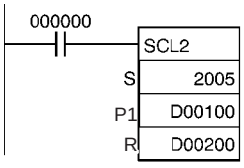


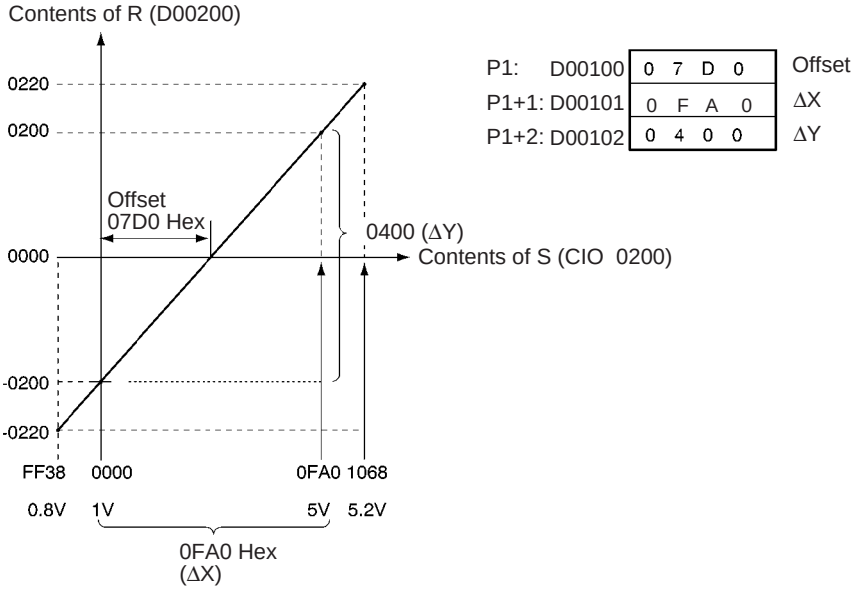


Scaling 1 to 5-V Analog Input to –200 to 200

In the following example, it is assume that an analog signal from 1 to 5 V is converted and input to CIO 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 000000 is ON, the contents of CIO 2005 is scaled using the linear function defined by ΔX (0FA0), ΔY (0400), and the offset (07D0). These values are contained in D00100 to D00102, and the result is output to D00200.



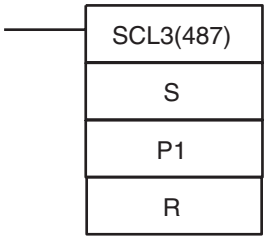


3-18-9 SCALING 3: SCL3(487)

Purpose

Converts signed BCD data into signed binary data according to the specified linear function. An offset can be input in defining the linear function.

Ladder Symbol



S: Source word

P1: First parameter word

R: Result word

Variations

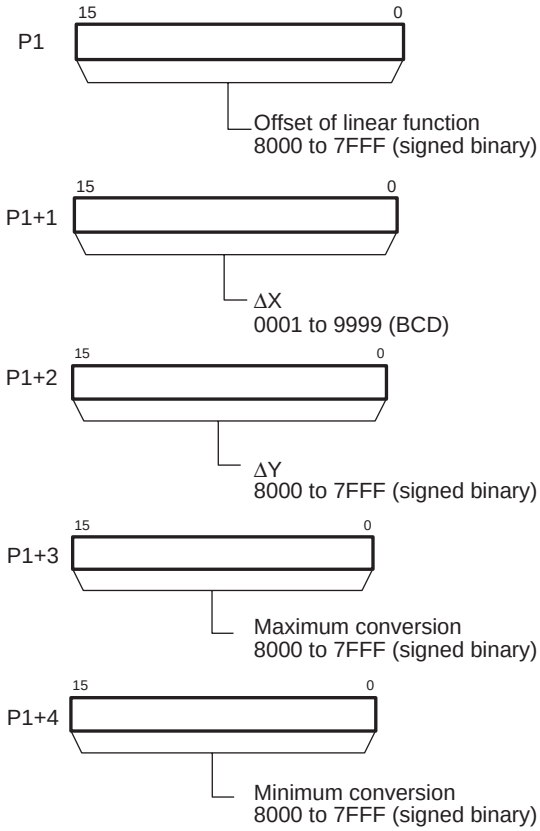
Variations		
	Executed Each Cycle for ON Condition	SCL3(487)
	Executed Once for Upward Differentiation	@SCL3(487)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

The contents of the five words starting with the first parameter word (P1) are shown in the following diagram.



Note P1 to P1+4 must be in the same area.

Operand Specifications

Area	S	P1	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6139	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W507	W000 to W511
Holding Bit Area	H000 to H511	H000 to H507	H000 to H511
Auxiliary Bit Area	A000 to A447 A448 to A959	A000 to A443 A448 to A955	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4091	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4091	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32763	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32763	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32763 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SCL3(487) is used to convert the signed BCD data (the BCD data contains the absolute value and the Carry Flag shows the sign) contained in the source word S into signed binary data and place the result in the result word R according to the linear function defined by the slope (ΔX, ΔY) and an offset. The maximum and minimum conversion values are also specified. The address of the first word containing ΔX, ΔY, the offset, the maximum conversion, and the minimum conversion is specified for the first parameter word P1.

The sign of the result is indicated by the status of the Carry Flag (ON: negative, OFF: positive). Use STC(040) and CLC(041) to turn the Carry Flag ON and OFF.

The following equations are used for the conversion.

$$R = \frac{\Delta Y}{\text{Binary conversion of } \Delta X} \times ((\text{Binary conversion of S}) + (\text{Offset}))$$

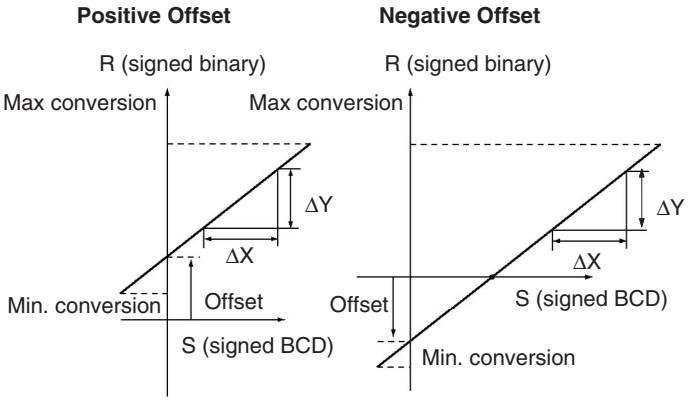
The slope of the line is ΔY/ΔX.

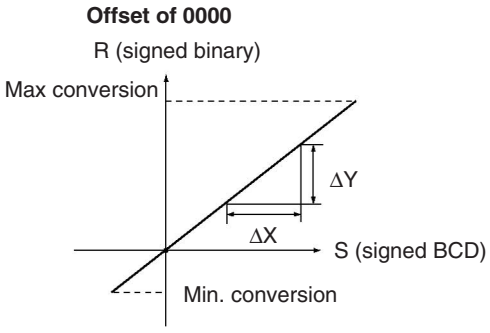
The offset and slope can be a positive value, 0, or a negative value. Using a negative slope enables reverse scaling.

The result will be rounded to the nearest integer.

The source value in S is treated as an absolute BCD value and the sign is indicated by the Carry Flag. The source value can thus be between –9999 and 9999.

If the result is less than the minimum conversion value, the minimum conversion value will be output as the result. If the result is greater than the maximum conversion value, the maximum conversion value will be output.





SCL3(487) is used to convert data using a user-defined scale to signed binary for Analog Output Units. For example, SCL3(487) can convert 0 to 200 °C to 0000 to 0FA0 (hex) and output an analog output signal 1 to 5 V from the Analog Output Unit.

Flags

Name	Label	Operation
Error Flag	ER	ON if the contents of S is not BCD. ON if the contents of C+1 (ΔX) is not between 0001 and 9999 BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Negative Flag	N	ON when the MSB of the R (the result) is 1. OFF in all other cases.

Precautions

An error will occur and the Error Flag will turn ON if the contents of S is not BCD or if the value for ΔX (C+1) is not between 0001 and 9999 BCD.

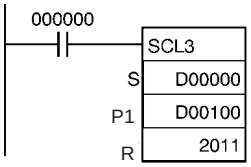
The Equals Flag will turn ON when the contents of the result word D is 0000.

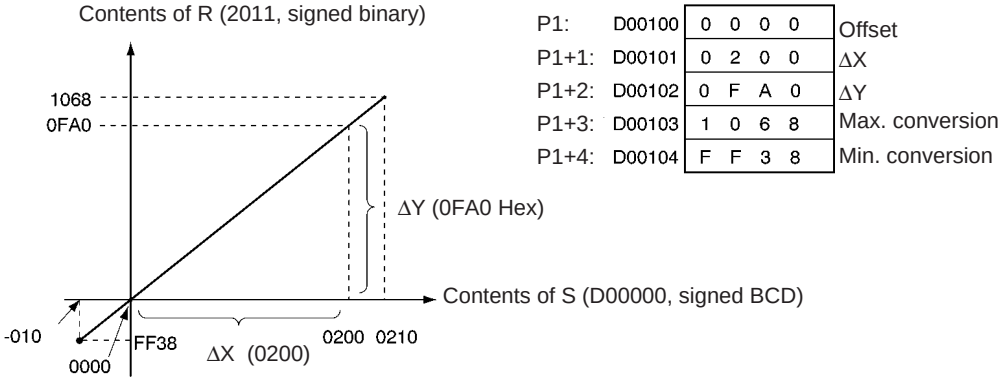
The Negative Flag will turn ON if the MSB of the result in R is 1, i.e., if the result is negative.

Examples

When a value from 0 to 200 is scaled to an analog signal (1 to 5 V, for example), a signed BCD value of 0000 to 0200 is converted (scaled) to signed

binary value of 0000 to 0FA0 for an Analog Output Unit. When CIO 000000 turns ON in the following example, the contents of D00000 is scaled using the linear function defined by ΔX (0200), ΔY (0FA0), and the offset (0). These values are contained in D00100 to D00102. The sign of the BCD value in D00000 is indicated by the Carry Flag. The result is output to CIO 2011.



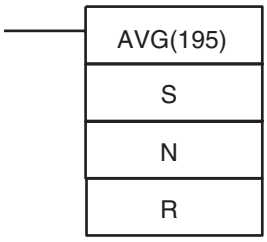


3-18-10 AVERAGE: AVG(195)

Purpose

Calculates the average value of an input word for the specified number of cycles.

Ladder Symbol



S: Source word

N: Number of cycles

R: Result word

R+1: First work area word

Variations

Variations		
	Executed Each Cycle for ON Condition	AVG(195)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

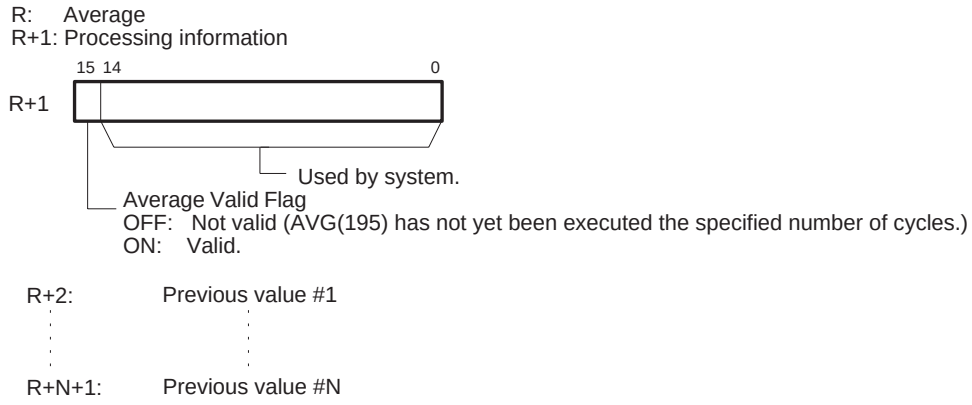
Operands

N: **Number of Cycles**

The number of cycles must be between 0001 and 0040 hexadecimal (0 to 64 cycles).

R: Result Word and R+1: First Work Area Word

R will contain the average value after the specified number of cycles. R+1 provides information on the averaging process and R+2 to R+N+1 contain the previous values of S as shown in the following diagram.



Note R to R+N+1 must be in the same area.

Operand Specifications

Area	S	N	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	#0001 to #0040 (binary)	---
Data Registers	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15		

Description

For the first N–1 cycles when the execution condition is ON, AVG(195) writes the values of S in order to words starting with R+2. The Previous Value Pointer (bits 00 to 07 of R+1) is incremented each time a value is written. Until the Nth value is written, the contents of S will be output unchanged to R and the Average Value Flag (bit 15 of R+1) will remain OFF.

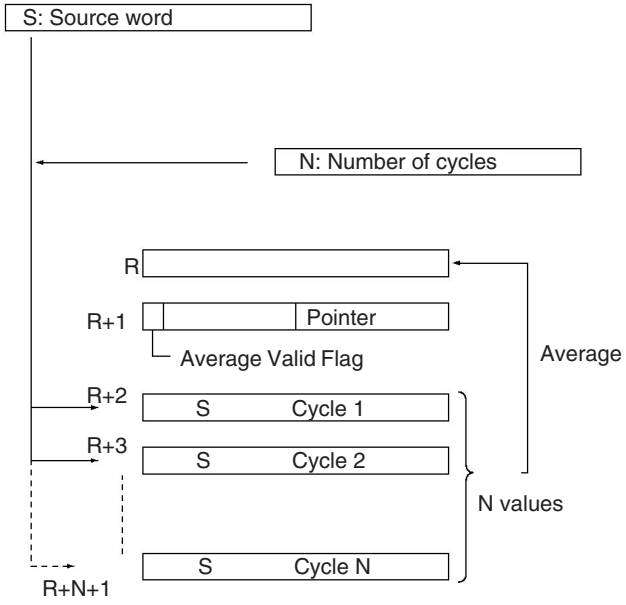
When the Nth value is written to R+N+1, the average of all the values that have been stored will be computed, the average will be output to R as an unsigned binary value, and the Average Value Flag (bit 15 of R+1) will be

turned ON. For all further cycles, the value in R will be updated for the most current N values of S.

The maximum value of N is 64. 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.



Flags

Name	Label	Operation
Error Flag	ER	ON if the contents of N is 0. OFF in all other cases.

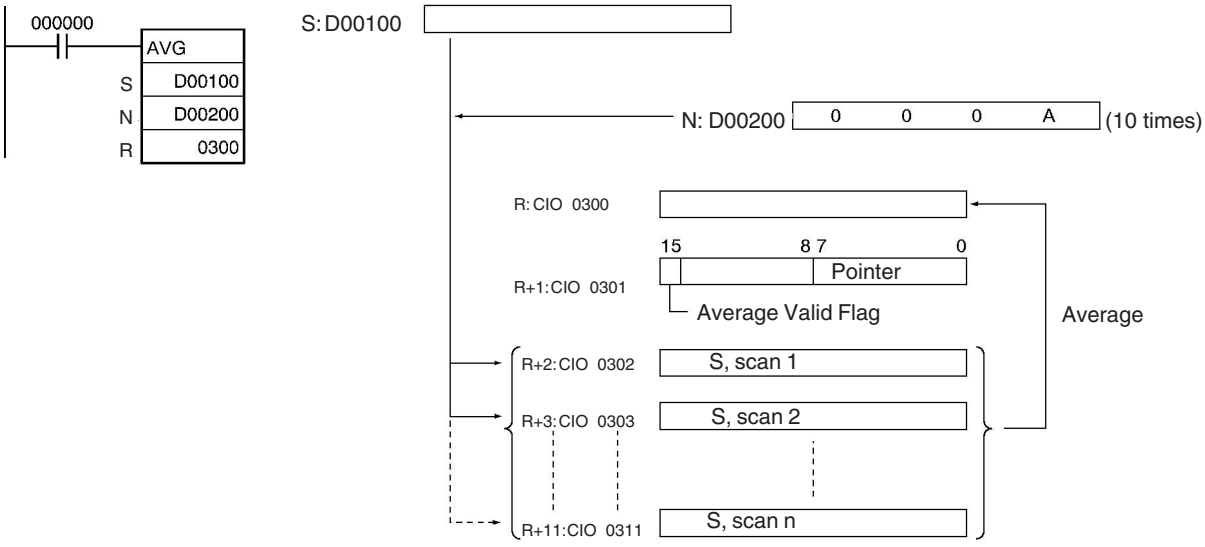
Precautions

The contents of the First Work Area Word (D+1) is cleared to 0000 each time the execution condition changes from OFF to ON.

The contents of the First Work Area Word (D+1) will not be cleared to 0000 the first time the program is executed at the start of operation. If AVG(195) is to be executed in the first program scan, clear the First Work Area Word from the program.

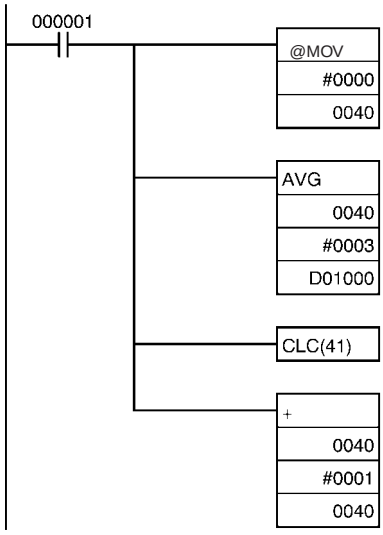
If N (Number of Cycles) contains 0000, an error will occur and the Error Flag will turn ON.

When CIO 000000 is ON in the following example, the contents of D00100 will be stored one time each scan for the number of scans specified in D00200. The contents will be stored in order in the ten words from CIO 0302 to CIO 0311. The average of the contents of these ten words will be placed in CIO 0300 and then bit 15 of CIO 0301 will be turned ON.



Examples

In the following example, the content of CIO 0040 is set to #0000 and then incremented by 1 each cycle. For the first two cycles, AVG(195) moves the content of CIO 0040 to D01002 and D01003. The contents of D01001 will also change (which can be used to confirm that the results of AVG(195) has changed). On the third and later cycles AVG(195) calculates the average value of the contents of D01002 to D01004 and writes that average value to D01000.



	1st cycle	2nd cycle	3rd cycle	4th cycle
CIO 0040	0000	0001	0002	0003

D01000	0000	0001	0001	0002	Average
D01001	0001	0002	8000	8001	Pointer
D01002	0000	0000	0000	0003	3 previous values of IR 40
D01003	---	0001	0001	0001	
D01004	---	---	0002	0002	

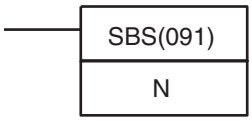
3-19 Subroutines

3-19-1 SUBROUTINE CALL: SBS(091)

Purpose

Calls the subroutine with the specified subroutine number and executes that program.

Ladder Symbol



N: Subroutine number

Variations

Variations	**Executed Each Cycle for ON Condition**	SBS(091)
	Executed Once for Upward Differentiation	@SBS(091)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Subroutine number
Specifies the subroutine number between 0 and 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

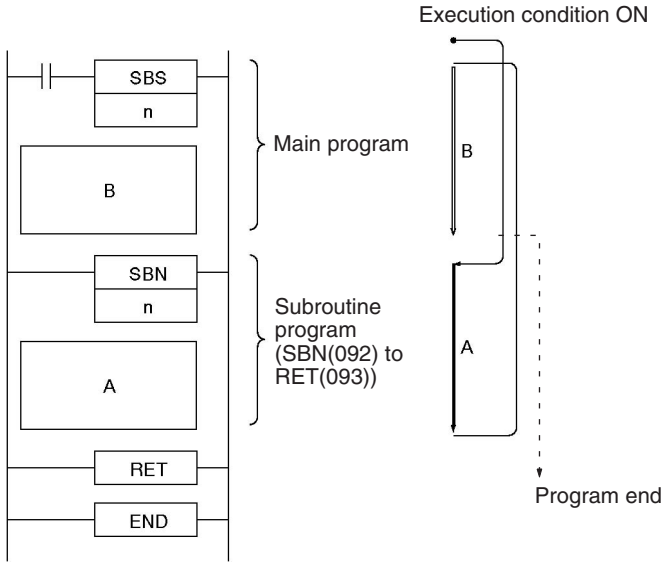
Area	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	0 to 1023 (decimal) (See note.)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the range is &0 to &255 decimal.

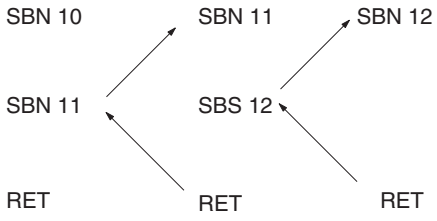
Description

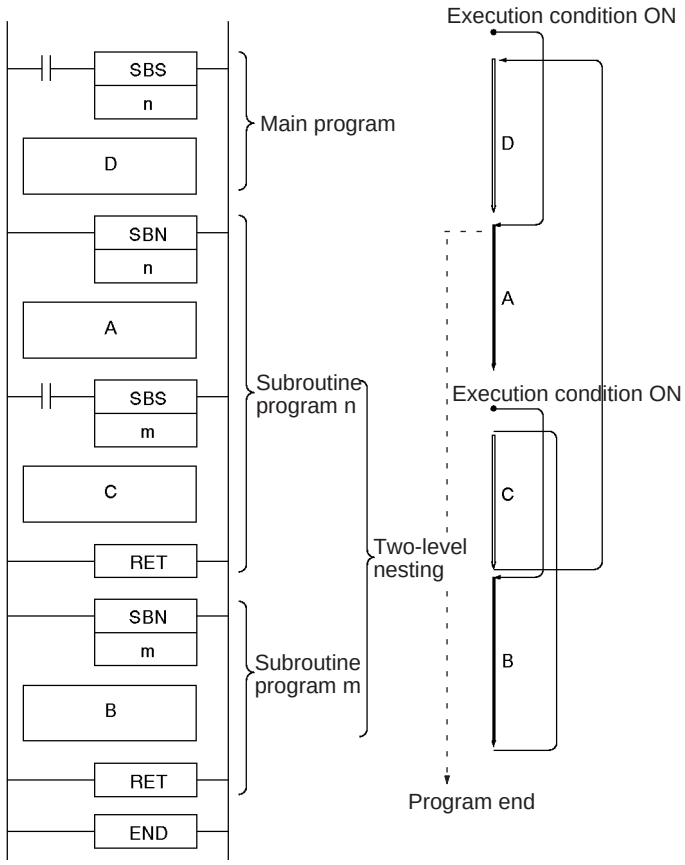
SBS(091) calls the subroutine with the specified subroutine number. The subroutine is the program section between SBN(092) and RET(093). When the

subroutine is completed, program execution continues with the next instruction after SBS(091).



Subroutines can be nested up to 16 levels. Nesting is when another subroutine is called from within a subroutine program, such as shown in the following example, which is nested to 3 levels.



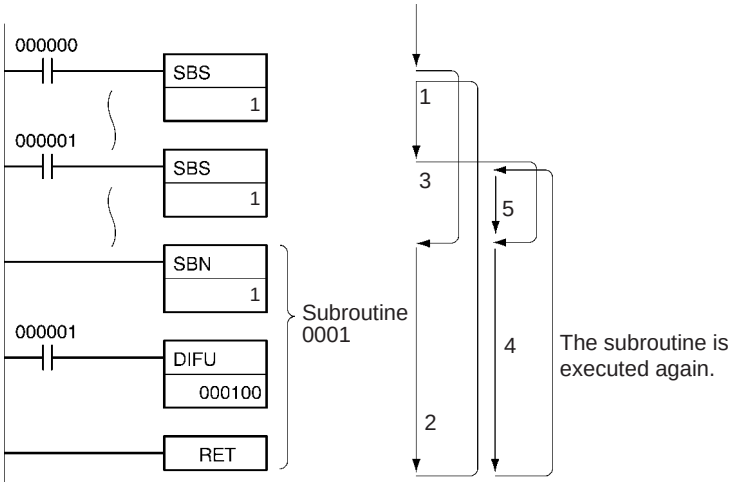


Note A subroutine can be called more than once in a program.

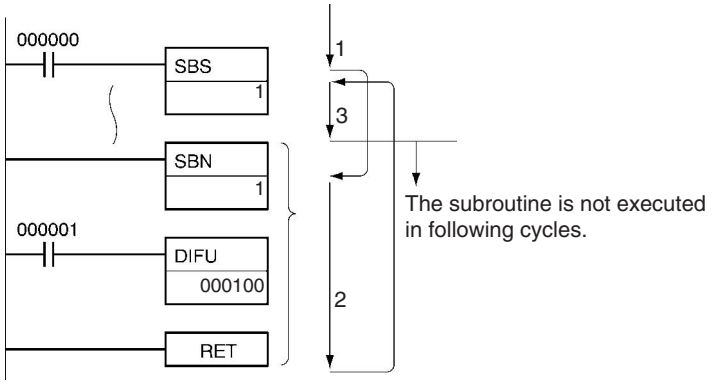
Subroutines and Differentiation

Observe the following precautions when using differentiated instructions (DIFU(013), DIFU(014), or up/down differentiated instructions) in subroutines.

The operation of differentiated instructions in a subroutine is unpredictable if a subroutine is executed more than once in the same cycle. In the following example, subroutine 0001 is executed when CIO 000000 is ON and CIO 000100 is turned ON by DIFU(013) when CIO 000001 has gone from OFF to ON. If CIO 000001 is ON in the same cycle, subroutine 0001 will be executed again but this time DIFU(013) will turn CIO 000100 OFF without checking the status of CIO 000001.



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 0001 is executed if CIO 000000 is ON. Output CIO 000100 is turned ON by DIFU(013) when CIO 000001 has gone from OFF to ON. If CIO 000000 is OFF in the following cycle, subroutine 0001 will not be executed again and output CIO 000100 will remain ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if nesting exceeds 16 levels. ON if the specified subroutine number does not exist. ON if a subroutine calls itself. ON if a subroutine being executed is called. ON if the specified subroutine is not defined in the current task. OFF in all other cases.

Precautions

Each subroutine must have a unique subroutine number. Do not use the same subroutine number for more than one subroutine.

SBS(091) and the corresponding SBN(092) must be programmed in the same task. An error will occur if the corresponding SBN(092) is not in the task.

SBS(091) will be treated as NOP(000) when it is within a program section interlocked by IL(002) and ILC(003).

When SBS(091) is executed in the following cases, the subroutine will not actually be called and the Error Flag will be turned ON:

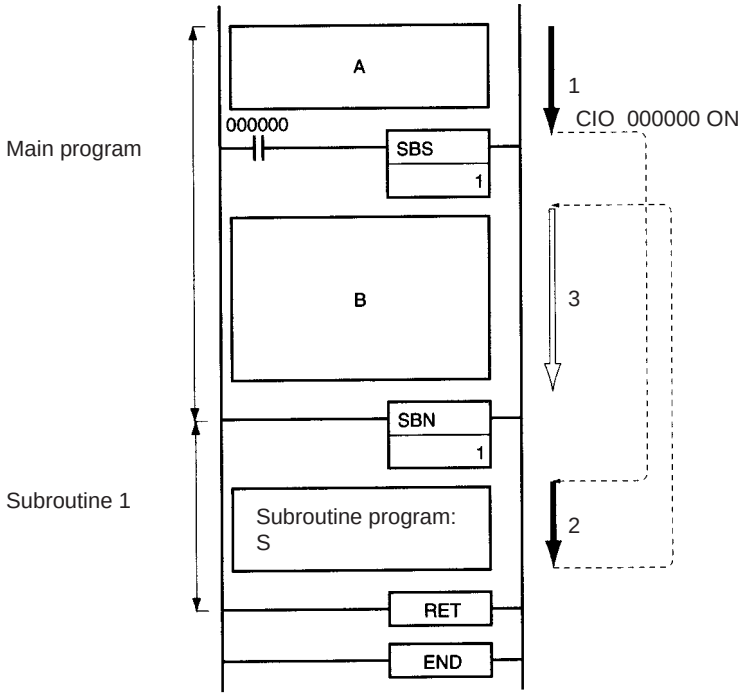
1,2,3...

1. The specified subroutine is not defined within the current task.
2. The subroutine is calling itself.
3. Subroutine nesting exceeds 16 levels.
4. The specified subroutine is being executed.

Examples

Example 1: Sequential (Non-nested) Subroutines

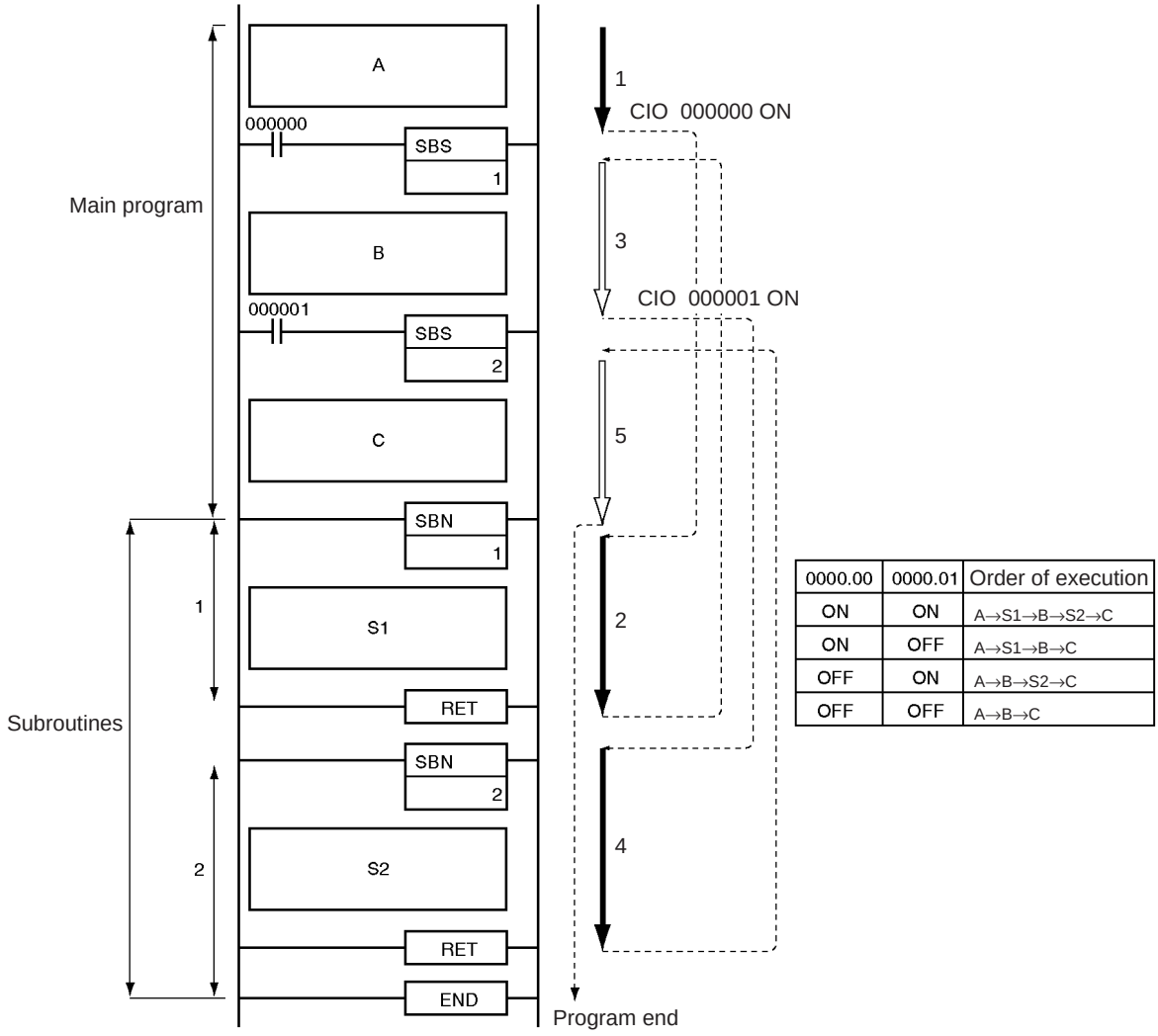
When CIO 000000 is ON in the following example, subroutine 1 is executed and program execution returns to the next instruction after SBS(091). The remainder of the main program (through the instruction just before SBN(092) 1) is then executed.



0000.00	Order of execution
ON	A→S→B
OFF	A→B

Example 2: Sequential (Non-nested) Subroutines

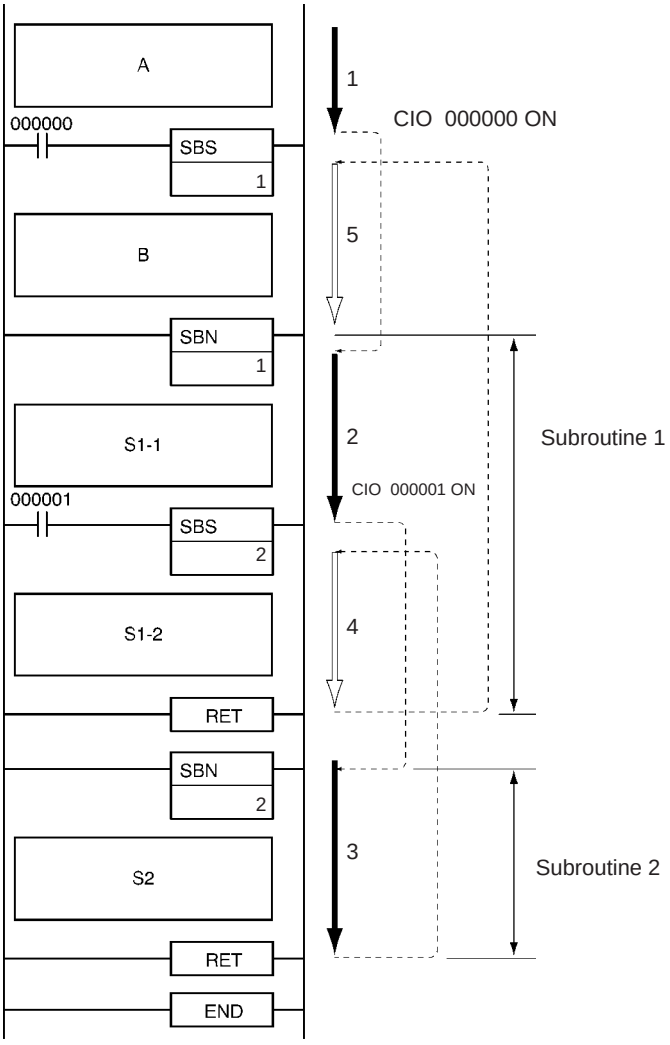
When CIO 000000 is ON in the following example, subroutine 1 is executed and program execution returns to the next instruction after SBS(091) 1. When CIO 000001 is ON, subroutine 2 is executed and program execution returns to the next instruction after SBS(091) 2.



0000.00	0000.01	Order of execution
ON	ON	A→S1→B→S2→C
ON	OFF	A→S1→B→C
OFF	ON	A→B→S2→C
OFF	OFF	A→B→C

Example 3: Nested Subroutines

When CIO 000000 is ON in the following example, subroutine 1 is executed. If CIO 000001 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.



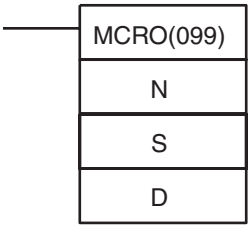
0000.00	0000.01	Order of execution
ON	ON	A→S1-1→S2→S1-2→B
ON	OFF	A→S1-1→S1-2→B
OFF	ON	A→B
OFF	OFF	A→B

3-19-2 MACRO: MCRO(099)

Purpose

Calls the subroutine with the specified subroutine number and executes that program using the input parameters in S to S+3 and the output parameters in D to D+3.

Ladder Symbol



N: Subroutine number

S: First input parameter word

D: First output parameter word

Variations

Variations		
	Executed Each Cycle for ON Condition	MCRO(099)
	Executed Once for Upward Differentiation	@MCRO(099)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Subroutine number
Specifies the subroutine number between 0 and 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

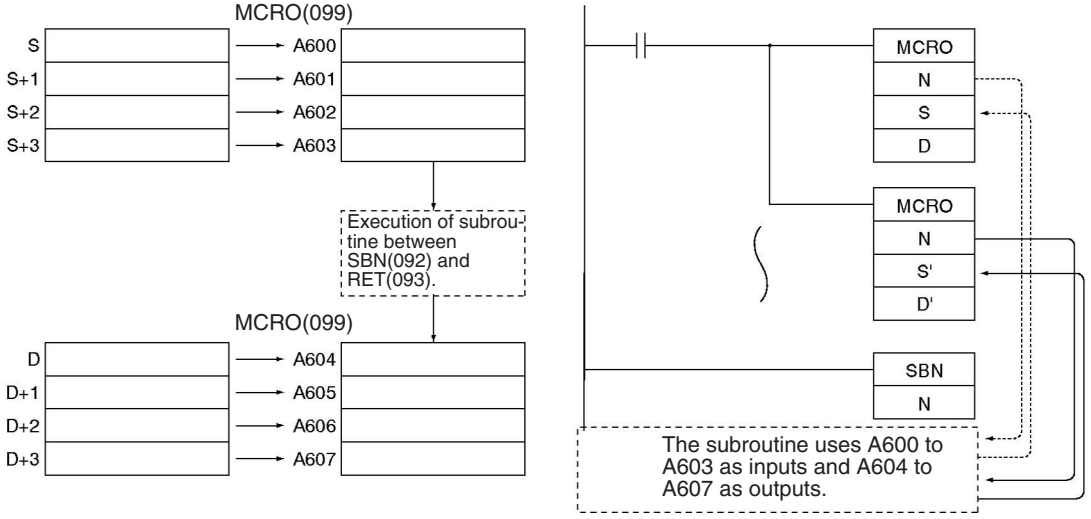
Area	N	S	D
CIO Area	---	CIO 0000 to CIO 6140	
Work Area	---	W000 to W508	
Holding Bit Area	---	H000 to H508	
Auxiliary Bit Area	---	A000 to A444 A448 to A956	A448 to A956
Timer Area	---	T0000 to T4092	
Counter Area	---	C0000 to C4092	
DM Area	---	D00000 to D32764	
EM Area without bank	---	E00000 to E32764	
EM Area with bank	---	En_00000 to En_32764 (n = 0 to C)	
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	0 to 1023 (decimal) (See note.)	---	
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15, IR0+(++) to IR015+(++) ,–(– –)IR0 to, –(– –)IR15	

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the range is 0 to 255 decimal.

Description

MCRO(099) calls the subroutine with the specified subroutine number just like SBS(091). Unlike SBS(091), MCRO(099) operands S and D can be used to change bit and word addresses in the subroutine, although the structure of the subroutine is constant.

When MCRO(099) is executed, the contents of S through S+3 are copied to A600 through A603 (macro area inputs) and the specified subroutine is executed. When the subroutine is completed, the contents of A604 through A607 (macro area outputs) are copied to D through D+3 and program execution continues with the next instruction after MCRO(099).



MCRO(099) can be used to consolidate two or more subroutines with the same structure but different input and output addresses into a single subroutine program. When MCRO(099) is executed, the specified input and output data is transferred to the specified subroutine.

Flags

Name	Label	Operation
Error Flag	ER	ON if nesting exceeds 16 levels. ON if the specified subroutine number does not exist. ON if a subroutine calls itself. ON if a subroutine being executed is called. ON if the specified subroutine is not defined in the current task. OFF in all other cases.

The following table shows relevant words in the Auxiliary Area.

Name	Address	Operation
Macro area input words	A600 to A603	When MCRO(099) is executed the four words from S to S+3 are copied to A600 to A603. These input words are passed to the subroutine.
Macro area input words	A604 to A607	After the subroutine specified in MCRO(099) has been executed, the output data in these output words and copied to D to D+3.

Precautions

The four words of input data (words or bits) in A600 to A603 and the four words of output data (words or bits) in A604 to A607 must be used in the subroutine called by MCRO(099). It is not possible to pass more than four words of data.

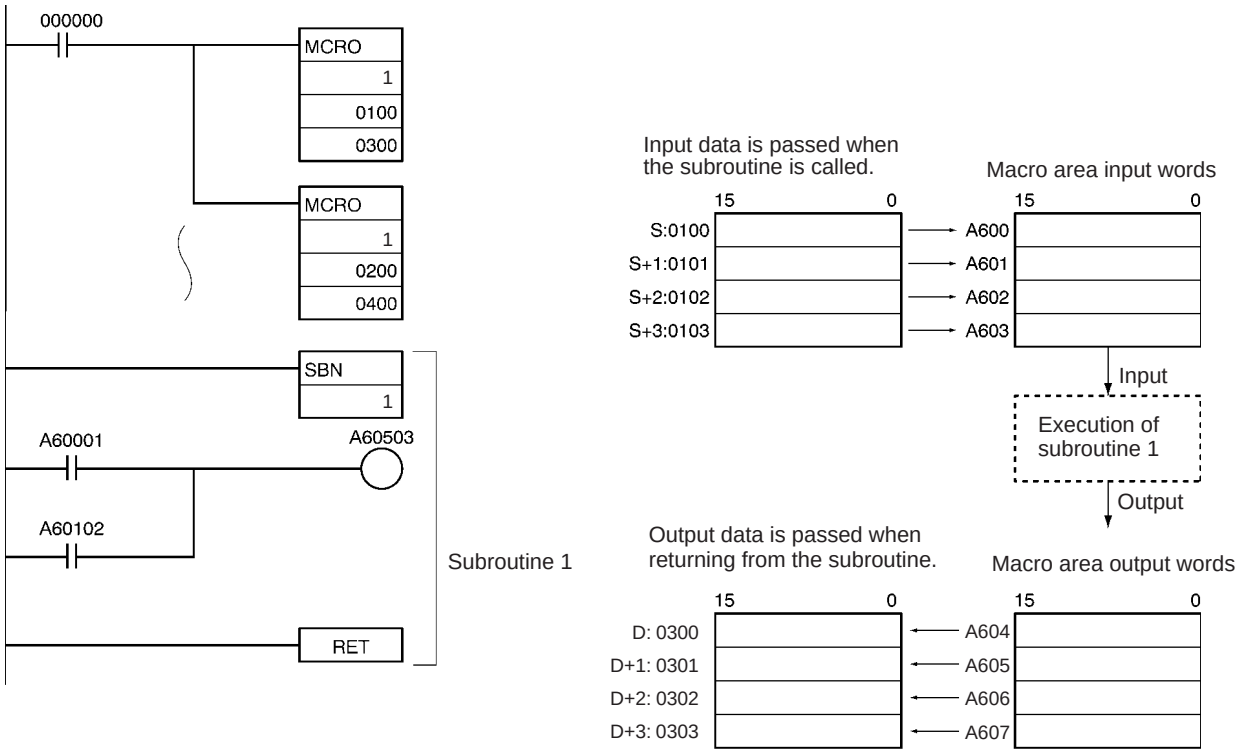
It is possible to nest MCRO(099) instructions, but the data in the macro area input and output words (A600 to A607) must be saved before calling another subroutine because all MCRO(099) instructions use the same 8 words.

Example

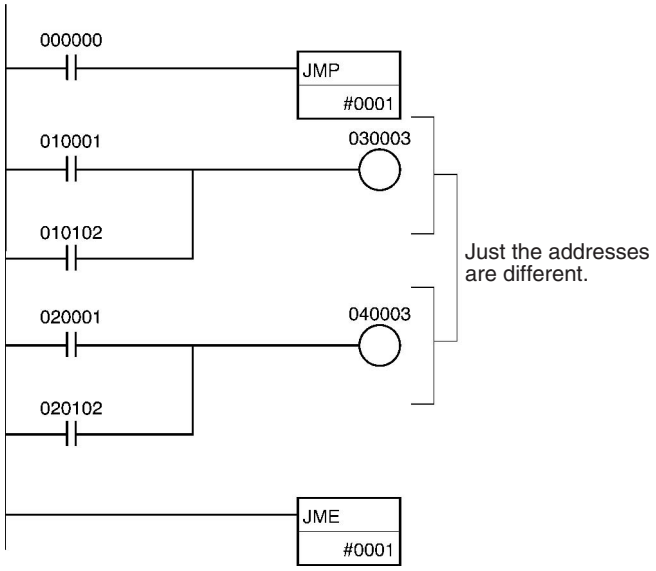
When CIO 000000 is ON in the following example, two MCRO(099) instructions pass different input and output data to subroutine 1.

1,2,3... 1. The first MCRO(099) instruction passes the input data in CIO 0100 to CIO 0103 and executes the subroutine. When the subroutine is completed, the output data is stored in CIO 0300 to CIO 0303.

2. The second MCRO(099) instruction passes the input data in CIO 0200 to CIO 0203 and executes the subroutine. When the subroutine is completed, the output data is stored in CIO 0400 to CIO 0403.



The second MCRO(099) instruction operates in the same way, but the input data in CIO 0200 to CIO 0203 is passed to A600 to A603 and the output data in A604 to A607 is passed to CIO 0400 to CIO 0403.

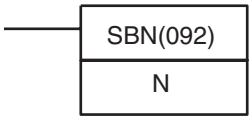


3-19-3 SUBROUTINE ENTRY: SBN(092)

Purpose

Indicates the beginning of the subroutine program with the specified subroutine number. Used in combination with RET(093) to define a subroutine region.

Ladder Symbol



N: Subroutine number

Variations

Variations	Executed Each Cycle for ON Condition	SBN(092)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Subroutine number
Specifies the subroutine number between 0 and 1023 decimal.

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the subroutine number must be between the range 0 to 255 decimal.

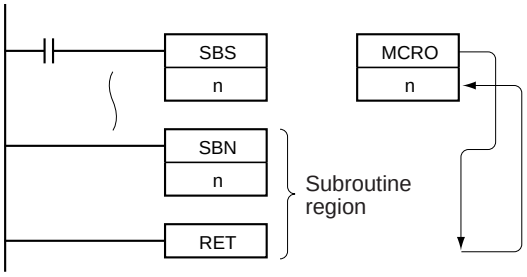
Operand Specifications

Area	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	0 to 1023 (decimal)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Description

SBN(092) indicates the beginning of the subroutine with the specified subroutine number. The end of the subroutine is indicated by RET(093).

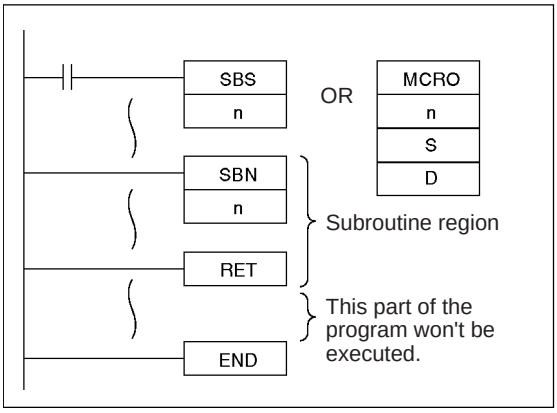
The region of the program beginning at the first SBN(092) instruction is the subroutine region. A subroutine is executed only when it has been called by SBS(091) or MCRO(099).



Precautions

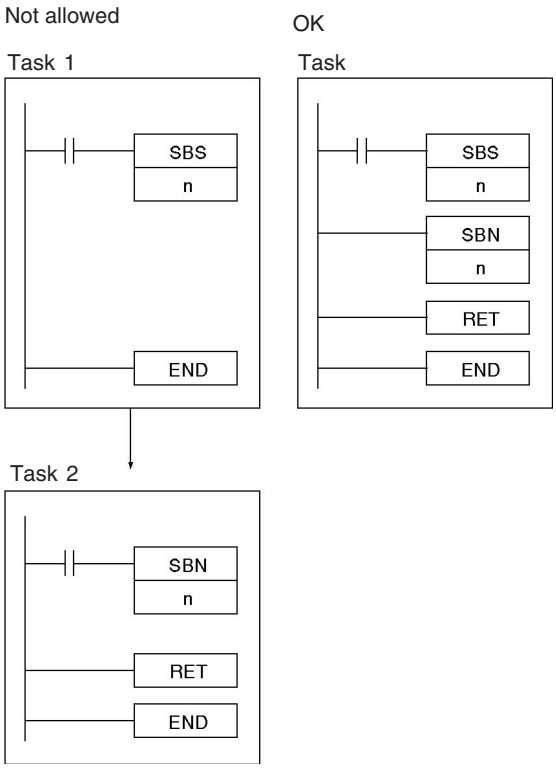
When the subroutine is not being executed, the instructions are treated as NOP(000).

Place the subroutines after the main program and just before the END(001) instruction in the program for each task. If part of the main program is placed after the subroutine region, that program section will be ignored.

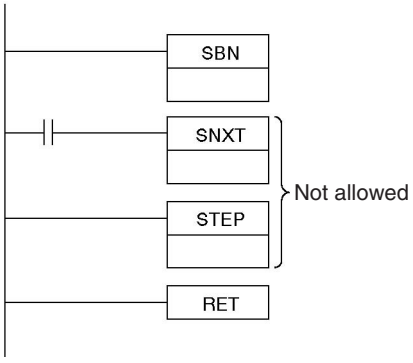


Note The input method for the subroutine number, N, is different for the CX-Programmer and a Programming Console. Input #0 to #1023 on the CX-Programmer and 0000 to 1023 on a Programming Console.

Be sure to place each subroutine in the same program (task) as its corresponding SBS(091) or MCRO(099) instruction. A subroutine in one task cannot be called from another task. It is possible to program a subroutine within an interrupt task.

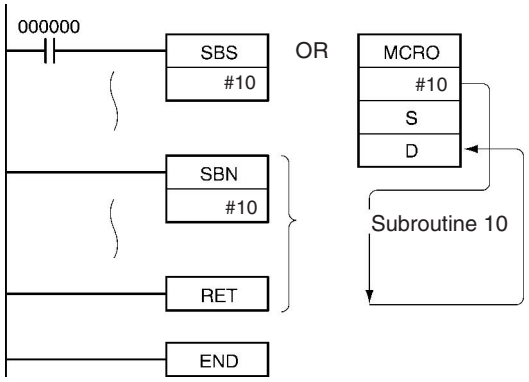


The step instructions, STEP(008) and SNXT(009) cannot be used in subroutines.



Example

When CIO 000000 is ON in the following example, subroutine 10 is executed and program execution returns to the next instruction after the SBS(091) or MCRO(099) instruction that called the subroutine.



3-19-4 SUBROUTINE RETURN: RET(093)

Purpose

Indicates the end of a subroutine program. Used in combination with SBN(092) to define a subroutine region.

Ladder Symbol

Variations

Variations	Executed Each Cycle for ON Condition	RET(093)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Description

RET(093) indicates the end of a subroutine and SBN(092) indicates the beginning of a subroutine. See *3-19-3 SUBROUTINE ENTRY: SBN(092)* for more details on the operation of subroutines.

When program execution reaches RET(093), it is automatically returned to the next instruction after the SBS(091) or MCRO(099) instruction that called the subroutine. When the subroutine has been called by MCRO(099), the output data in A604 through A607 is written to D through D+3 before program execution is returned.

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.

Precautions

When the subroutine is not being executed, the instructions are treated as NOP(000).

Example

See *3-19-3 SUBROUTINE ENTRY: SBN(092)* for examples of the operation of RET(093).

3-19-5 GLOBAL SUBROUTINE CALL: GSBS(750)

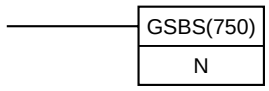
Purpose

Calls the global subroutine with the specified subroutine number and executes that program. The same global subroutine can be called from two or more tasks.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

GSBS(750) is used in combination with GSBN(751) and GRET(752), the GLOBAL SUBROUTINE ENTRY and GLOBAL SUBROUTINE RETURN instructions.

Ladder Symbol



N: Global subroutine number

Variations

Variations	Executed Each Cycle for ON Condition	GSBS(750)
	Executed Once for Upward Differentiation	@GSBS(750)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Global subroutine number
Specifies the global subroutine number between 0 and 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	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	0 to 1023 (decimal) (See note.)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the range is 0 to 255 decimal.

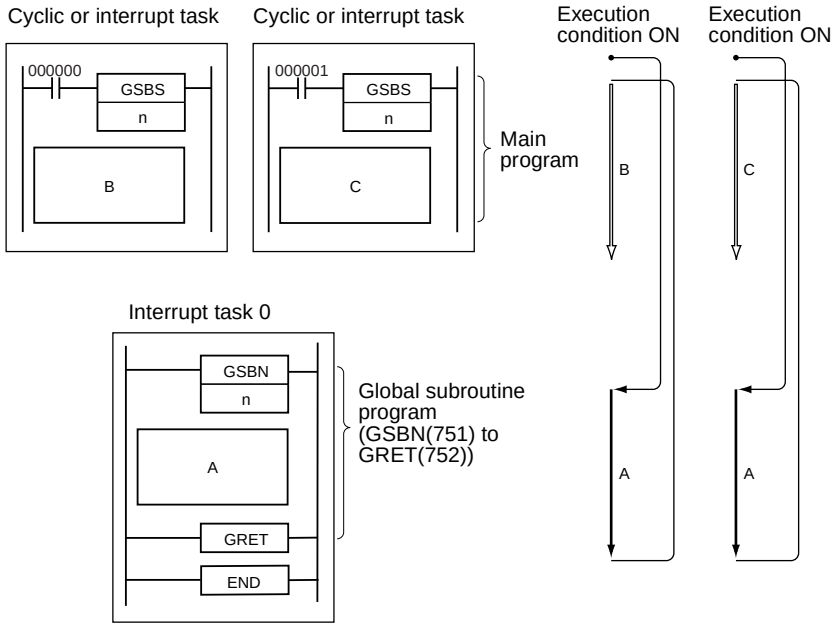
Description

GSBS(750) calls the global subroutine with the specified global subroutine number. The global subroutine is the program section between GSBN(751) and GRET(752). When the global subroutine is completed, program execution continues with the next instruction after GSBS(750).

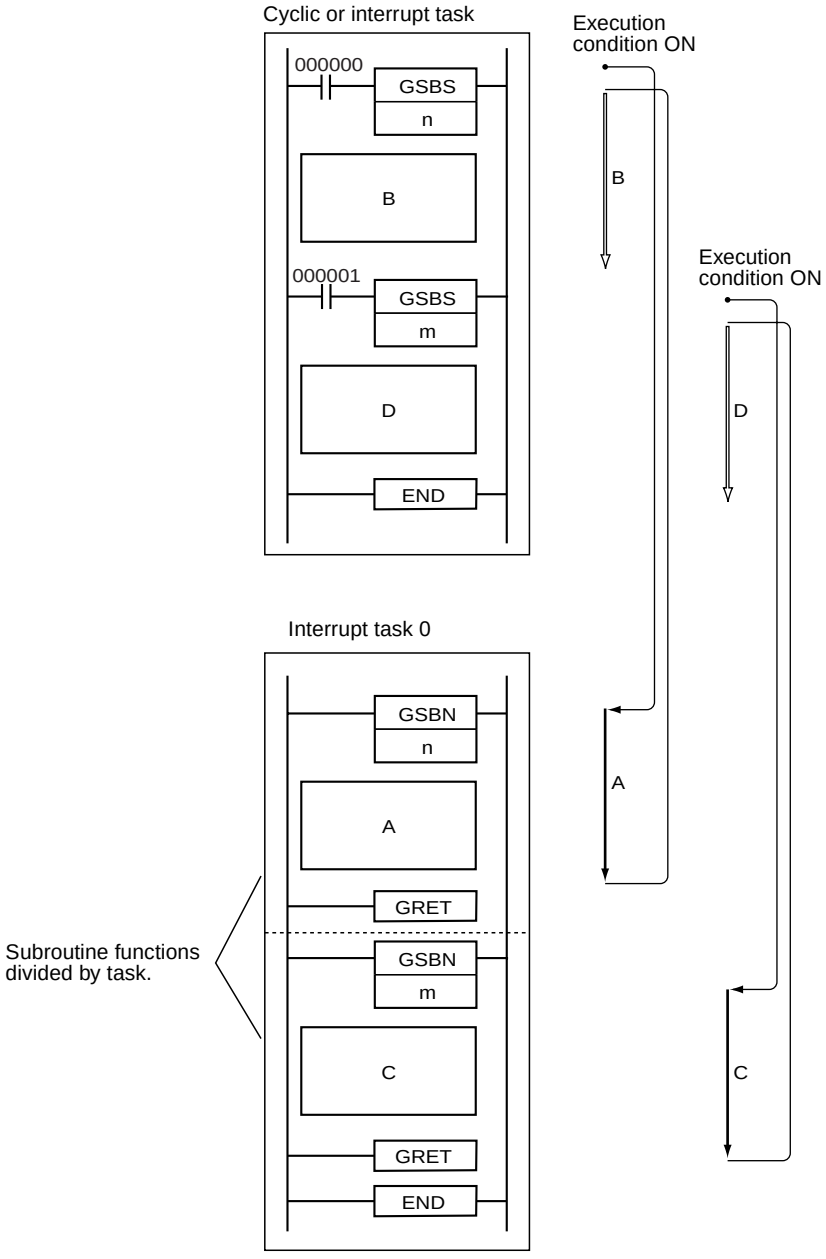
This instruction can be written into multiple tasks with the same global subroutine number to call that program from the different tasks. The program can be modularized by making global subroutines into standard subroutines that are common to many tasks.

The global subroutine region (between GSBN(751) and GRET(752)) must be defined in interrupt task 0. If it is defined in another task, an error will occur and the Error Flag will be turned ON when the GSBS(750) instruction is executed.

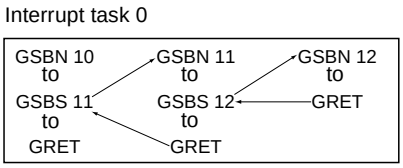
The GSBS(750) instruction can be written in both cyclic tasks (including extra cyclic tasks) and interrupt tasks.



Multiple global subroutine regions (GSBN(751) to GRET(752)) can be defined in interrupt task 0.



An SBS(091) or GSBS(750) instruction can be written within a subroutine region (SBN(092) to RET(093)) or global subroutine region (GSBN(751) to GRET(752)) to “nest” subroutines. Subroutines can be nested up to 16 levels.

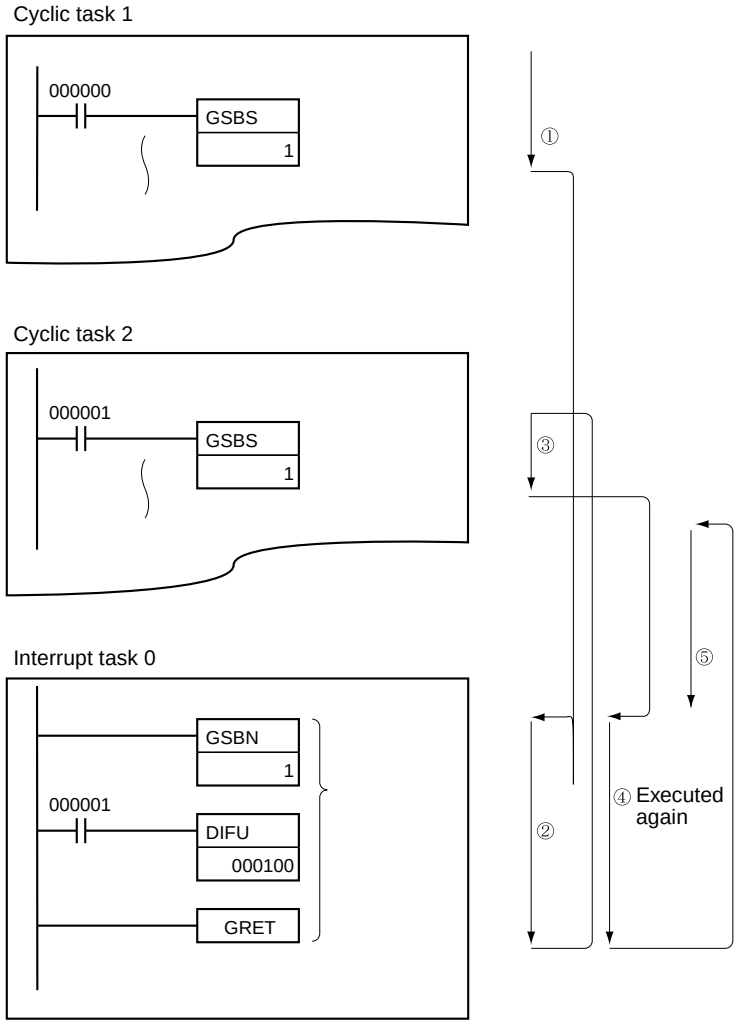


Global Subroutines and Differentiation

Observe the following precautions when using differentiated instructions (UP, DOWN, DIFU(013), DIFU(014), or up/down differentiated instructions) in subroutines.

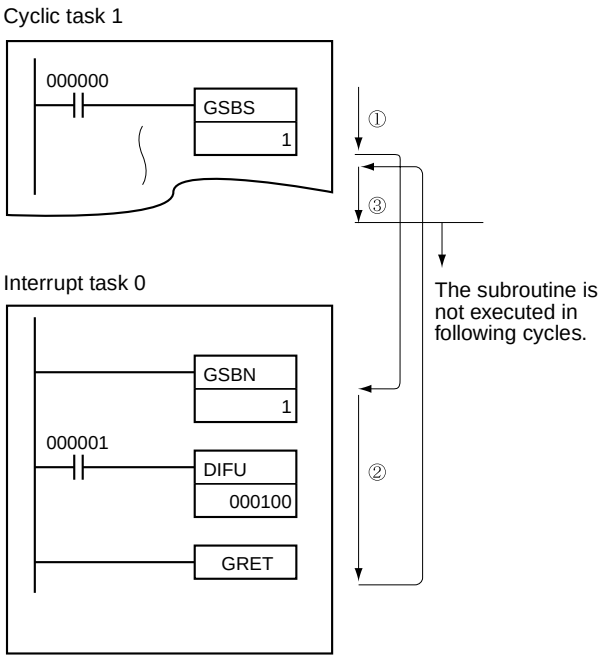
The operation of differentiated instructions in a global subroutine is unpredictable if a subroutine is executed more than once in the same cycle. In the following example, global subroutine 0001 is executed when CIO 000000 is ON

and CIO 000100 is turned ON by DIFU(013) when CIO 000001 has gone from OFF to ON. If CIO 000001 is ON in the same cycle, global subroutine 0001 will be executed again but this time DIFU(013) will not detect the rising edge of CIO 000001 and CIO 000100 will be turned OFF.



In contrast, the output of a differentiated instruction (DIFU(013) or DIFD(014)) would remain ON if the instruction was executed and the output was turned ON but the same global subroutine was not called a second time.

In the following example, global subroutine 0001 is executed if CIO 000000 is ON. Output CIO 000100 is turned ON by DIFU(013) when CIO 000001 has gone from OFF to ON. If CIO 000000 is OFF in the following cycle, subroutine 0001 will not be executed again and output CIO 000100 will remain ON.



Flags

Name	Label	Operation
Error Flag	ER	ON if nesting exceeds 16 levels (counting both regular and global subroutines). ON if the specified global subroutine does not exist. ON if a global subroutine calls itself. ON if a global subroutine being executed is called. ON if the specified subroutine is not defined in interrupt task 0. OFF in all other cases.

Precautions

The GLOBAL SUBROUTINE ENTRY instruction, GSBN(751), and the corresponding GLOBAL SUBROUTINE RETURN instruction, GRET(752) must be programmed in interrupt task 0. If the global subroutine region is not programmed in interrupt task 0, an error will occur and the Error Flag will be turned ON when the GSBS(750) instruction is executed.

The regular SUBROUTINE CALL instruction, SBS(091), cannot call a global subroutine region (GSBN(751) to GRET(752)).

GSBS(750) will not be executed when it is within a program section interlocked by IL(002) and ILC(003), so interlocks are not allowed within global subroutine regions.

The same global subroutine region (GSBN(751) to GRET(752)) can be called more than once.

When GSBS(750) is executed in the following cases, the global subroutine will not actually be called and the Error Flag will be turned ON:

1,2,3...

1. The specified global subroutine is not defined.
2. Subroutine nesting (counting both regular and global subroutines) exceeds 16 levels.
3. The global subroutine is calling itself.
4. The specified global subroutine is being executed.
5. The specified global subroutine is not defined in interrupt task 0.

Examples

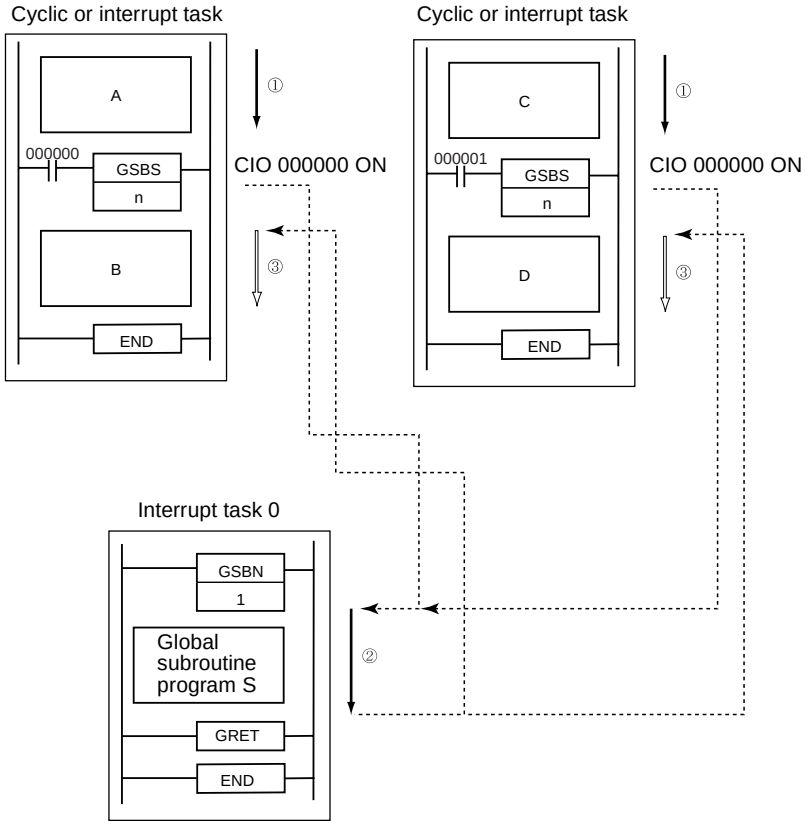
Example 1

When CIO 000000 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 000000	Order of program execution
ON	A → S → B
OFF	A → B

When CIO 000001 is ON in the following example, global subroutine 1 is executed and program execution returns to the next instruction after GSBS(750).

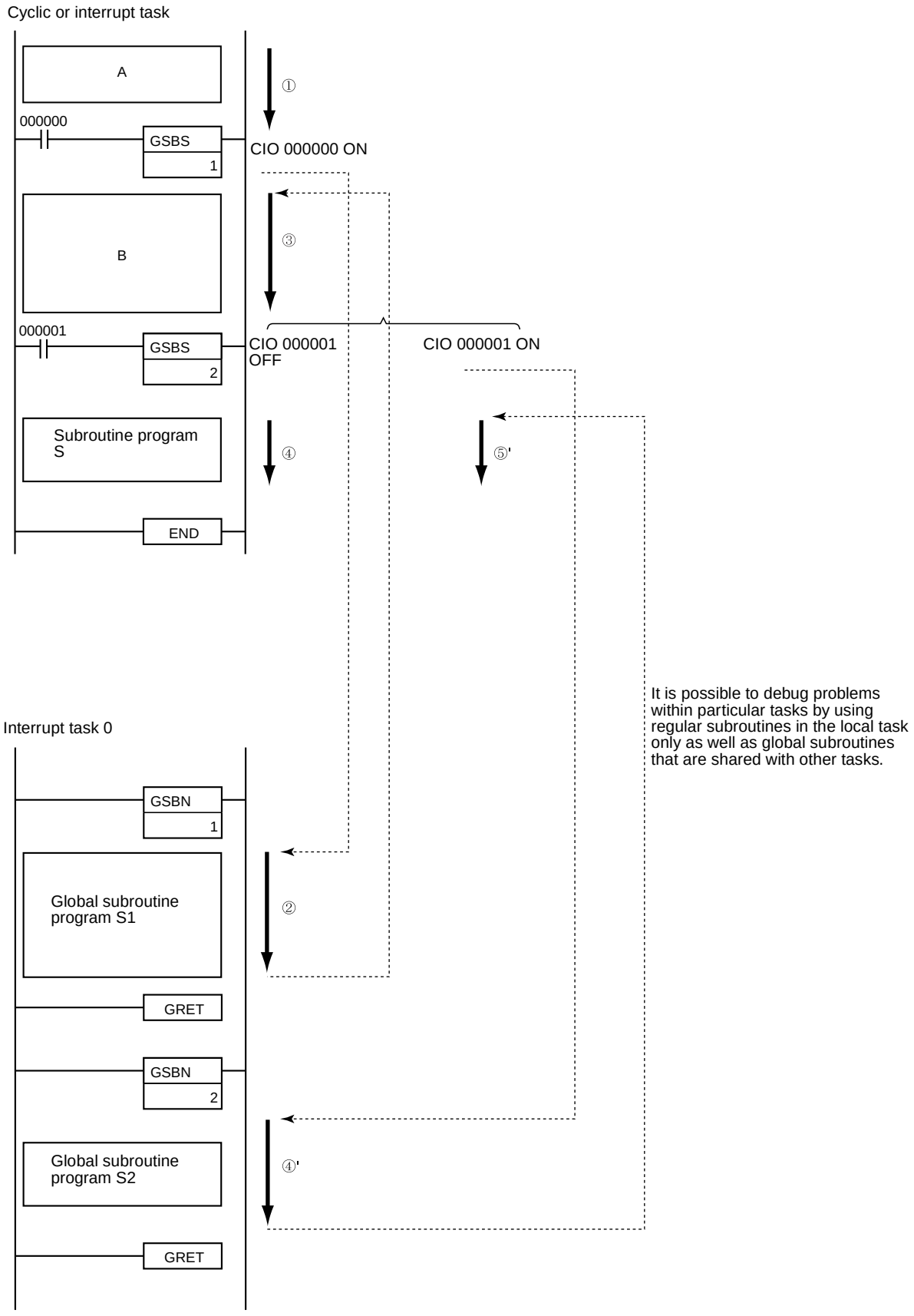
Status of CIO 000000	Order of program execution
ON	C → S → D
OFF	C → D



Example 2

Two or more global subroutine programs can be programmed in interrupt task 0. In this case, interrupt task 0 can be divided and used as the subroutine function's task.

When CIO 000000 is ON, global subroutine program 1 is executed.
When CIO 000001 is ON, global subroutine program 2 is executed.



3-19-6 GLOBAL SUBROUTINE ENTRY: GSBN(751)

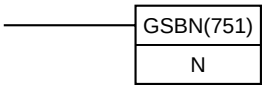
Purpose

Indicates the beginning of the global subroutine program with the specified subroutine number. Used in combination with GRET(752) to define a global subroutine region.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

GSBN(751) is used in combination with GSBS(750) and GRET(752), the GLOBAL SUBROUTINE CALL and GLOBAL SUBROUTINE RETURN instructions.

Ladder Symbol



Variations

Variations	Executed Each Cycle for ON Condition	GSBN(751)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Global subroutine number
Specifies the global subroutine number between 0 and 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	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	0 to 1023 (decimal) (See note.)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Note For CJ1M-CPU11 and CJ1M-CPU21 CPU Units, the range is 0 to 255 decimal.

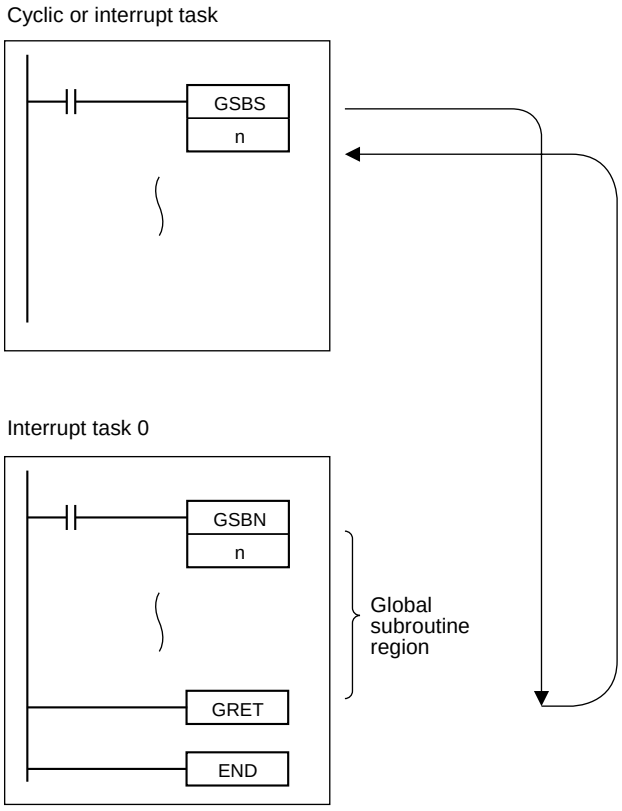
Description

GSBN(751) indicates the beginning of the global subroutine with the specified subroutine number. The end of the subroutine is indicated by GRET(752).

The region of the program beginning at the first GSBN(751) instruction is the subroutine region. A subroutine is executed only when it has been called by GSBS(750).

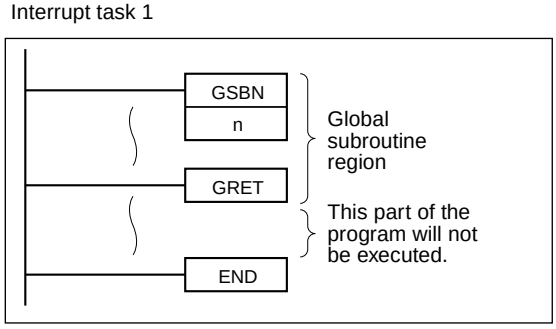
The global subroutine region (between GSBN(751) and GRET(752)) must be defined in interrupt task 0. If it is defined in another task, an error will occur and the Error Flag will be turned ON when the GSBS(750) instruction is executed.

The GSBS(750) instruction can be written both cyclic tasks (including extra cyclic tasks) and interrupt tasks.

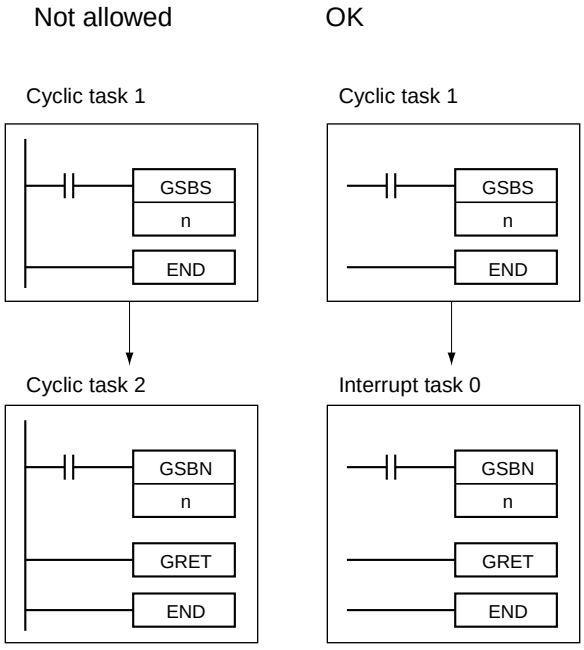


Precautions

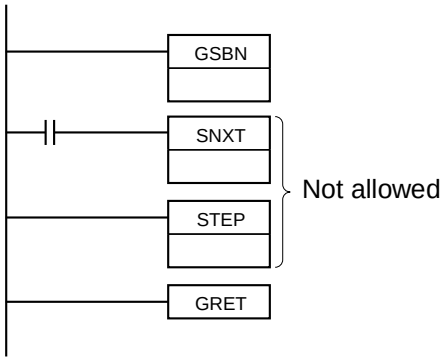
- When the subroutine is not being executed, the instructions are treated as NOP(000).
- Place the global subroutine region (GSBN(751) to GRET(752)) in interrupt task 0 just before the END(001) instruction. When two or more global subroutines are being used, group them together in interrupt task 0 after the end of the main program. If part of the main program is placed after the global subroutine region, that program section will be ignored.



- 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 0000 to 1023 on a Programming Console.
- Always place the global subroutines in interrupt task 0. An error will occur if a global subroutine is called and the subroutine is not in interrupt task 0.

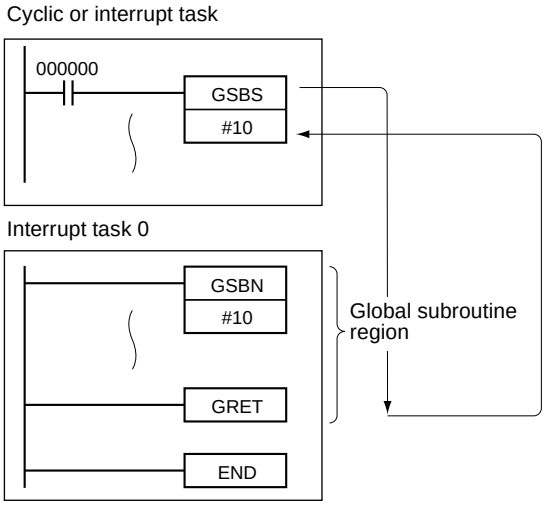


- The step instructions, STEP(008) and SNXT(009) cannot be used in global subroutines.



Example

When CIO 000000 is ON in the following example, global subroutine 10 is executed and program execution returns to the next instruction after the GSBS(750) instruction that called the subroutine.



3-19-7 GLOBAL SUBROUTINE RETURN: GRET(752)

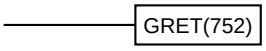
Purpose

Indicates the end of a subroutine program. Used in combination with GSBN(751) to define a subroutine region.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

GRET(752) is used in combination with GSBS(750) and GSBN(751), the GLOBAL SUBROUTINE CALL and GLOBAL SUBROUTINE ENTRY instructions.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	GRET(752)
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Description

GRET(752) indicates the end of a global subroutine and GSBN(751) indicates the beginning of a global subroutine. See *3-19-6 GLOBAL SUBROUTINE ENTRY: GSBN(751)* for more details on the operation of global subroutines.

When program execution reaches GRET(752) it is automatically returned to the next instruction after the GSBS(750) instruction that called the global subroutine.

Precautions

When the subroutine is not being executed, the instructions are treated as NOP(000).

Example

See *3-19-6 GLOBAL SUBROUTINE ENTRY: GSBN(751)* for examples of the operation of GRET(752).

3-20 Interrupt Control Instructions

The CS/CJ-series CPU Units support the following interrupts. For details, refer to the *SYSMAC CS/CJ/NSJ Series Programmable Controllers Programming Manual* (W394).

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)	None (always valid)

Outline of Interrupt Control Instructions

SET INTERRUPT MASK: MSKS(690)

Both I/O interrupt tasks and scheduled interrupt tasks are masked (disabled) when the PLC enters RUN mode. MSKS(690) can be used to unmask or mask I/O interrupts and set the time intervals for scheduled interrupts.

Note The power OFF interrupt is set in the PLC Setup.

CLEAR INTERRUPT: CLI(691)

CLI(691) clears or retains recorded interrupt inputs for I/O interrupts or sets the time to the first scheduled interrupt for scheduled interrupts. It also clears or retains recorded high-speed counter interrupts for CJ1M 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)

DI(693) disables execution of all interrupt tasks except the power OFF interrupt.

ENABLE INTERRUPTS: EI(694)

EI(694) enables execution of all interrupt tasks except the power OFF interrupt.

Precautions in Using Interrupt Tasks

Precautions for All Interrupts

When IORF(097), FIORF(225) (CJ1-H-R only), 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 (A40213) will be turned ON.

- I/O refreshing is performed for the same Special I/O Unit by IORF(097) or FIORF(225) (CJ1-H-R only).
- The same Special I/O Unit's data area is read by IORD(222) or written by IOWR(223).

Be sure that the interrupt task does not require more than 10 ms if a C200H Special I/O Unit 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.

Interrupts have different priority levels. A power OFF interrupt is given the highest priority, followed by I/O interrupts, external interrupts, and finally scheduled interrupts. Lower numbered I/O interrupts are given priority over a higher numbered I/O interrupts.

Precautions for I/O Interrupts

Only interrupt inputs from regular CS/CJ-series Interrupt Input Units and C200H Interrupt Input Units are supported for interrupt tasks. Interrupt inputs from Inner Boards and Special I/O Units are not supported.

Mount the Interrupt Input Unit in the CPU Rack. If a CJ1-H CPU Unit is being used, mount the Unit in slots 0 to 4, and if a CJ1M CPU Unit is being used, slots 0 to 2. It will not be possible to start the I/O interrupt task unless the Interrupt Input Unit is mounted in one of these slots.

Words are allocated to Interrupt Input Units in the order that they are mounted from left to right.

All interrupt inputs that have been detected will be cleared when the interrupt mask is cleared.

The CS1W-INT01 and the C200HS-INT01 cannot be used at the same time.

There is no limit on the number of I/O interrupt inputs that can be recorded, but only one interrupt is recorded for each I/O interrupt number. Furthermore, the recorded interrupt is not cleared until its interrupt task has been completed, so a new interrupt input will be ignored if it is received while its interrupt task is being executed.

Precautions for Scheduled Interrupts

Be sure that the 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 interrupt and the interrupt interval, program CLI(691) to set the time to the first schedule interrupt just before programming MSKS(690). If MSKS(690) is used to restart a schedule interrupt for a CJ1M 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.

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. A44115 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.

Name	Address	Operation
Interrupt Task Error Flag	A40213	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) (CJ1-H-R only), IORD(222), or IOWR(223) instruction executed in the interrupt task and an IORF(097), FIORF(225) (CJ1-H-R only), IORD(222), or IOWR(223) instruction executed in the cyclic task. • There is a conflict between an IORF(097), FIORF(225) (CJ1-H-R only), 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 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	A42615	Indicates whether Interrupt Task Error 1 or 2 occurred.
Interrupt Task Error Task Number	A42600 to A42611	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	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).	0: 10 ms (default) 1: 1.0 ms 2: 0.1 ms (See note.)

Note CJ1-H-R and CJ1M CPU Units only.

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.	0: OFF, 1: 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)

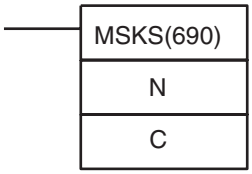
3-20-1 SET INTERRUPT MASK: MSKS(690)

Purpose

Controls whether I/O interrupt tasks and scheduled interrupt tasks are executed. When the program execution starts, the interrupt inputs that generate I/O interrupt tasks 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 and timer interrupts, so that the corresponding interrupt tasks can be executed.

Ladder Symbol



N: Interrupt identifier

C: Control data

Variations

Variations	**Executed Each Cycle for ON Condition**	MSKS(690)
	Executed Once for Upward Differentiation	@MSKS(690)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operands

■ **Disabling/Enabling an I/O Interrupt Task's Interrupt Input**

Inputs to a CS1W-INT01/CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)

Operand	Contents
N	Specify the Interrupt Input Unit's unit number. 0: Unit number 0 (interrupt tasks 100 to 115) 1: Unit number 1 (interrupt tasks 116 to 131)
C	Interrupt mask. 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)

Operand	Contents
N	Specify the Interrupt Input Unit's unit number. 0: Unit number 0 (interrupt tasks 100 to 107) 1: Unit number 1 (interrupt tasks 108 to 115) 2: Unit number 2 (interrupt tasks 116 to 123) 3: Unit number 3 (interrupt tasks 124 to 131)
C	Interrupt mask. 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 (masks) the interrupt.

Inputs to a CJ1M CPU Unit's Built-in Inputs (4 inputs/Unit)

Operand	Contents
N	Specify the interrupt input number. 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)
C	Interrupt mask. 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).

■ Specifying Up/Down Differentiation of an Interrupt Input (CS1W-INT01, CJ1W-INT01, and CJ1M CPU Unit Built-in Inputs Only)

Inputs to a CS1W-INT01/CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)

Operand	Contents
N	Specify the Interrupt Input Unit's unit number. 2: Unit number 0 (interrupt tasks 100 to 115) 3: Unit number 1 (interrupt tasks 116 to 131)
C	Specify either the rising or falling edge of the interrupt input signal. 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)

Operand	Contents
N	Specify the interrupt input number. 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)
C	Interrupt mask. 0000 hex: Up-differentiation (Detect rising edge.) 0001 hex: Down-differentiation (Detect falling edge.)

Note When the up/down differentiation setting is changed, all detected interrupt inputs will be cleared.

■ Disabling/Enabling a Scheduled Interrupt Task's Timer Interrupt

Operand	Contents	
N	Specify the scheduled interrupt number. 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.	
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 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.

■ **Resetting and Starting Scheduled Interrupts (CJ1M CPU Units Only)**

Operand	Contents	
N	Specify the scheduled interrupt number. 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.

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ 32767 @ E00000 to @ 32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	Specified values only	
Data Registers	---	DR0 to DR15

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

Description

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. N = 0 to 3: Enabling/Disabling the Interrupt Inputs of I/O Interrupt Tasks
   - 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 task is executed.
   - When an interrupt input is enabled, any interrupts detected up to that point will be cleared.
2. N = 6 to 13: Specifying the Differentiation of Interrupt Inputs
   - 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. N = 4 or 5: Specifying Timer Interrupts of Scheduled Interrupt Tasks
   - Sets the time interval (specified by C) for the specified scheduled interrupt task (specified by N) and starts the internal timer. The internal timer can also be stopped by setting C to 0. 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. N = 14 or 15: Resetting and Restarting Scheduled Interrupt Tasks
   - 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 (CJ1M 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 used to set the scheduled interrupt time is set as the Schedule Interrupt Interval in the PLC Setup.

Precautions

1. Be sure that the time interval is longer than the time required to execute the scheduled interrupt task.
2. 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 interrupt and the interrupt interval,

program CLI(691) to set the time to the first schedule interrupt just before programming MSKS(690). If MSKS(690) is used to restart a schedule interrupt for a CJ1M CPU Unit, however, the time to the first scheduled interrupt will be accurate even if CLI(691) is not used.

3. 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).

Related PLC Setup Settings

Scheduled Interrupts

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).	0:10 ms (default) 1: 1.0 ms 2: 0.1 ms (CJ1M and CJ1-H-R CPU Units only)

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0 to 5 (0 to 15 for the CJ1M CPU Unit's built-in interrupt inputs). Errors when specifying I/O Interrupts: • When using C200HS-INT01 interrupt inputs, the Error Flag will go ON if C is not between 0000 and 00FF hex. • When using the CJ1M 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: • 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). • 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). • When using a CJ1-H-R CPU Unit with the time units set to 0.1 ms, the Error Flag will go ON if C is not between 2 and 9,999 decimal (0002 to 270F hex). OFF in all other cases.
Equals Flag	=	OFF
Negative Flag	N	OFF

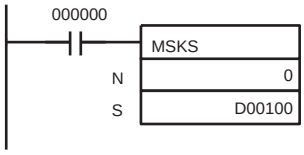
Related Auxiliary Area Flags and Words

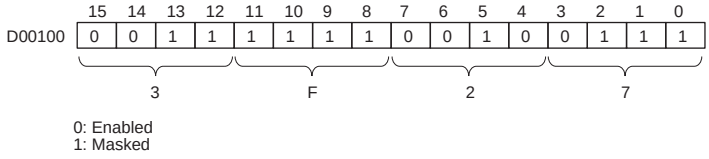
Name	Address	Operation
Interrupt Task Error Flag	A40213	ON in the following cases: 1. An interrupt task longer than 10 ms was executed during I/O refreshing with a C200H Special I/O Unit or Remote I/O Slave Rack. (CS Series only) 2. If Interrupt Task Error Detection is enabled in the PLC Setup, the Interrupt 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 (CJ1-H-R only), IORD, or IOWR instruction executed in the interrupt task and an IORF, FIORF (CJ1-H-R only), IORD, or IOWR instruction executed in the cyclic task. • There is a conflict between an IORF, FIORF (CJ1-H-R only), IORD, or IOWR instruction executed in the interrupt task and the CPU Unit's I/O refreshing (END refreshing). **Note** When Special I/O Unit Cyclic Refreshing is enabled in the PLC Setup, and an IORF, FIORF (CJ1-H-R only), IORD, or IOWR 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 Task Number	A42600 to A42611	Indicates the unit number of the Special I/O Unit where the simultaneous duplicate I/O refreshing occurred.

Operation Examples

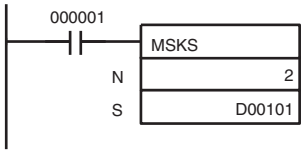
Examples for CS1W-INT01/CJ1W-INT01

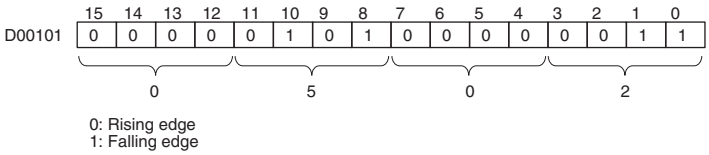
When CIO 000000 turns ON in the following example, MSKS(690) unmasks (enables) interrupt inputs in Interrupt Input Unit 0.





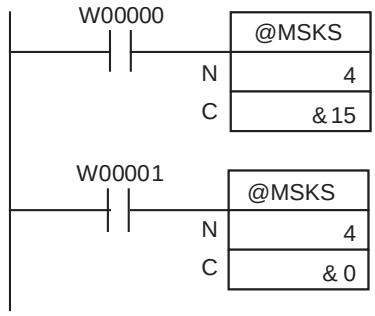
When CIO 000001 turns ON in the following example, MSKS(690) sets the rising/falling edge designations for Interrupt Input Unit 0.



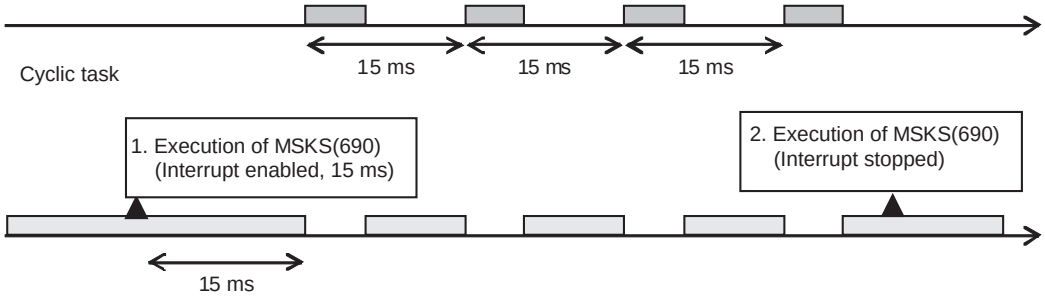


Example for Scheduled Interrupts

1. When W00000 goes from OFF to ON in the following example, MSKS(690) sets a 15-second 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 W00001 goes from OFF to ON, the internal timer is stopped for scheduled interrupt 0, which stops the generation of timer interrupts.



Scheduled interrupt task number 2

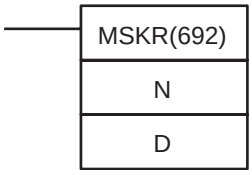


3-20-2 READ INTERRUPT MASK: MSKR(692)

Purpose

Reads the current interrupt control settings that were set with MSKS(690).

Ladder Symbol



N: Interrupt number

D: Destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	MSKR(692)
	Executed Once for Upward Differentiation	@MSKR(692)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operands

■ **Reading the Interrupt Mask Settings Set for I/O Interrupt Tasks**

Inputs to a CS1W-INT01/CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)

Operand	Contents
N	Specify the Interrupt Input Unit's unit number. 0: Unit number 0 (interrupt tasks 100 to 115) 1: Unit number 1 (interrupt tasks 116 to 131)
D	Range: 0000 to FFFF hex Bits 0 to 15 correspond to each interrupt task. The meaning of the individual flags is as follows: 0: Interrupt enabled (unmasked). 1: Interrupt disabled (masked).

Inputs to a C200HS-INT01 Interrupt Input Unit (8 inputs/Unit)

Operand	Contents
N	Specify the Interrupt Input Unit's unit number. 0: Unit number 0 (interrupt tasks 100 to 107) 1: Unit number 1 (interrupt tasks 108 to 115) 2: Unit number 2 (interrupt tasks 116 to 123) 3: Unit number 3 (interrupt tasks 124 to 131)
D	Range: 0000 to 00FF hex Bits 0 to 7 correspond to each interrupt task. The meaning of the individual flags is as follows: 0: Interrupt enabled (unmasked). 1: Interrupt disabled (masked).

Inputs to a CJ1M CPU Unit's Built-in Inputs (4 inputs/Unit)

Operand	Contents
N	Specify the interrupt input number. 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)
D	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.

■ Reading the Up/Down Differentiation Settings of I/O Interrupt Tasks (CS1W-INT01, CJ1W-INT01, and CJ1M CPU Unit Built-in Inputs Only)

Inputs to a CS1W-INT01/CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)

Operand	Contents
N	Specify the Interrupt Input Unit's unit number. 2: Unit number 0 (interrupt tasks 100 to 115) 3: Unit number 1 (interrupt tasks 116 to 131)
D	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 CPU Unit's Built-in Inputs (4 inputs/Unit)

Operand	Contents
N	Specify the interrupt input number. 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)
D	0000 hex: Up-differentiation (Detect rising edge.) 0001 hex: Down-differentiation (Detect falling edge.)

■ Reading the Set Value of a Scheduled Interrupt Task's Internal Timer

Operand	Contents	
N	Specify the scheduled interrupt number. 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.	
C	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.)

■ Reading the Present Value of a Scheduled Interrupt Task's Internal Timer (CJ1M CPU Units Only)

Operand	Contents
N	Specify the scheduled interrupt number. 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.

Operand	Contents	
C	Scheduled interrupt time units (Set in the PLC Setup.)	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	N	D
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A448 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	Specified values only	---
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15

Description

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. N = 0 to 3: Reading the Interrupt Mask Status of I/O Interrupt Tasks

   Reads the masked/unmasked status of the interrupt inputs specified by N, and outputs that information to the bits in D.

2. N = 6 to 13: Reading the Up/Down Differentiation of Interrupt Inputs

   Reads the up/down differentiation settings of the interrupt inputs specified by N, and outputs that information to the bits in D.

3. N = 4 or 5: Reading a Scheduled Interrupt Task's Time Interval

Reads the operating status of the internal timer of the scheduled interrupt task specified by N, and outputs that information to D. With this function, MSKR(692) can indicate whether the internal timer is stopped or operating, and indicate the interrupt time interval if it is operating.

4. N = 14 or 15: Reading a Scheduled Interrupt Task's Internal Timer PV

   Reads the internal timer PV of the scheduled interrupt task 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 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 used to set the scheduled interrupt time is set as the Schedule Interrupt Interval in the PLC Setup.

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0 to 5 (0 to 15 for the CJ1M). OFF in all other cases.

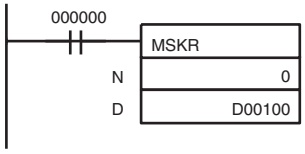
Precautions

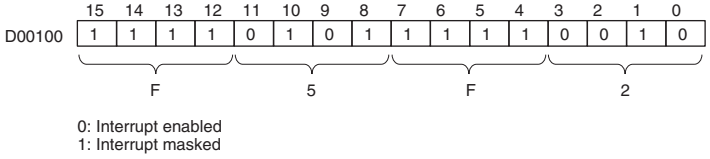
MSKR(692) can be executed in the main program or in interrupt tasks.

Operation Examples

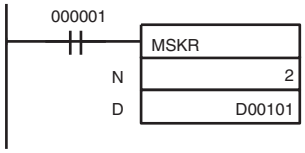
Example for CS1W-INT01/CJ1W-INT01

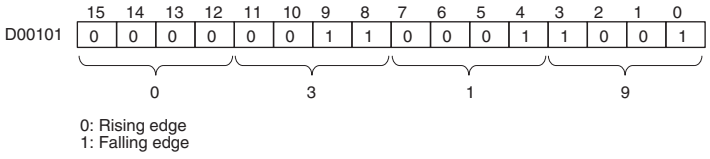
When CIO 000000 turns ON in the following example, MSKR(692) reads the current mask status of Interrupt Input Unit 2 and stores it in D00100.





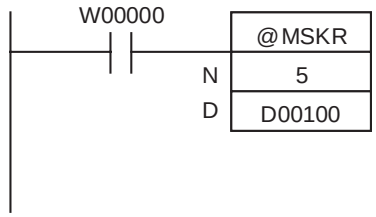
When CIO 000001 turns ON in the following example, MSKS(690) reads the rising/falling edge designations for Interrupt Input Unit 0 and stores it in D00101.



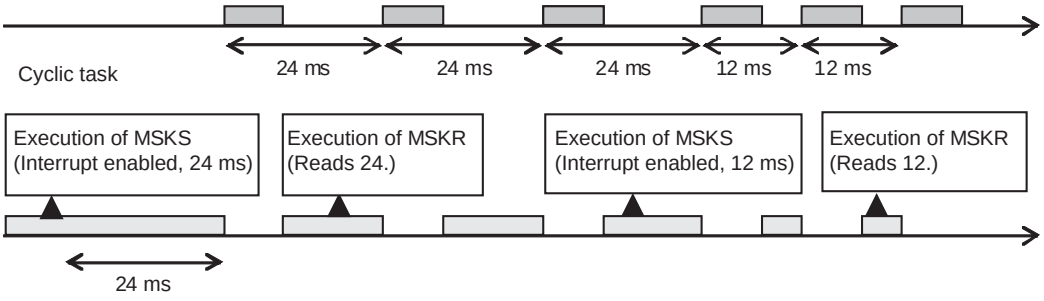


Example for Scheduled Interrupts

When W00000 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 D00100.



Scheduled interrupt task number 3

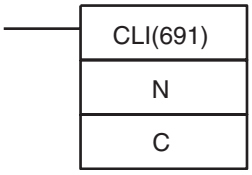


3-20-3 CLEAR INTERRUPT: CLI(691)

Purpose

Clears/retains recorded interrupt inputs, sets the time to the first scheduled interrupt for scheduled interrupt tasks, or clears/retains recorded high speed counter interrupts (CJ1M CPU Units only).

Ladder Symbol



N: Interrupt number

C: Control data

Variations

Variations	**Executed Each Cycle for ON Condition**	CLI(691)
	Executed Once for Upward Differentiation	@CLI(691)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operands

■ **Clearing/Retaining an I/O Interrupt Task's Recorded Interrupt Inputs**

Inputs to a CS1W-INT01/CJ1W-INT01 Interrupt Input Unit (16 inputs/Unit)

Operand	Contents
N	Specify the Interrupt Input Unit's unit number. 0: Unit number 0 (interrupt tasks 100 to 115) 1: Unit number 1 (interrupt tasks 116 to 131)
C	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)

Operand	Contents
N	Specify the Interrupt Input Unit's unit number. 0: Unit number 0 (interrupt tasks 100 to 107) 1: Unit number 1 (interrupt tasks 108 to 115) 2: Unit number 2 (interrupt tasks 116 to 123) 3: Unit number 3 (interrupt tasks 124 to 131)
C	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 CPU Unit's Built-in Inputs (4 inputs/Unit)

Operand	Contents
N	Specify the interrupt input number. 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)
C	0000 hex: Retain the recorded interrupt. 0001 hex: Clear the recorded interrupt.

■ **Setting the TIme to the First Scheduled Interrupts**

Operand	Contents	
N	Specify the scheduled interrupt number. 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.	
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.)

■ **Clearing/Retaining High-speed Counter Interrupts (CJ1M Only)**

Operand	Contents
N	Specify the high-speed counter input. 10: High-speed counter input 0 (interrupt task 2) 11: High-speed counter input 1 (interrupt task 3)
C	0000 hex: Retain the recorded interrupt. 0001 hex: Clear the recorded interrupt.

Operand Specifications

Area	N	C
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---	DR0 to DR15
Data Registers	Specified values only	
Index Registers	---	
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15

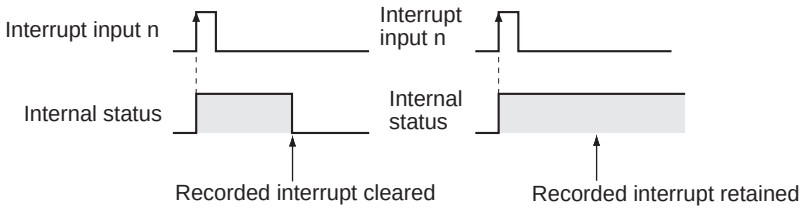
Description

Depending on the value of N, CLI(691) clears the specified recorded I/O interrupts, sets the time before execution of the first scheduled interrupt, or clears the specified recorded high-speed counter interrupts (CJM1 CPU Units only). With the CJ1M, it can also be used to clear interrupts for the high-speed counters.

N = 0 to 3, or 6 to 9: Clearing Interrupt Inputs

CLI(691) clears a recorded interrupt input specified by N, when the corre-

sponding bit of C is ON and retains the recorded interrupt input when the corresponding bit is OFF.

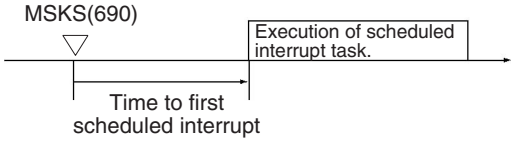


If an I/O interrupt task is being executed and an interrupt input with a different interrupt number is received, that interrupt number is recorded internally. The recorded I/O interrupts are executed later in order of their priority (from the lowest number to the highest).

If you want to ignore interrupt inputs that are received while an interrupt task is being executed, use CLI(691) to clear the recorded interrupts before they are executed.

N = 4 or 5: Setting the Time to the First Scheduled Interrupt Task

When N is 4 or 5, the content of C specifies the time interval to the first scheduled interrupt task.



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 tasks is set in the PLC Setup as the Scheduled Interrupt Interval.

■ N = 10 or 11: Clearing High-speed Counter Interrupts (CJ1M Only)

When N is 10 or 11, CLI(691) clears or retains the recorded high-speed counter interrupt (either target or range comparison) specified by N.

Flags

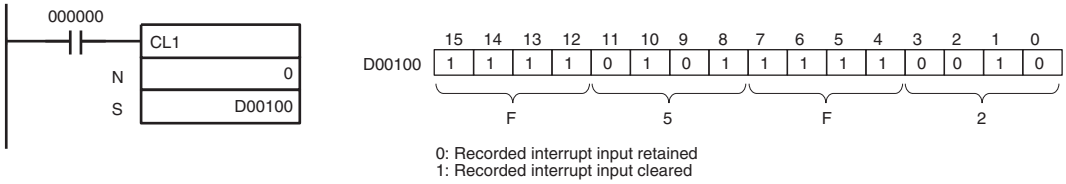
Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0 to 5 (0, 1, or 4 to 11 for CJ1M). ON if C is not within the specified range of 0000 to 00FF hex when N is 0 to 3 (for I/O interrupts and C200HS-INT only). ON if C is not 0000 or 0001 hex (for high-speed counter interrupts and CJ1M built-in interrupt inputs only). ON if C is not within the specified range of 0 to 9,999 decimal (0000 to 270F hex) for scheduled interrupts. OFF in all other cases.

Interrupts have different priority levels. A power OFF interrupt is given the highest priority, followed by I/O interrupts, external interrupts, and finally scheduled interrupts. Lower numbered I/O interrupts are given priority over a higher numbered I/O interrupts.

Operation Examples

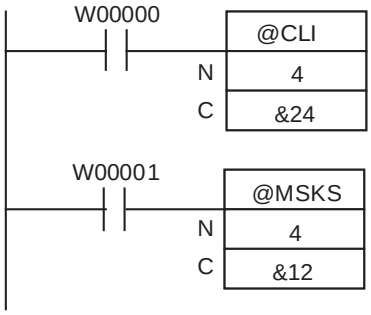
Example for CS1W-INT01/CJ1W-INT01

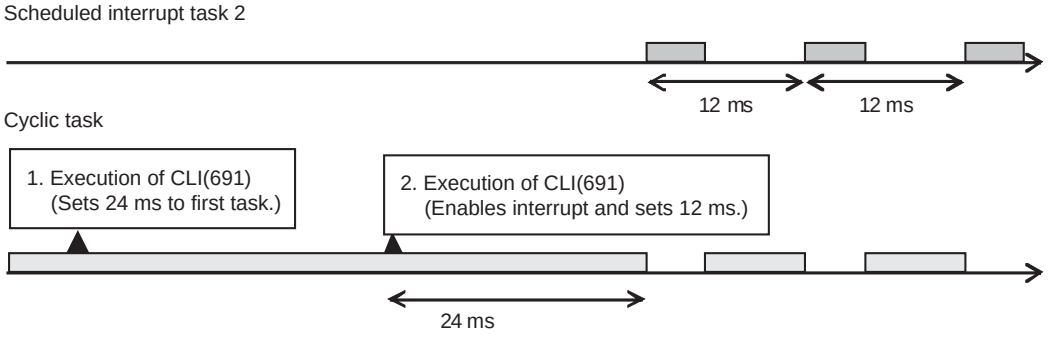
When CIO 000000 is ON in the following example, CLI(691) clears the recorded interrupts for the specified interrupt inputs in Interrupt Input Unit 0.



Setting the Time to the First Scheduled Interrupt

1. When W00000 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 time interval units are set to 1 ms in the PLC Setup.)
2. When W00001 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.)





3-20-4 DISABLE INTERRUPTS: DI(693)

Purpose

Disables execution of all interrupt tasks except the power OFF interrupt. When a CS1D CPU Unit for Single-CPU System or a CS1-H, CJ1-H, or CJ1M CPU Unit is being used and the power OFF interrupt task is disabled, it is possible to disable power OFF interrupt processing simultaneously.

Ladder Symbol

Variations

Variations	**Executed Each Cycle for ON Condition**	DI(693)
	Executed Once for Upward Differentiation	@DI(693)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Description

DI(693) is executed from the main program to temporarily disable all interrupt tasks except the power OFF interrupt (I/O interrupts, scheduled interrupts, and external interrupts).

All interrupt tasks will be disabled until they are enabled again by execution of EI(694).

CS1-H, CJ1-H, and CJ1M CPU Units and Power OFF Interrupts

When a CS1-H, CJ1-H, and CJ1M 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 information on the power OFF interrupt task in the *CS/CJ Series Programming Manual*.

Flags

Name	Label	Operation
Error Flag	ER	ON if DI(693) is executed from an interrupt task. OFF in all other cases.

Related Flags and Words

The following word is in the Auxiliary Area.

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.

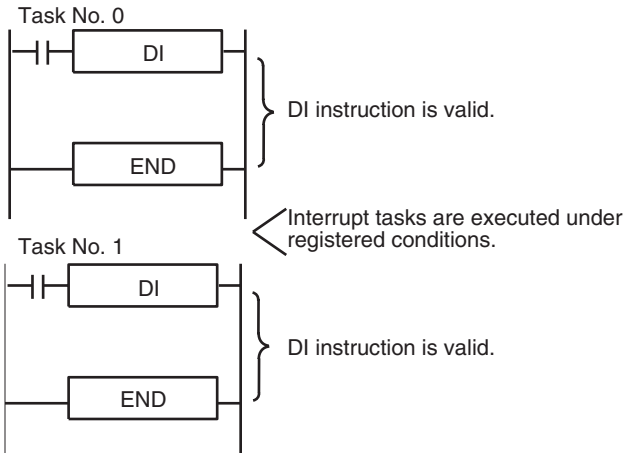
Precautions

All interrupt tasks will remain disabled until EI(694) is executed.

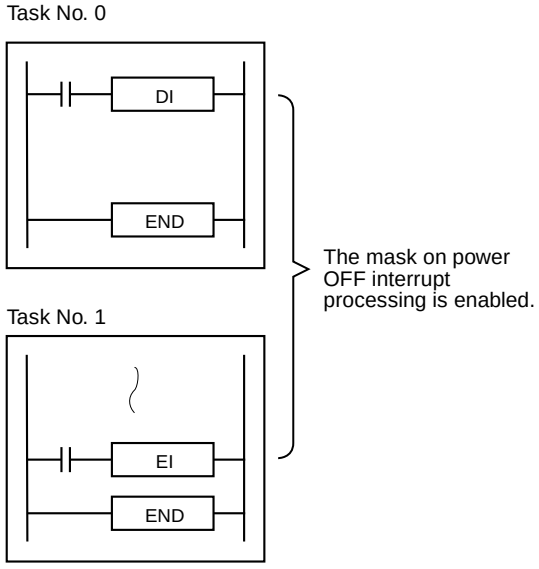
DI(693) cannot be executed from an interrupt task.

DI(693) cannot be executed for more than one cyclic task. To disable more than one cycle execution task, insert DI(693) in each cyclic task. Any interrupts that occur while one cycle execution task is being executed will be executed after the cycle execution task has been completed unless they are disabled by CLI(691) as shown in the following example.

When using DI(693) to disable Power OFF Interrupt Processing in a CS1-H, CJ1-H, and CJ1M CPU Unit, 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.)

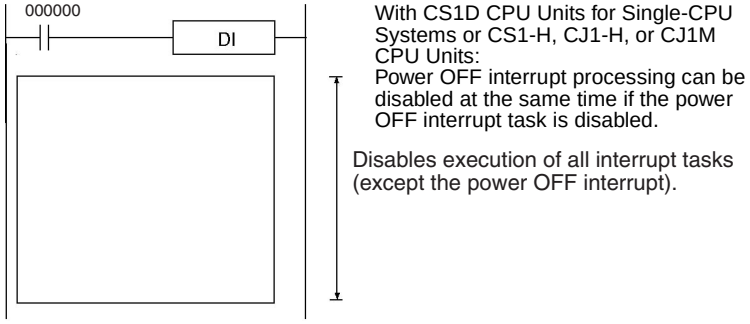


When a CS1D CPU Unit for Single-CPU System or a CS1-H, CJ1-H, or CJ1M CPU Unit is being used, the power OFF interrupt task is disabled, and A530 is set to A5A5 hex, the CPU Unit will be reset after execution of EI(694) in the event that a power interruption is detected during execution of the instructions between DI(693) and EI(694).



Examples

When CIO 000000 is ON in the following example, DI(693) disables all interrupt tasks other than the power OFF interrupt task.

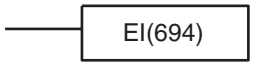


3-20-5 ENABLE INTERRUPTS: EI(694)

Purpose

Enables execution of all interrupt tasks that were disabled with DI(693). When a CS1D CPU Unit for Single-CPU System or a CS1-H, CJ1-H, or CJ1M CPU Unit is being used and the power OFF interrupt task is disabled, EI(694) simultaneously releases the disabled power OFF interrupt processing.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for Normally ON Condition**	EI(694)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Description

EI(694) is executed from the main program to temporarily enable all interrupt tasks that were disabled by DI(693). DI(693) disables all interrupts except the power OFF interrupt (I/O interrupts, scheduled interrupts, and external interrupts).

CS1-H, CJ1-H, and CJ1M CPU Units and Power OFF Interrupts

When a CS1-H, CJ1-H, and CJ1M CPU 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 instruction s between DI(693) and EI(694) have been executed. Refer to *3-20-4 DISABLE INTERRUPTS: DI(693)* for details on using DI(693) to disable power OFF interrupt processing.

Flags

Name	Label	Operation
Error Flag	ER	ON if EI(694) is executed from an interrupt task. OFF in all other cases.

Related Flags and Words

The following word is in the Auxiliary Area.

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.

Precautions

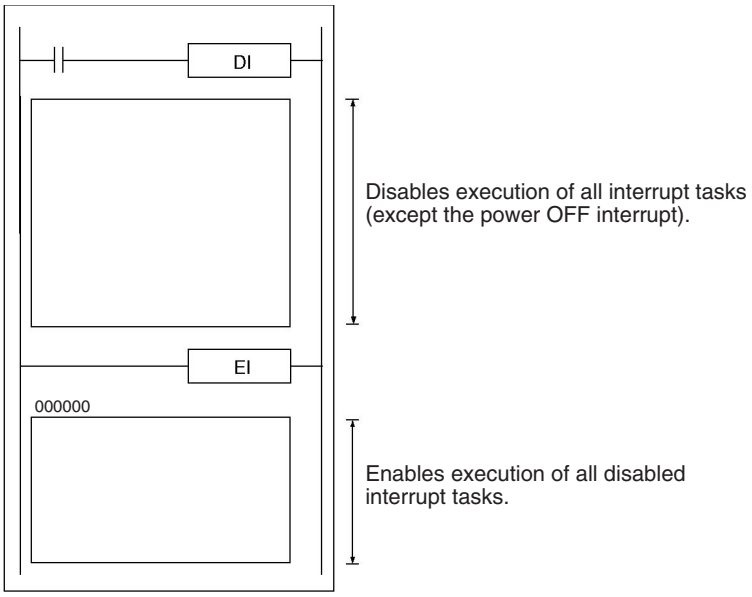
EI(694) does not require an execution condition. It is always executed with an ON execution condition. EI(694) enables the interrupt tasks that were disabled by DI(693).

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).

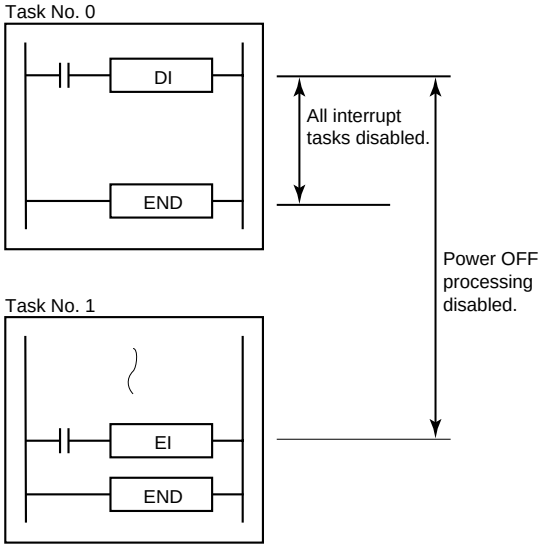
EI(694) cannot be executed in an interrupt task.

Examples

In the following example, EI(694) enables all interrupt tasks that were disabled by DI(693).



Note When the power OFF interrupt task is disabled for a 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.



3-20-6 Summary of Interrupt Control

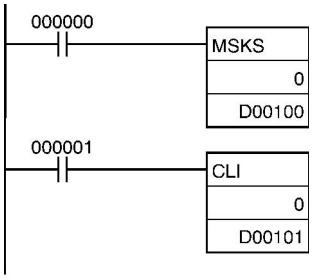
The interrupt control instructions control or read settings for I/O interrupts and scheduled interrupts. (DI(693) and EI(694) control the operation of external interrupts as well as I/O interrupts and scheduled interrupts.)

The instructions that act on individual interrupts have an operand, N, that identifies the source of the interrupt. Numbers 0 to 3 indicate Interrupt Input Units 0 to 3 and numbers 4 and 5 indicate scheduled interrupts 2 and 3.

I/O Interrupt Processing (N=0 to 3)

An I/O interrupt is caused by an input signal from an Interrupt Input Unit. Up to four Interrupt Input Units can be connected to the PLC. Unit numbers 0 to 3 are assigned to the Units based on their position in the PLC from left to right.

The following program example demonstrates the operation of MSKS(690) and CLI(691) when they are used to control I/O interrupts.



D00100	0 0 F 0
D00101	0 0 F 2

Operation of MSKS(690)

Both I/O interrupt tasks and scheduled interrupt tasks are masked (disabled) when the PLC is first turned on. MSKS(690) can be used to unmask or mask I/O interrupts and set the time intervals for scheduled interrupts.

In this example, MSKS(690) uses the contents of D00100 to unmask interrupt inputs 0 to 3 and mask interrupt inputs 4 to 7 from Interrupt Input Unit 0.

	F				0			
Interrupt inputs from Unit 0	7	6	5	4	3	2	1	0
Interrupt mask settings	1	1	1	1	0	0	0	0

1=Mask (Disable) 0=Unmask (Enable)

When interrupt input 3 goes from OFF to ON, execution of the main program will be interrupted and I/O interrupt task 3 (interrupt task 103) will be executed. Execution of the main program execution is resumed at the point of interruption after I/O interrupt task 3 has been completed.

I/O Interrupt Task Priority Levels

When two or more interrupt inputs are received simultaneously, the interrupts will be executed in order of their interrupt numbers from lowest to highest (100 to 131).

Unit	Interrupt tasks
Interrupt Input Unit 0	Inputs 0 to 7 correspond to I/O interrupt tasks 100 to 107.
Interrupt Input Unit 1	Inputs 0 to 7 correspond to I/O interrupt tasks 108 to 115.
Interrupt Input Unit 2	Inputs 0 to 7 correspond to I/O interrupt tasks 116 to 123.
Interrupt Input Unit 3	Inputs 0 to 7 correspond to I/O interrupt tasks 124 to 131.

When more interrupt inputs are received while an interrupt task is being executed, the recorded interrupts will be executed in order of their priority after the current interrupt task is completed.

If a scheduled interrupt occurs, the scheduled interrupt task will take priority over the I/O interrupt tasks.

Operation of CLI(691)

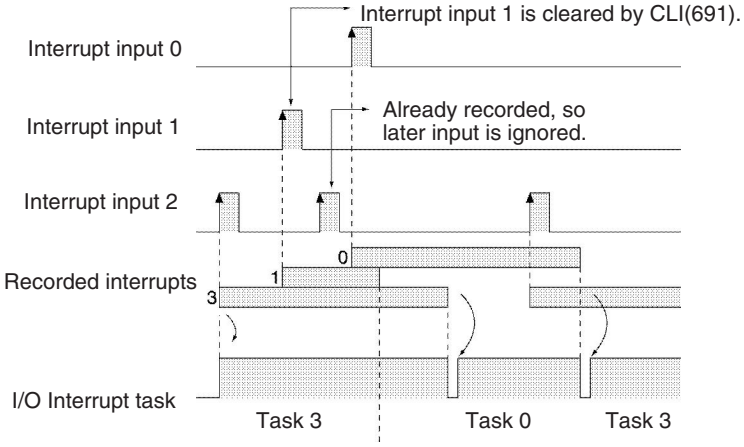
If an interrupt input is received while a different I/O interrupt task is being executed, the input's interrupt number is recorded internally until the current task and any higher priority tasks have been completed. CLI(691) can be used to clear recorded interrupts before they are executed, but cannot clear interrupt tasks that are being executed.

In this example, CLI(691) uses the contents of D00101 to clear all of the recorded interrupt inputs from Interrupt Input Unit 0 except inputs 0, 2, and 3.

	F				2			
Interrupt inputs from Unit 0	7	6	5	4	3	2	1	0
Interrupt clear/retain settings	1	1	1	1	0	0	1	0

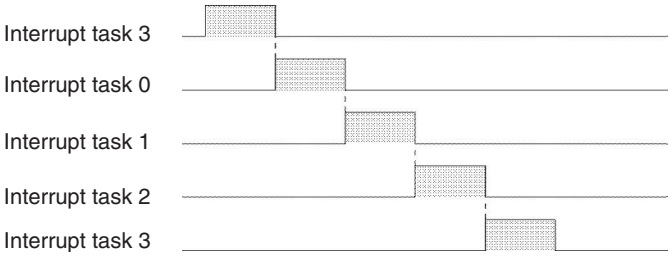
1=Clear recorded input 0=Retain recorded input

After completion of interrupt task 3, recorded interrupts are executed in order of their priority. Since an input from interrupt input 0 was recorded, I/O interrupt task 0 (interrupt task 100) will be executed when task 3 is completed. Interrupt input 1 is not retained by CLI(691), so that input is cleared.



CLI(691) is executed. Interrupt inputs 0 and 3 are retained and input 1 is cleared.

If interrupt inputs 0 through 3 all go ON and CLI(691) is not executed, all of the inputs will be recorded and the interrupt tasks will be executed in order after interrupt task 3 is completed. (The interrupt tasks are executed in order of their priority, from the lowest interrupt number to the highest.)



Note

1. It is not always necessary to use CLI(691).
2. When CLI(691) is not executed, all of the I/O interrupt inputs received during the execution of an interrupt task will be recorded. If a recorded input is received again, the later input will be ignored.
3. When two or more I/O interrupt inputs are recorded, they are executed in order of their priority. The order in which the recorded inputs were received is irrelevant.

Scheduled Interrupt Processing (N=4 or 5)

A scheduled interrupt is repeated at regular intervals set with MSKS(690) and independent of the timing of the PLC cycle. N numbers 4 and 5 correspond to scheduled interrupt numbers 2 and 3, respectively.

Scheduled Interrupt Processing

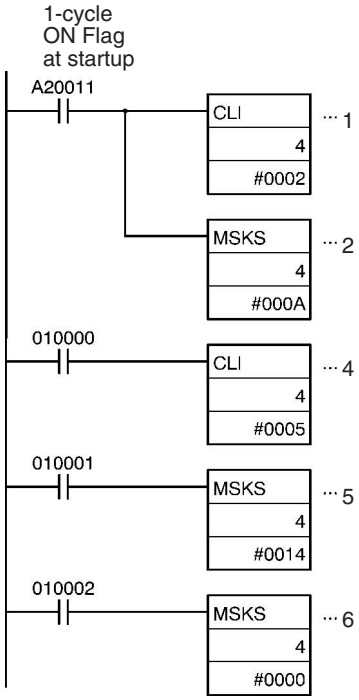
The main features of scheduled interrupt processing are listed below.

1,2,3...

1. The scheduled interrupts are masked (disabled) when the PLC is first turned on.
2. Set the time to the first scheduled interrupt (after execution of MSKS(690)) with CLI(691). The time to the first scheduled interrupt is unpredictable if it is not set with CLI(691).
3. The scheduled time interval setting and interrupt processing
   - Set the scheduled time interval with MSKS(690).
   - After MSKS(690) has been executed and the time to the first scheduled interrupt (set with CLI(691)) has passed, the task currently being processed will be interrupted and the scheduled interrupt task will be executed.
   - When the scheduled interrupt task execution reaches an END(001) instruction, program execution will resume at the point where the scheduled interrupt occurred.
   - Program execution will be interrupted and the scheduled interrupt task will be executed again when the scheduled time interval has passed. The scheduled interrupt task will be executed repeatedly until it is disabled.
4. Disabling a Scheduled Interrupt
   - A scheduled interrupt task can be disabled by setting the scheduled time interval to 0000 with MSKS(690).
   - When enabling the scheduled interrupt task again, be sure to set the time to the first scheduled interrupt with CLI(691) before setting the scheduled time interval again with MSKS(690).

Scheduled Interrupt Operation

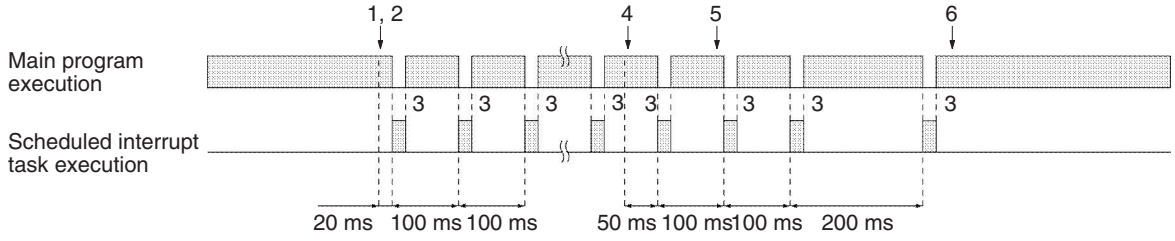
In the following example, the scheduled time interval units are set to 10 ms in the PLC Setup.



1,2,3...

1. The time to the first scheduled interrupt is set to 20 ms with CLI(691).
2. The scheduled time interval is set to 100 ms and execution of scheduled interrupt 2 is enabled with MSKS(690).
3. Scheduled interrupt 2 is executed 20 ms after execution of MSKS(690) and every 100 ms thereafter.
4. After scheduled interrupt processing has begun, the time to the next scheduled interrupt can be changed with CLI(690), but this setting is effective only one time.
5. After scheduled interrupt processing has begun, the scheduled time interval can be changed by executing MSKS(690). In this case, the time interval is changed from 100 ms to 200 ms.
6. Scheduled interrupt processing is disabled by executing MSKS(690) with a time interval of 0000.

The following timing chart shows the operation of the example listed above.



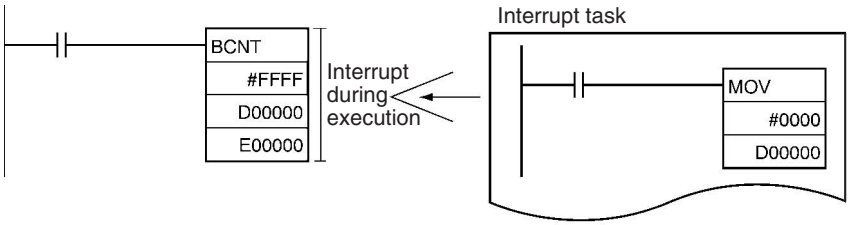
Precautions

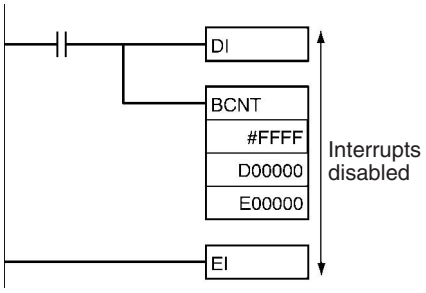
Be sure that the scheduled time interval is longer than the time required to execute the scheduled interrupt task. If the scheduled time interval is too short, the interrupt task will be executed continuously and a Cycle Time Too Long Error will occur. (A long scheduled interrupt task can seriously affect the main program's overall execution time.)

The scheduled interrupt is executed after the specified time interval plus the execution time for one instruction. Normally the time required to execute one instruction is negligible, but it can cause errors when instructions that take a

long time are being used; it can also cause errors in timers (TIM and TIMH) and data tracing. Be particularly careful when the scheduled time interval units are set to 0.5 ms or 1 ms in the PLC Setup.

Interrupts are accepted even while one instruction is being executed. Therefore, if an interrupt is accepted while an instruction requiring a long processing time is being executed, correct processing results may not be obtained because both the interrupt task and the instruction may access the same data. In such a case, use DI(693) and EI(694) to disable and enable the interrupt.





3-21 High-speed Counter/Pulse Output Instructions

This section describes instructions used to control the high-speed counters and pulse outputs.

Instruction	Mnemonic	Function code	Page
MODE CONTROL	INI	880	864
HIGH-SPEED COUNTER PV READ	PRV	881	868
COUNTER FREQUENCY CONVERT	PRV2	881	874
REGISTER COMPARISON TABLE	CTBL	882	878
SPEED OUTPUT	SPED	885	882
SET PULSES	PULS	886	887
PULSE OUTPUT	PLS2	887	890
ACCELERATION CONTROL	ACC	888	896
ORIGIN SEARCH	ORG	889	903
PULSE WITH VARIABLE DUTY FACTOR	PWM	891	906

3-21-1 MODE CONTROL: INI(880) (CJ1M-CPU21/22/23 Only)

Purpose

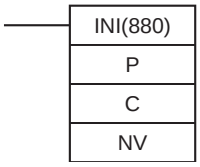
INI(880) can be used to execute the following operations for built-in I/O of CJ1M CPU Units:

- To start comparison with the high-speed counter comparison table
- To stop comparison with the high-speed counter comparison table
- To change the PV of the high-speed counter.
- To change the PV of interrupt inputs in counter mode.

- To change the PV of the pulse output (origin fixed at 0).
- To stop pulse output.

This instruction is supported by CJ1M-CPU21/22/23 CPU Units only.

Ladder Symbol



P: Port specifier
C: Control data
NV: First word with new PV

Variations

Variations		
	Executed Each Cycle for ON Condition	INI(880)
	Executed Once for Upward Differentiation	@INI(880)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

P: Port Specifier

P specifies the port to which the operation applies.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0010 hex	High-speed counter 0
0011 hex	High-speed counter 1
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
1000 hex	PWM(891) output 0
1001 hex	PWM(891) output 1

C: Control Data

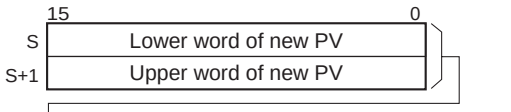
The function of INI(880) is determined by the control data, C.

C	INI(880) function
0000 hex	Starts comparison.
0001 hex	Stops comparison.
0002 hex	Changes the PV.
0003 hex	Stops pulse output.

NV: First Word with New PV

NV and NV+1 contain the new PV when changing the PV.

If C is 0002 hex (i.e., when changing a PV), NV and NV+1 contain the new PV. Any values in NV and NV+1 are ignored when C is not 0002 hex.



For Pulse Output or High-speed Counter Input:
0000 0000 to FFFF FFFF hex

For Interrupt Input in Counter Mode:
0000 0000 to 0000 FFFF hex

Operand Specifications

Area	P	C	NV
CIO Area	---	---	CIO 0000 to CIO 6142
Work Area	---	---	W000 to W510
Holding Bit Area	---	---	H000 to H510
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	---	---	D00000 to D32766
EM Area without bank	---	---	---
EM Area with bank	---	---	---
Indirect DM/EM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM/EM addresses in BCD	---	---	*D00000 to *D32767
Constants	See description of operand.	See description of operand.	---
Data Registers	---	---	---
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

INI(880) performs the operation specified in C for the port specified in P. The possible combinations of operations and ports are shown in the following table.

P: Port specifier	C: Control data			
	0000 hex: Start comparison	0001 hex: Stop comparison	0002 hex: Change PV	0003 hex: Stop pulse output
0000 or 0001 hex: Pulse output	Not allowed.	Not allowed.	OK	OK
0010 or 0011 hex: High-speed counter input	OK	OK	OK	Not allowed.
0100, 0101, 0102, or 0103 hex: Interrupt input in counter mode	Not allowed.	Not allowed.	OK	Not allowed.
1000 or 1001 hex: PWM (891) output	Not allowed.	Not allowed.	Not allowed.	OK

■ **Starting Comparison (C = 0000 hex)**

If C is 0000 hex, INI(880) starts comparison of a high-speed counter's PV to the comparison table registered with CTBL(882).

Note A target value comparison table must be registered in advance with CTBL(882). If INI(880) is executed without registering a table, the Error Flag will turn ON.

■ **Stopping Comparison (C = 0001 hex)**

If C is 0001 hex, INI(880) stops comparison of a high-speed counter's PV to the comparison table registered with CTBL(882).

■ **Changing a PV (C = 0002 hex)**

If C is 0002 hex, INI(880) changes a PV as shown in the following table.

Port and mode			Operation	Setting range
Pulse output (P = 0000 or 0001 hex)			The present value of the pulse output is changed. The new value is specified in NV and NV+1. **Note:** This instruction can be executed only when pulse output is stopped. An error will occur if it is executed during pulse output.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
High-speed counter input (P = 0010 or 0011 hex)	Linear Mode	Differential inputs, increment/ decrement pulses, or pulse + direction inputs	The present value of the high-speed counter is changed. The new value is specified in NV and NV+1. **Note:** An error will occur for the instruction if the specified port is not set for a high-speed counter.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
		Increment pulse input		0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)
	Ring Mode			0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)
Interrupt inputs in counter mode (P = 0100, 0101, 0102, or 0103 hex)			The present value of the interrupt input is changed. The new value is specified in NV and NV+1.	0000 0000 to 0000 FFFF hex (0 to 65,535) **Note:** An error will occur if a value outside this range is specified.

■ **Stopping Pulse Output (P = 1000 or 1001 hex and C = 0003 hex)**

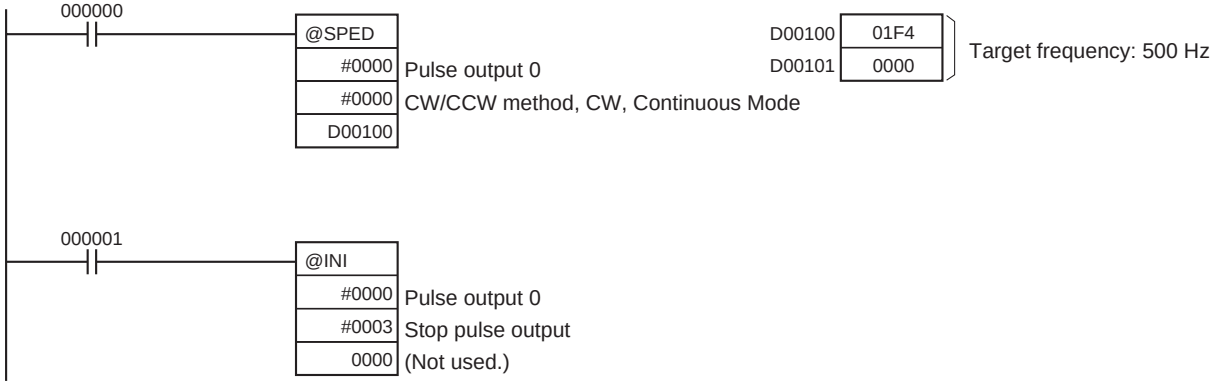
If C is 0003 hex, INI(880) immediately stops pulse output for the specified port. If this instruction is executed when pulse output is already stopped, then the pulse amount setting will be cleared.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P, C, or NV is exceeded. ON if the combination of P and C is not allowed. ON if a comparison table has not been registered but starting comparison is specified. ON if a new PV is specified for a port that is currently outputting pulses. ON if changing the PV of a high-speed counter is specified for a port that is not specified for a high-speed counter. ON if a value that is out of range is specified as the PV for an interrupt input in counter mode. ON if INI(880) is executed in an interrupt task for a high-speed counter and an interrupt occurs when CTBL(882) is executed. ON if executed for a port not set for an interrupt input in counter mode.

Example

When CIO 000000 turns ON in the following example, SPED(885) starts outputting pulses from pulse output 0 in Continuous Mode at 500 Hz. When CIO 000001 turns ON, pulse output is stopped by INI(880).



3-21-2 HIGH-SPEED COUNTER PV READ: PRV(881) (CJ1M-CPU21/22/23 Only)

Purpose

PRV(881) reads the following data on the built-in I/O of CJ1M CPU Units.

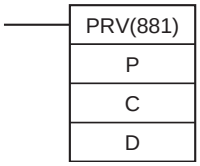
- PVs: High-speed counter PV, pulse output PV, interrupt input PV in counter mode.
- The following status information.

Status type	Contents
Pulse output status	Pulse Output Status Flag PV Underflow/Overflow Flag Pulse Output Amount Set Flag Pulse Output Completed Flag Pulse Output Flag No-origin Flag At Origin Flag Pulse Output Stopped Error Flag
High-speed counter input status	Comparison In-progress Flag PV Underflow/Overflow Flag
PWM(891) output status	Pulse Output In-progress Flag

- Range comparison results
- Pulse output frequency of pulse output 0 or pulse output 1 (Supported only by CJ1M CPU Units Ver. 2.0 or later.)
- High-speed counter frequency for high-speed counter input 0.

This instruction is supported by CJ1M-CPU21/22/23 CPU Units only.

Ladder Symbol



P: Port specifier
C: Control data
D: First destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	PRV(881)
	Executed Once for Upward Differentiation	@PRV(881)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

P: Port Specifier

P specifies the port to which the operation applies.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1
0010 hex	High-speed counter 0
0011 hex	High-speed counter 1
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
1000 hex	PWM(891) output 0
1001 hex	PWM(891) output 1

C: Control Data

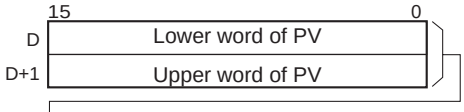
The function of INI(880) is determined by the control data, C.

C	PRV(881) function	Variations
0000 hex	Reads the PV.	---
0001 hex	Reads status.	---

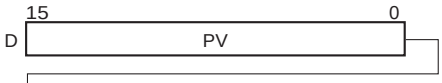
C	PRV(881) function	Variations
0002 hex	Reads range comparison results.	---
00□3 hex	P = 0000 or 0001: Reads the output frequency of pulse output 0 or pulse output 1. P = 0010: Reads the frequency of high-speed counter input 0.	C = 0003 hex: Standard operation C = 0013 hex: 10-ms sampling method for high frequency (supported only by CJ1M CPU Units Ver. 3.0 or later) C = 0023 hex: 100-ms sampling method for high frequency (supported only by CJ1M CPU Units Ver. 3.0 or later) C = 0033 hex: 1-s sampling method for high frequency (supported only by CJ1M CPU Units Ver. 3.0 or later)

D: First Destination Word

The PV is output to D or to D and D+1.



2-word PV
Pulse output PV, high-speed counter input PV,
high-speed counter input frequency for high-speed counter input 0



1-word PV
Interrupt input PV in counter mode, status, range comparison results

Operand Specifications

Area	P	C	D
CIO Area	---	---	CIO 0000 to CIO 6142
Work Area	---	---	W000 to W510
Holding Bit Area	---	---	H000 to H510
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	---	---	D00000 to D32766
EM Area without bank	---	---	---
EM Area with bank	---	---	---
Indirect DM/EM addresses in binary	---	---	@ D00000 to @ D32766
Indirect DM/EM addresses in BCD	---	---	*D00000 to *D32766
Constants	See description of operand.	See description of operand.	---
Data Registers	---	---	---

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

Description

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			
				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 or 0001 hex: Pulse output	OK	OK	Not allowed.	OK (See note.)	Not allowed.	Not allowed.	Not allowed.
0010 or 0011 hex: High-speed counter input	OK	OK	OK	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, 0101, 0102, or 0103 hex: Interrupt input in counter mode	OK	Not allowed.	Not allowed.	Not allowed.	Not allowed.	Not allowed.	Not allowed.
1000 or 1001 hex: PWM (891) output	Not allowed.	OK	Not allowed.	Not allowed.	Not allowed.	Not allowed.	Not allowed.

Note CJ1M CPU Units with unit version 3.0 or later only.

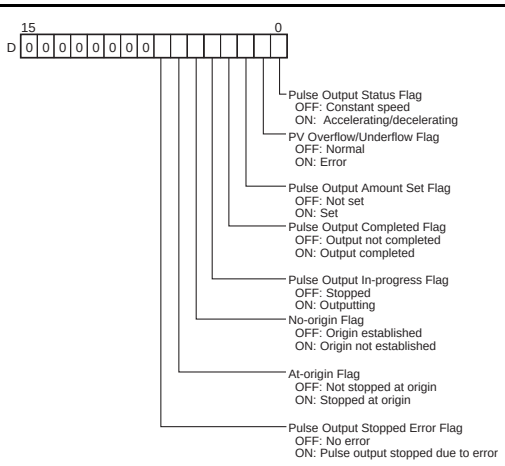
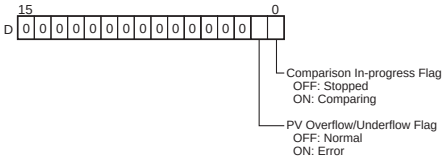
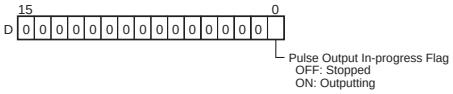
■ **Reading a PV (C = 0000 hex)**

If C is 0000 hex, PRV(881) reads a PV as shown in the following table.

Port and mode		Operation	Setting range
Pulse output (P = 0000 or 0001 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 or 0011 hex)	Linear Mode	The present value of the high-speed counter is stored in D and D+1.	8000 0000 to 7FFF FFFF hex (–2,147,483,648 to 2,147,483,647)
	Ring Mode		0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)
Interrupt inputs in counter mode (P = 0100, 0101, 0102, or 0103 hex)		The present value of the interrupt input is stored in D.	0000 to FFFF hex (0 to 65,535)

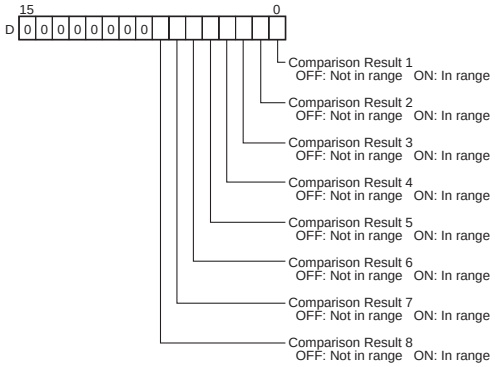
■ Reading Status (C = 0001 hex)

If C is 0001 hex, PRV(881) reads status as shown in the following table.

Port and mode	Operation	Results of reading
Pulse output	The pulse output status is stored in D.	
High-speed counter input	The high-speed counter status is stored in D.	
PWM(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.



■ 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 or 1 or the frequency being input to high-speed counter 0 and stores it in D and D+1.

Frequency Ranges

Value of P	Conversion result
0000 or 0001 hex (Reading the frequency of pulse output 0 or 1)	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 = 00000000 to 000186A0 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 000186A0 hex.
	Counter input method: 4× differential phase mode Result = 00000000 to 00030D40 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 00030D40 hex.

Pulse Frequency Calculation Methods

When the CPU Unit is a CJ1M CPU Unit with version number 3.0 or later, there are two ways to calculate the frequency of pulses output from pulse output 0 or 1 or pulses input to high-speed counter 0.

1. Standard Calculation Method (Earlier Method)

   The count is calculated by counting each pulse regardless of the frequency. At high frequencies, the rising or falling edges of some pulses will be corrupted, resulting in errors (roughly 1% error max. at 100 kHz).

2. High-frequency Calculation Method

   In this case, the counting method is switched at high and low frequencies.

   - High-frequency counting

     At high frequencies (above 1 kHz), the function counts the number of pulses within a fixed interval (the sampling time) and calculates the frequency from that count. One of the following three sampling times can be selected by setting the rightmost two digits of C.

Sampling time	Value of C	Description
10 ms	0013 hex	Counts the number of pulses every 10 ms. The error is 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.

   - Low-frequency counting

     At frequencies below 1 kHz, the Standard Calculation Method is used, regardless of the sampling time setting.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P or C is exceeded. ON if the combination of P and C is not allowed. ON if reading range comparison results is specified even though range comparison is not being executed. ON if reading the output frequency is specified for anything except for high-speed counter 0. ON if specified for a port not set for a high-speed counter. ON if executed for a port not set for an interrupt input in counter mode.

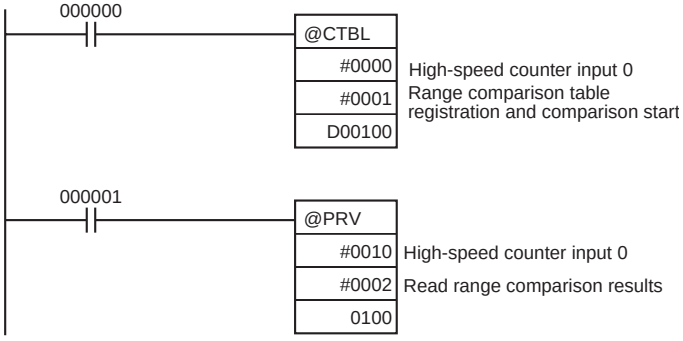
Precautions

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.

Examples

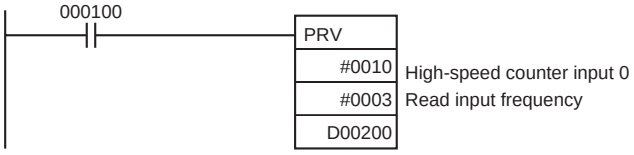
■ **Example 1**

When CIO 000000 turns ON in the following programming example, CTBL(882) registers a range comparison table for high-speed counter 0 and starts comparison. When CIO 000001 turns ON, PRV(881) reads the range comparison results at that time and stores them in CIO 0100.



■ **Example 2**

When CIO 000100 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 D00200 and D00201.



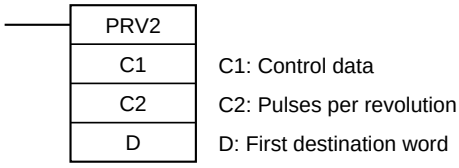
3-21-3 COUNTER FREQUENCY CONVERT: PRV2(883)

Purpose

PRV2(883) reads the pulse frequency input from a high-speed counter and either converts the frequency to a rotational speed or converts the counter PV to the total number of revolutions. The result is output to the destination words as 8-digit hexadecimal. Pulses can be input from high-speed counter 0 only.

This instruction is supported only by the CJ1M-CPU21/22/23 CPU Unit Ver. 2.0 or later.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	PRV2(883)
	Executed Once for Upward Differentiation	@PRV2(883)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

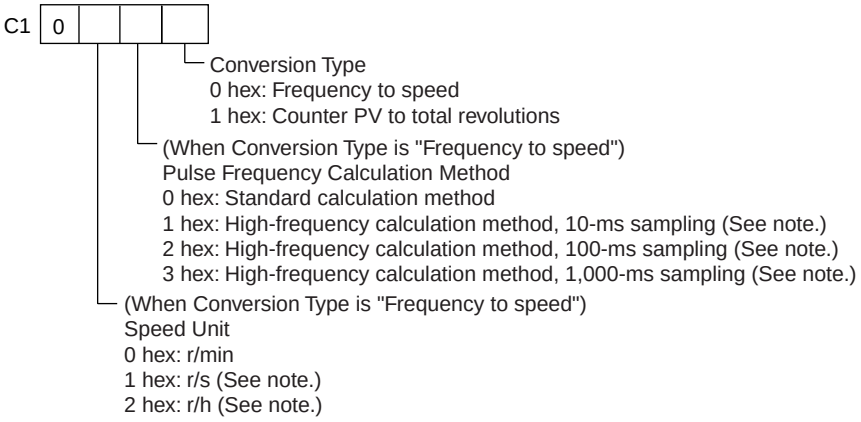
Operands

C1: Control Data

The function of PRV2(883) is determined by the control data, C1.

C1	PRV2(883) function
0□∗0 hex (See note.)	Converts frequency to rotation speed.
0001 hex	Converts counter PV to total number of revolutions.

Note The second digit of C (□) specifies the units and the third digit (∗) specifies the frequency calculation method.

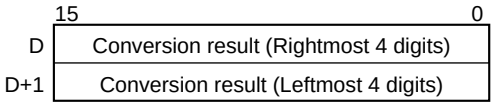


C2: Pulses per Revolution

Specifies the number of pulses per revolution (0001 to FFFF hex).

D: First Destination Word

The PV is output to D or to D and D+1.



Operand Specifications

Area	C1	C2	D
CIO Area	---	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142
Work Area	---	W000 to W511	W000 to W510

Area	C1	C2	D
Holding Bit Area	---	H000 to H511	H000 to H510
Auxiliary Bit Area	---	A448 to A959	A448 to A958
Timer Area	---	T0000 to T4095	T0000 to T4094
Counter Area	---	C0000 to C4095	C0000 to C4094
DM Area	---	D00000 to D32767	D00000 to D32766
EM Area without bank	---	---	---
EM Area with bank	---	---	---
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767	@ D00000 to @ D32767
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767	*D00000 to *D32767
Constants	See description of operand.	---	---
Data Registers	---	---	---
Index Registers	---	---	---
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

PRV2(883) converts the pulse frequency input from high-speed counter 0, according to the conversion method specified in C1 and the pulses/revolution coefficient specified in C2, and outputs the result to D and D+1.

Select one of the following conversion methods by setting C1 to 0000 hex or 0001 hex.

Converting Frequency to Rotation Speed (C1 = 0□∗0 hex)

If C1 is 0□∗0 hex, PRV2(883) calculates the rotation speed (r/min) from the frequency data and pulses/revolution setting. The second digit of C (□) specifies the units and the third digit (∗) specifies the frequency calculation method.

1. Rotation Speed Units
   - Rotation Speed Units = r/min

     When the second digit of C (□) is 0, PRV2(883) calculates the rotation speed in r/min from the frequency data and pulses/revolution setting.

     Rotation speed (r/min) = (Frequency ÷ Pulses/revolution) × 60
   - Rotation Speed Units = r/s (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
       Conversion result = 00000000 to 000186A0 hex (0 to 100,000)

(If a frequency higher than 100 kHz has been input, the output will remain at the maximum value of 000186A0 hex.)

- Counter input method: 4× differential phase mode
  Conversion result = 00000000 to 00030D40 hex (0 to 200,000)
  (If a frequency higher than 200 kHz has been input, the output will remain at the maximum value of 00030D40 hex.)

2. Frequency Calculation Method

When the CPU Unit is a CJ1M CPU Unit with version number 3.0 or later, there are two ways to calculate the frequency of pulses input to high-speed counter 0.

a) Standard Calculation Method (C1 = 0□00)

The count is calculated by counting each pulse regardless of the frequency. At high frequencies, the rising or falling edges of some pulses will be corrupted, resulting in errors (about 1% error max. at 100 kHz).

b) High-frequency Calculation Method

In this case, the counting method is switched at high and low frequencies. (Supported by CJM1 CPU Unit Ver. 3.0 or later only)

- High-frequency counting (C1 = 0□10, 0□20, or 0□30)

  At high frequencies (above 1 kHz), the function counts the number of pulses within a fixed interval (the sampling time) and calculates the frequency from that count. One of the following three sampling times can be selected by the third digit of C1.

Sampling time	Value of C1	Description
10 ms	0□10 hex	Counts the number of pulses every 10 ms. The error is 10% max. at 1 kHz.
100 ms	0□20 hex	Counts the number of pulses every 100 ms. The error is 1% max. at 1 kHz.
1 s	0□30 hex	Counts the number of pulses every 1 s. The error is 0.1% max. at 1 kHz.

- Low-frequency counting

  At frequencies below 1 kHz, the Standard Calculation Method is used, regardless of the sampling time setting.

Converting Counter PV to Total Number of Revolutions (C1 = 0001 hex)

If C1 is 0001 hex, PRV2(883) calculates the cumulative number of revolutions from the counter PV and pulses/revolution setting.

Conversion result = Counter PV ÷ Pulses/revolution

Flags

Name	Label	Operation
Error Flag	ER	ON if high-speed counter 0 is disabled in the settings. ON if C1 is not in the specified range (0000 or 0001). ON if the pulses/revolution setting in C2 is 0000.

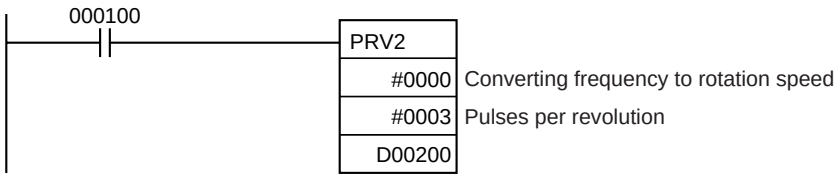
Precautions

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.

Examples

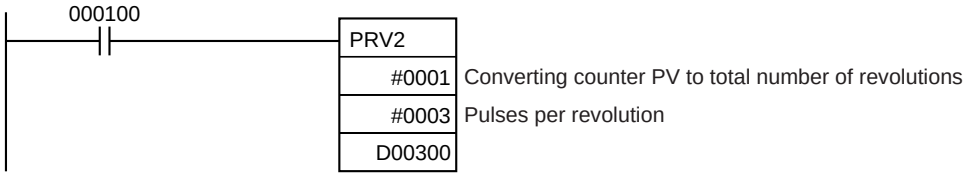
■ **Example 1**

When CIO 000100 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 D00201 and D00200.



■ **Example 2**

When CIO 000100 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 D00301 and D00300.



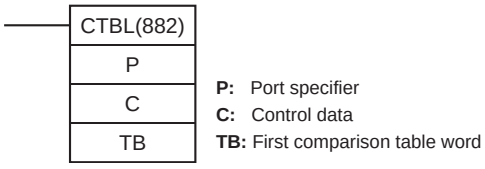
3-21-4 REGISTER COMPARISON TABLE: CTBL(882) (CJ1M-CPU21/22/23 Only)

Purpose

CTBL(882) is used to register a comparison table and perform comparisons for a high-speed counter PV. Either target value or range comparisons are possible. An interrupt task is executed when a specified condition is met.

This instruction is supported by CJ1M-CPU21/22/23 CPU Units only.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	CTBL(882)
	Executed Once for Upward Differentiation	@CTBL(882)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

P: Port Specifier

P specifies the port for which pulses are to be counted as shown in the following table.

P	Port
0000 hex	High-speed counter 0
0001 hex	High-speed counter 1

C: Control Data

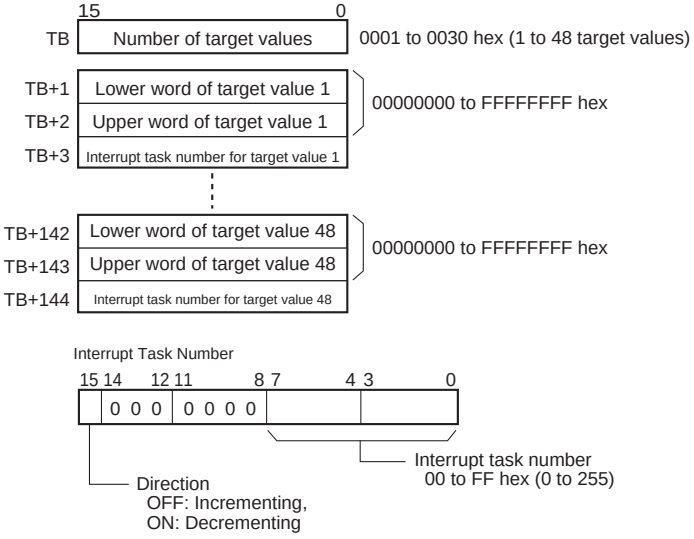
The function of CTBL(882) is determined by the control data, C, as shown in the following table.

C	CTBL(882) function
0000 hex	Registers a target value comparison table and starts comparison.
0001 hex	Registers a range comparison table and performs one comparison.
0002 hex	Registers a target value comparison table. Comparison is started with INI(880).
0003 hex	Registers a range comparison table. Comparison is started with INI(880).

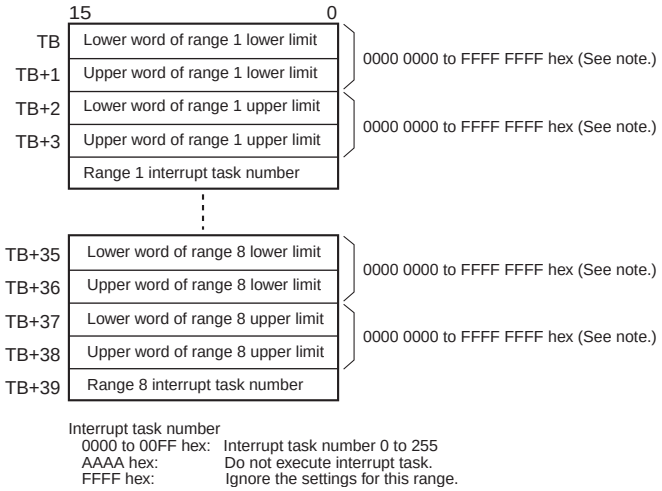
TB: First Table Comparison Word

TB is the first word of the comparison table. The structure of the comparison table depends on the type of comparison being performed.

For target value comparison, the length of the comparison table is determined by the number of target values specified in TB. The table can be between 4 and 145 words long, as shown below.



For range comparison, the comparison table always contains eight ranges. The table is 40 words long, as shown below. If it is not necessary to set eight ranges, set the interrupt task number to FFFF hex for all unused ranges.



Note Always set the upper limit greater than or equal to the lower limit for any one range.

Operand Specifications

Area	P	C	TB
CIO Area	---	---	CIO 0000 to CIO 6143
Work Area	---	---	W000 to W511
Holding Bit Area	---	---	H000 to H511
Auxiliary Bit Area	---	---	A448 to A959
Timer Area	---	---	T0000 to T4095
Counter Area	---	---	C0000 to C4095
DM Area	---	---	D00000 to D32767
EM Area without bank	---	---	---
EM Area with bank	---	---	---
Indirect DM/EM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM/EM addresses in BCD	---	---	*D00000 to *D32767
Constants	See description of operand.	See description of operand.	---
Data Registers	---	---	---
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

CTBL(882) registers a comparison table or registers and comparison table and starts comparison for the port specified in P and the method specified in C. Once a comparison table is registered, it is valid until a different table is registered or until the CPU Unit is switched to PROGRAM mode.

Each time CTBL(882) is executed, comparison is started under the specified conditions. When using CTBL(882) to start comparison, it is normally suffi-

cient to use the differentiated version (@CTBL(882)) of the instruction or an execution condition that is turned ON only for one scan.

Note If an interrupt task that has not been registered is specified, a fatal program error will occur the first time an interrupt is generated.

■ Registering a Comparison Table (C = 0002 or 0003 hex)

If C is set to 0002 or 0003 hex, a comparison table will be registered, but comparison will not be started. Comparison is started with INI(880).

■ Registering a Comparison Table and Starting Comparison (C = 0000 or 0001 hex)

If C is set to 0000 or 0001 hex, a comparison table will be registered, and comparison will be started.

■ Stopping Comparison

Comparison is stopped with INI(880). It makes no difference what instruction was used to start comparison.

■ Target Value Comparison

The corresponding interrupt task is called and executed when the PV matches a target value.

- The same interrupt task number can be specified for more than one target value.
- The direction can be set to specify whether the target value is valid when the PV is being incremented or decremented. If bit 15 in the word used to specify the interrupt task number for the range is OFF, the PV will be compared to the target value only when the PV is being incremented, and if bit 00 is ON, only when the PV is being decremented.
- The comparison table can contain up to 48 target values, and the number of target values is specified in TB (i.e., the length of the table depends on the number of target values that is specified).
- Comparisons are performed for all target values registered in the table.

Note

1. An error will occur if the same target value with the same comparison direction is registered more than once in the same table.
2. If the high-speed counter is set for incremental pulse mode, an error will occur if decrementing is set in the table as the direction for comparison.
3. If the count direction changes while the PV equals a target value that was reached in the direction opposite to that set as the comparison direction, the comparison condition for that target value will not be met. Do not set target values at peak and bottom values of the count value.

Range Comparison

The corresponding interrupt task is called and executed when the PV enters a set range.

- The same interrupt task number can be specified for more than one target value.
- The range comparison table contains 8 ranges, each of which is defined by a lower limit and an upper limit. If a range is not to be used, set the interrupt task number to FFFF hex to disable the range.
- The interrupt task is executed only once when the PV enters the range.
- If the PV is within more than one range when the comparison is made, the interrupt task for the range closest to the beginning of the table will be given priority and other interrupt tasks will be executed in following cycles.

- If there is no reason to execute an interrupt task, specify AAAA hex as the interrupt task number. The range comparison results can be read with PRV(881) or using the Range Comparison In-progress Flags.

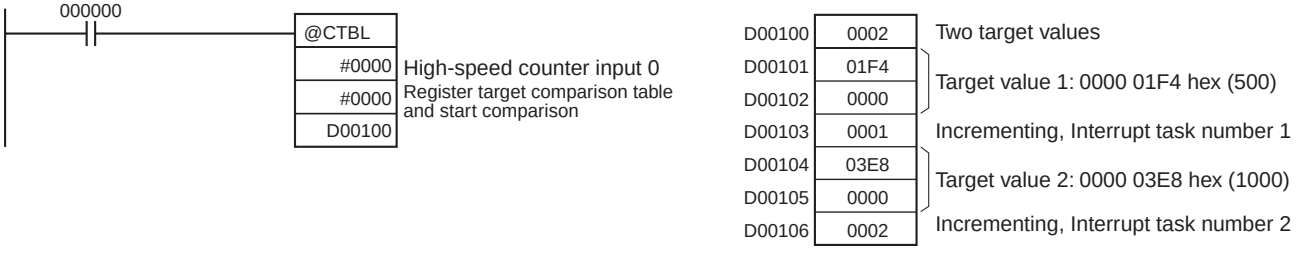
Note An error will occur if the upper limit is less than the lower limit for any one range.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P or C is exceeded. ON if the number of target values specified for target value comparison is set to 0. ON if the number of target values specified for target value comparison exceeds 48. ON if the same target value is specified more than once in the same comparison direction for target comparison. ON if the upper value is less than the lower value for any range. ON if the set values for all ranges are disabled during a range comparison. ON if the high-speed counter is set for incremental pulse mode and decrementing is set in the table as the direction for comparison. ON if an instruction is executed when the high-speed counter is set to Ring Mode and the specified value exceeds the maximum ring value. ON if specified for a port not set for a high-speed counter. ON if executed for a different comparison method while comparison is already in progress.

Example

When CIO 000000 turns ON in the following programming example, CTBL(882) registers a target value comparison table and starts comparison for high-speed counter 0. The PV of the high-speed counter is counted incrementally and when it reaches 500, it equals target value 1 and interrupt task 1 is executed. When the PV is incremented to 1000, it equals target value 2 and interrupt task 2 is executed.



3-21-5 SPEED OUTPUT: SPED(885) (CJ1M-CPU21/22/23 Only)

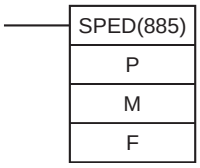
Purpose

SPED(885) is used to set the output pulse frequency for a specific port and start pulse output without acceleration or deceleration. Either independent mode positioning or continuous mode speed control is possible. For independent mode positioning, the number of pulses is set using PULS(886).

SPED(885) can also be executed during pulse output to change the output frequency, creating stepwise changes in the speed.

This instruction is supported by CJ1M-CPU21/22/23 CPU Units only.

Ladder Symbol



P: Port specifier
M: Output mode
F: First pulse frequency word

Variations

Variations	**Executed Each Cycle for ON Condition**	SPED(885)
	Executed Once for Upward Differentiation	@SPED(885)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

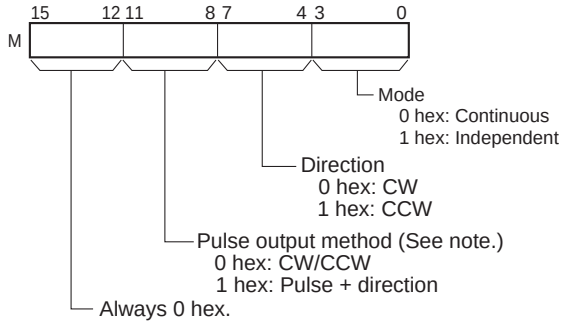
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

P: Port Specifier
The port specifier specifies the port where the pulses will be output.

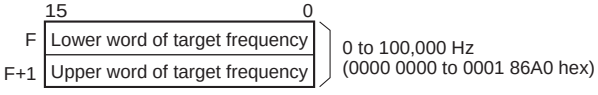
P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1

M: Output Mode
The value of M determines the output mode.



Note: Use the same pulse output method when using both pulse outputs 0 and 1.

F: First Pulse Frequency Word
The value of F and F+1 sets the pulse frequency in Hz.



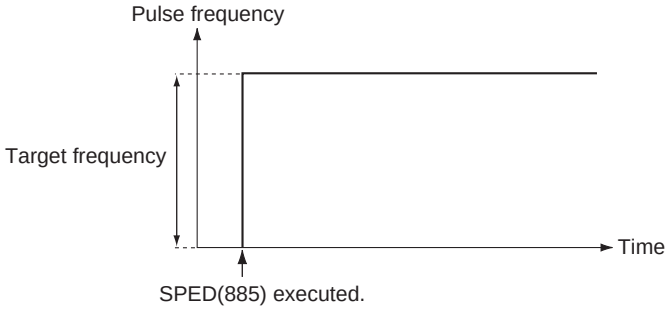
Operand Specifications

Area	P	M	F
CIO Area	---	---	CIO 0000 to CIO 6142
Work Area	---	---	W000 to W510
Holding Bit Area	---	---	H000 to H510
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	---	---	D00000 to D32766
EM Area without bank	---	---	---

Area	P	M	F
EM Area with bank	---	---	---
Indirect DM/EM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM/EM addresses in BCD	---	---	*D00000 to *D32767
Constants	See description of operand.	See description of operand.	See description of operand.
Data Registers	---	---	---
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

SPED(885) starts pulse output on the port specified in P using the method specified in M at the frequency specified in F. Pulse output will be started each time SPED(885) is executed. It is thus normally sufficient to use the differentiated version (@SPED(885)) of the instruction or an execution condition that is turned ON only for one scan.



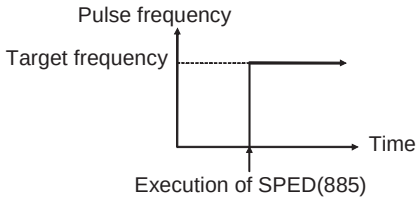
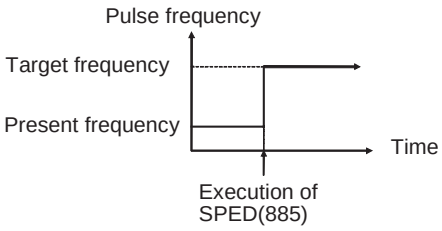
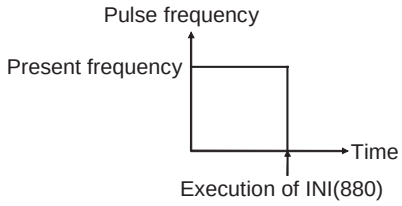
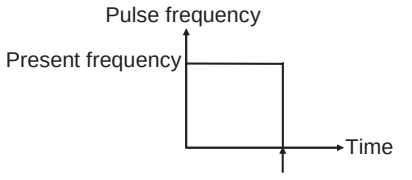
In independent mode, pulse output will stop automatically when the number of pulses set with PULS(886) in advance have been output. In continuous mode, pulse output will continue until stopped from the program.

An error will occur if the mode is changed between independent and continuous mode while pulses are being output.

■ Continuous Mode Speed Control

When continuous mode operation is started, pulse output will be continued until it is stopped from the program.

Note Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	To output with specified speed	Changing the speed (frequency) in one step		Outputs pulses at a specified frequency.	SPED(885) (Continuous)
Changing settings	To change speed in one step	Changing the speed during operation		Changes the frequency (higher or lower) of the pulse output in one step.	SPED(885) (Continuous) ↓ SPED(885) (Continuous)
Stopping pulse output	Stop pulse output	Immediate stop		Stops the pulse output immediately.	SPED(885) (Continuous) ↓ INI(880)
	Stop pulse output	Immediate stop		Stops the pulse output immediately.	SPED(885) (Continuous) ↓ SPED(885) (Continuous, Target frequency of 0 Hz)

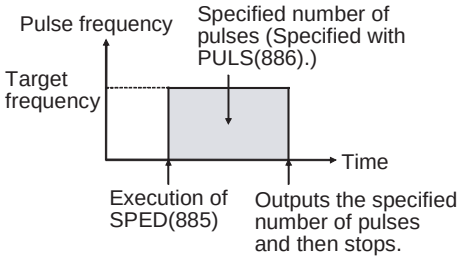
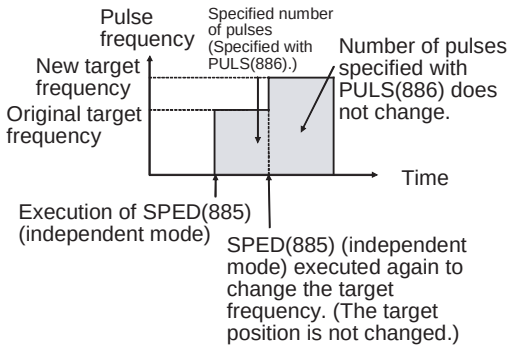
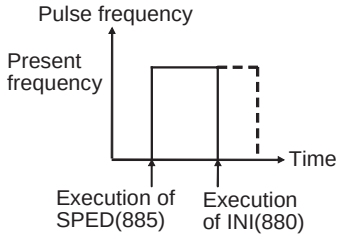
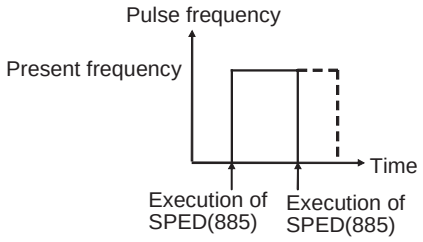
■ Independent Mode Positioning

When independent mode operation is started, pulse output will be continued until the specified number of pulses has been output.

Note

1. Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.
2. The number of output pulses must be set each time output is restarted.
3. The number of output pulses must be set in advance with PULS(881). Pulses will not be output for SPED(885) if PULS(881) is not executed first.

4. The direction set in the SPED(885) operand will be ignored if the number of pulses is set with PULS(881) as an absolute value.

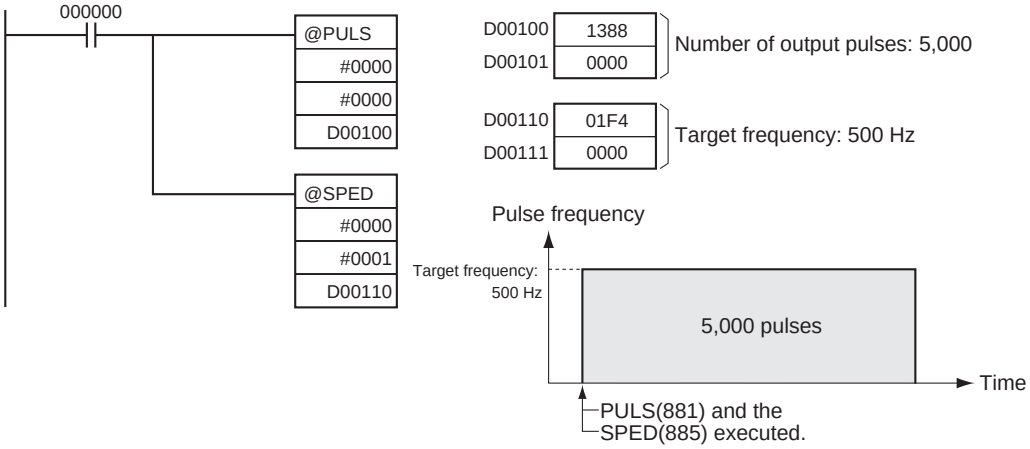
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	To output with specified speed	Positioning without acceleration or deceleration		Starts outputting pulses at the specified frequency and stops immediately when the specified number of pulses has been output. **Note** The target position (specified number of pulses) cannot be changed during positioning.	PULS(886) ↓ SPED(885) (Independent)
Changing settings	To change speed in one step	Changing the speed in one step during operation		SPED(885) can be executed during positioning to change (raise or lower) the pulse output frequency in one step. The target position (specified number of pulses) is not changed.	PULS(886) ↓ SPED(885) (Independent) ↓ SPED(885) (Independent)
Stopping pulse output	To stop pulse output (Number of pulses setting is not preserved.)	Immediate stop		Stops the pulse output immediately and clears the number of output pulses setting.	PULS(886) ↓ SPED(885) (Independent) ↓ INI(880) PLS2(887) ↓ INI(880)
	Stop pulse output (Number of pulses setting is not preserved.)	Immediate stop		Stops the pulse output immediately and clears the number of output pulses setting.	PULS(886) ↓ SPED(885) (Independent) ↓ SPED(885), (Independent, Target frequency of 0 Hz)

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P, M, or F is exceeded. ON if PLS2(887) or ORG(889) is already being executed to control pulse output for the specified port. ON if SPED(885) or INI(880) is used to change the mode between continuous and independent output during pulse output. ON if SPED(885) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task. ON if SPEC(885) is executed in independent mode with an absolute number of pulses and the origin has not been established.

Example

When CIO 000000 turns ON in the following programming example, PULS(886) sets the number of output pulses for pulse output 0. An absolute value of 5,000 pulses is set. SPED(885) is executed next to start pulse output using the CW/CCW method in the clockwise direction in independent mode at a target frequency of 500 Hz.



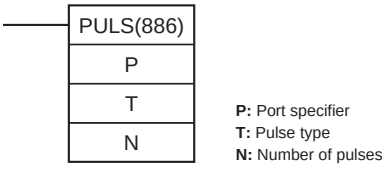
3-21-6 SET PULSES: PULS(886) (CJ1M-CPU21/22/23 Only)

Purpose

PULS(886) is used to set the pulse output amount (number of output pulses) for pulse outputs that are started later in the program using SPED(885) or ACC(888) in independent mode.

This instruction is supported by CJ1M-CPU21/22/23 CPU Units only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	PULS(886)
	Executed Once for Upward Differentiation	@PULS(886)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

P: Port Specifier

The port specifier indicates the port. The parameters set in D and N will apply to the next SPED(885) or ACC(888) instruction in which the same port output location is specified.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1

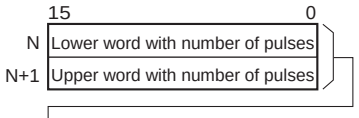
T: Pulse Type

T specifies the type of pulses that are output as follows:

T	Pulse type
0000 hex	Relative
0001 hex	Absolute

N and N+1: Number of Pulses

N and N+1 specify the number of pulses for relative pulse output or the absolute target position for absolute pulse in 8-digit hexadecimal.



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	P	T	N
CIO Area	---	---	CIO 0000 to CIO 6142
Work Area	---	---	W000 to W510
Holding Bit Area	---	---	H000 to H510
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T4094
Counter Area	---	---	C0000 to C4094
DM Area	---	---	D00000 to D32766
EM Area without bank	---	---	---
EM Area with bank	---	---	---
Indirect DM/EM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM/EM addresses in BCD	---	---	*D00000 to *D32767
Constants	See description of operand.	See description of operand.	See description of operand.
Data Registers	---	---	---

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

Description

PULS(886) sets the pulse type and number of pulses specified in T and N for the port specified in P. Actual output of the pulses is started later in the program using SPED(885) or ACC(888) in independent mode.

Flags

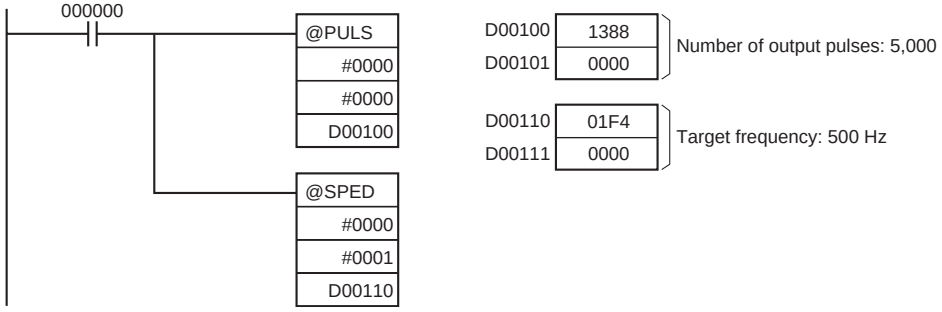
Name	Label	Operation
Error Flag	ER	ON if the specified range for P, T, or N is exceeded. ON if PULS(886) is executed for a port that is already outputting pulses. ON if PULS(886) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task.

Precautions

- An error will occur if PULS(886) is executed when pulses are already being output. Use the differentiated version (@PULS(886)) of the instruction or an execution condition that is turned ON only for one scan to prevent this.
- The calculated number of pulses output for PULS(886) will not change even if INI(880) is used to change the PV of the pulse output.
- The direction set for SPED(885) or ACC(888) will be ignored if the number of pulses is set with PULS(881) as an absolute value.
- It is possible to move outside of the range of the PV of the pulse output amount (–2,147,483,648 to 2,147,483,647).

Example

When CIO 000000 turns ON in the following programming example, PULS(886) sets the number of output pulses for pulse output 0. An absolute value of 5,000 pulses is set. SPED(885) is executed next to start pulse output using the CW/CCW method in the clockwise direction in independent mode at a target frequency of 500 Hz.



3-21-7 PULSE OUTPUT: PLS2(887) (CJ1M-CPU21/22/23 Only)

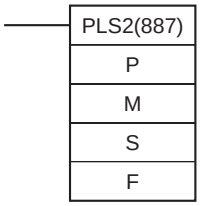
Purpose

PLS2(887) outputs a specified number of pulses to the specified port. Pulse output starts at a specified startup frequency, accelerates to the target frequency at a specified acceleration rate, decelerates at the specified deceleration rate, and stops at approximately the same frequency as the startup frequency. Only independent mode positioning is supported.

PLS2(887) can also be executed during pulse output to change the number of output pulses, target frequency, acceleration rate, or deceleration rate. PLS2(887) can thus be used for sloped speed changes with different acceleration and deceleration rates, target position changes, target and speed changes, or direction changes.

This instruction is supported by CJ1M-CPU21/22/23 CPU Units only.

Ladder Symbol



P: Port specifier
M: Output mode
S: First word of settings table
F: First word of starting frequency

Variations

Variations	**Executed Each Cycle for ON Condition**	PLS2(887)
	Executed Once for Upward Differentiation	@PLS2(887)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

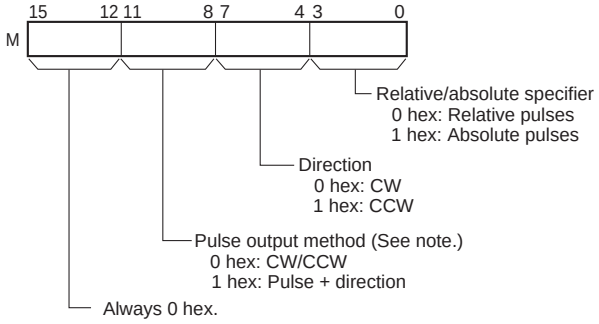
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

P: Port Specifier
The port specifier indicates the port.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1

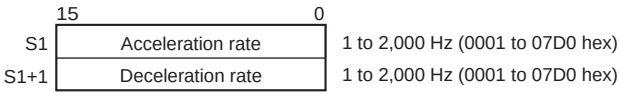
M: Output Mode
The content of M specifies the parameters for the pulse output as follows:



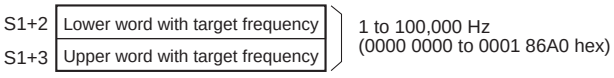
Note: Use the same pulse output method when using both pulse outputs 0 and 1.

S: First Word of Settings Table

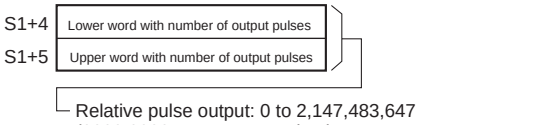
The contents of S to S+5 control the pulse output as shown in the following diagrams.



Specify the increase or decrease in the frequency per pulse control period (4 ms).



Specify the frequency after acceleration in Hz.



Relative pulse output: 0 to 2,147,483,647
(0000 0000 to 7FFF FFFF hex)

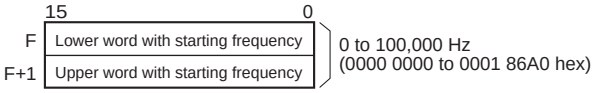
Absolute pulse output: –2,147,483,648 to 2,147,483,647
(8000 0000 to 7FFF FFFF hex)

The actual number of movement pulses that will be output are as follows:

For relative pulse output, the number of movement pulses = the set number of pulses. For absolute pulse output, the number of movement pulses = the set number of pulses – the PV.

F: First Word of Starting Frequency

The starting frequency is given in F and F+1.



Specify the starting frequency in Hz.

Operand Specifications

Area	P	M	S	F
CIO Area	---	---	CIO 0000 to CIO 6138	CIO 0000 to CIO 6142
Work Area	---	---	W000 to W506	W000 to W510
Holding Bit Area	---	---	H000 to H506	H000 to H510
Auxiliary Bit Area	---	---	A448 to A954	A448 to A958
Timer Area	---	---	T0000 to T4090	T0000 to T4094
Counter Area	---	---	C0000 to C4090	C0000 to C4094
DM Area	---	---	D00000 to D32762	D00000 to D32766
EM Area without bank	---	---	---	---
EM Area with bank	---	---	---	---
Indirect DM/EM addresses in binary	---	---	@ D00000 to @ D32767	@ D00000 to @ D32767
Indirect DM/EM addresses in BCD	---	---	*D00000 to *D32767	*D00000 to *D32767
Constants	See description of operand.	See description of operand.	---	See description of operand.
Data Registers	---	---	---	---

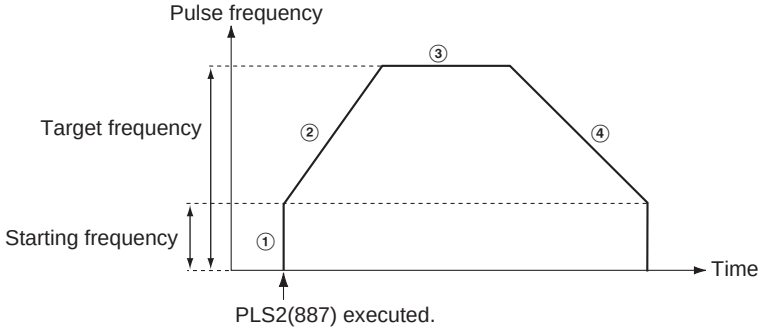
Area	P	M	S	F
Index Registers	---	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

PLS2(887) starts pulse output on the port specified in P using the mode specified in M at the start frequency specified in F (1 in diagram). The frequency is increased every pulse control period (4 ms) at the acceleration rate specified in S until the target frequency specified in S is reached (2 in diagram). When the target frequency has been reached, acceleration is stopped and pulse output continues at a constant speed (3 in diagram).

The deceleration point is calculated from the number of output pulses and deceleration rate set in S and when that point is reached, the frequency is decreased every pulse control period (4 ms) at the deceleration rate specified in S until the starting frequency specified in S is reached, at which point pulse output is stopped (4 in diagram).

Pulse output is started each time PLS2(887) is executed. It is thus normally sufficient to use the differentiated version (@PLS2(887)) of the instruction or an execution condition that is turned ON only for one scan.



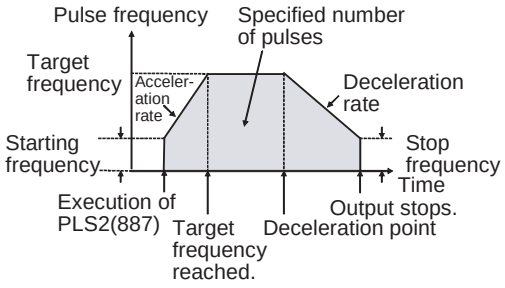
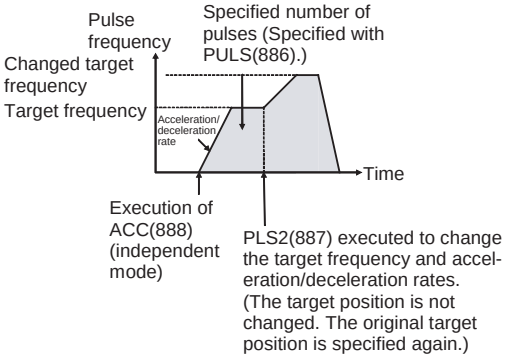
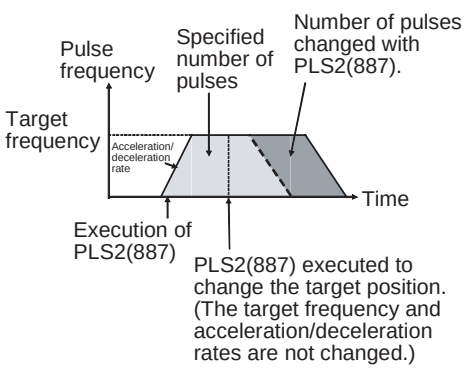
PLS2(887) can be used only for positioning.

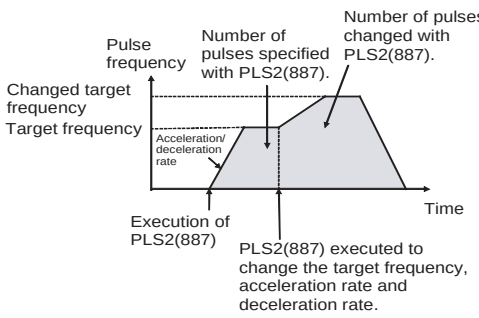
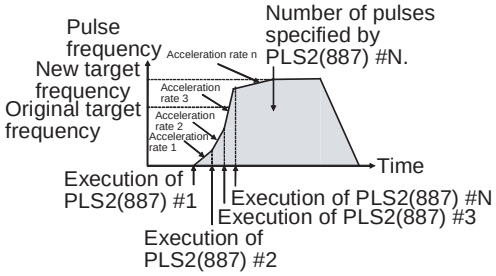
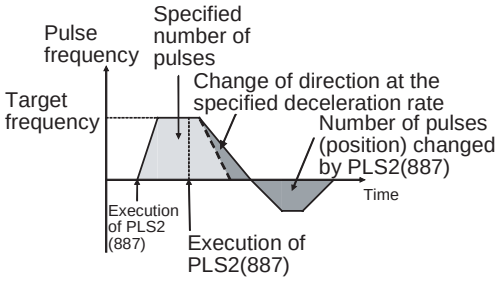
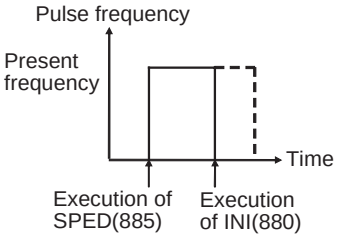
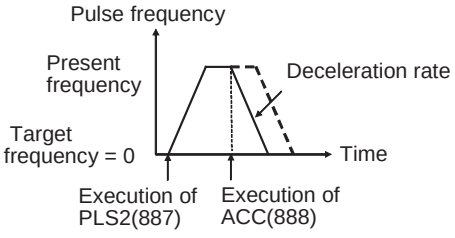
With the CJ1M CPU Units, PLS2(887) can be executed during pulse output for ACC(888) in either independent or continuous mode, and during acceleration, constant speed, or deceleration. (See note.) ACC(888) can also be executed during pulse output for PLS2(887) during acceleration, constant speed, or deceleration.

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.

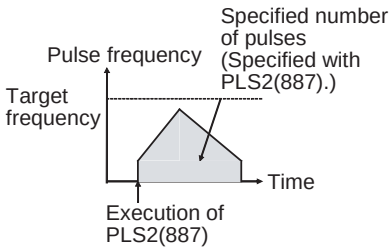
■ Independent Mode Positioning

Note Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.

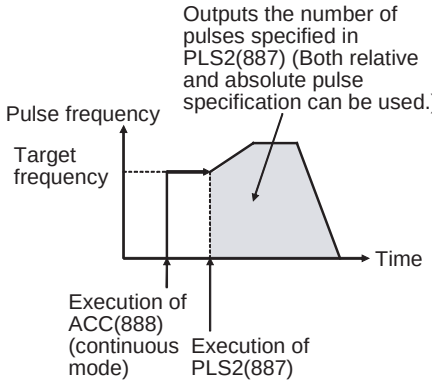
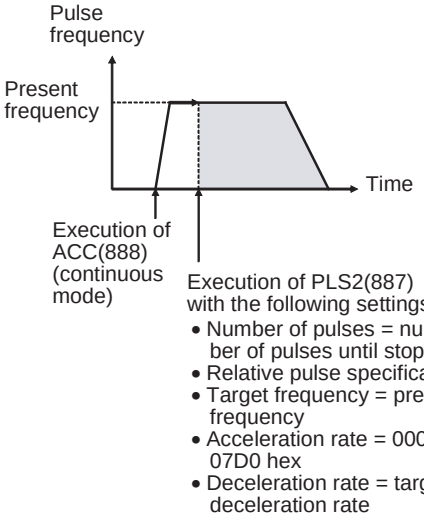
Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	Complex trapezoidal control	Positioning with trapezoidal acceleration and deceleration (Separate rates used for acceleration and deceleration; starting speed) The number of pulses can be changed during positioning.		Accelerates and decelerates at a fixed rates. The pulse output is stopped when the specified number of pulses has been output. (See note.) **Note** The target position (specified number of pulses) can be changed during positioning.	PLS2(887)
Changing settings	To change speed smoothly (with unequal acceleration and deceleration rates)	Changing the target speed (frequency) during positioning (different acceleration and deceleration rates)		PLS2(887) can be executed during positioning to change the acceleration rate, deceleration rate, and target frequency. **Note** To prevent the target position from being changed intentionally, the original target position must be specified in absolute coordinates.	PLS2(887) ↓ PLS2(887) PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)
	To change target position	Changing the target position during positioning (multiple start function)		PLS2(887) can be executed during positioning to change the target position (number of pulses), acceleration rate, deceleration rate, and target frequency. **Note** If a constant speed cannot be maintained after changing the settings, an error will occur and the original operation will continue to the original target position.	PLS2(887) ↓ PLS2(887) PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Changing settings, continued	To change target position and speed smoothly	Changing the target position and target speed (frequency) during positioning (multiple start function)		PLS2(887) can be executed during positioning to change the target position (number of pulses), acceleration rate, deceleration rate, and target frequency. **Note** If a constant speed cannot be maintained after changing the settings, an error will occur and the original operation will continue to the original target position.	PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)
		Changing the acceleration and deceleration rates during positioning (multiple start function)		PLS2(887) can be executed during positioning (acceleration or deceleration) to change the acceleration rate or deceleration rate.	PLS2(887) ↓ PLS2(887) PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)
	To change direction	Changing the direction during positioning		PLS2(887) can be executed during positioning with absolute pulse specification to change to absolute pulses and reverse direction.	PLS2(887) ↓ PLS2(887) PULS(886) ↓ ACC(888) (Independent) ↓ PLS2(887)
Stopping pulse output	Stop pulse output (Number of pulses setting is not preserved.)	Immediate stop		Stops the pulse output immediately and clears the number of output pulses.	PLS2(887) ↓ INI(880)
	Stop pulse output smoothly. (Number of pulses setting is not preserved.)	Decelerate to a stop		Decelerates the pulse output to a stop.	PLS2(887) ↓ ACC(888) (Independent, target frequency of 0 Hz)

Note Triangular Control
If the specified number of pulses is less than the number required to reach the target frequency and return to zero, the function will automatically reduce the acceleration/deceleration time and perform triangular control (acceleration and deceleration only.) An error will not occur.



■ Switching from Continuous Mode Speed Control to Independent Mode Positioning

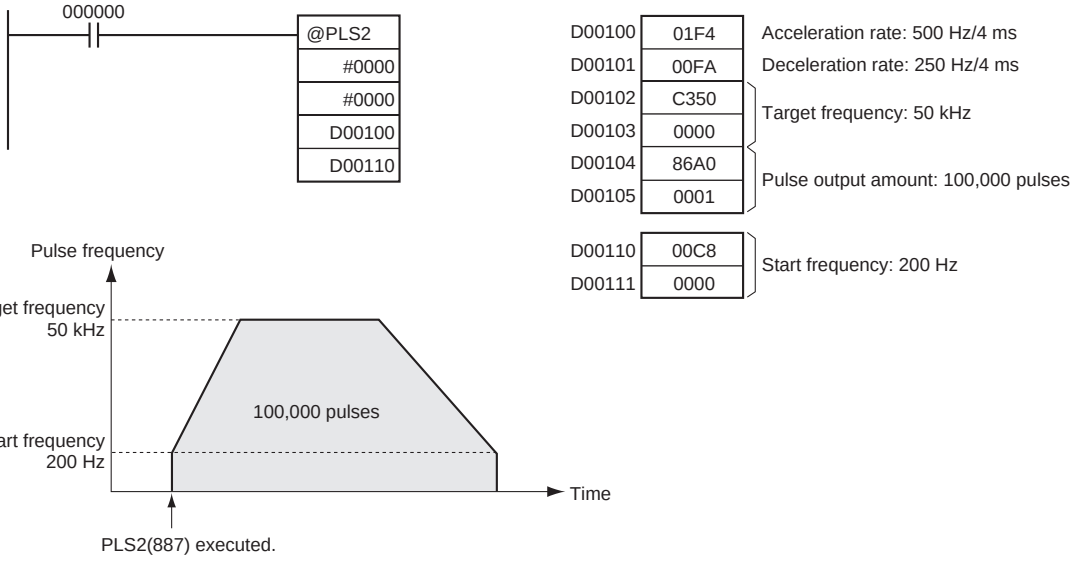
Example application	Frequency changes	Description	Procedure/ instruction
Change from speed control to fixed distance positioning during operation		PLS2(887) can be executed during a speed control operation started with ACC(888) to change to positioning operation.	ACC(888) (Continuous) ↓ PLS2(887)
Fixed distance feed interrupt			

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P, M, S, or F is exceeded. ON if PLS2(887) is executed for a port that is already outputting pulses for SPED(885) or ORG(889). ON if PLS2(887) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task. ON if PLS2(887) is executed for an absolute pulse output but the origin has not been established.

Example

When CIO 000000 turns ON in the following programming example, PLS2(887) starts pulse output from pulse output 0 with an absolute pulse specification of 100,000 pulses. Pulse output is accelerated at a rate of 500 Hz every 4 ms starting at 200 Hz until the target speed of 50 kHz is reached. From the deceleration point, the pulse output is decelerated at a rate of 250 Hz every 4 ms starting until the starting speed of at 200 Hz is reached, at which point pulse output is stopped.



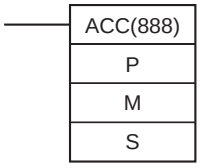
3-21-8 ACCELERATION CONTROL: ACC(888) (CJ1M-CPU21/22/23 Only)

Purpose

ACC(888) outputs pulses to the specified output port at the specified frequency using the specified acceleration and deceleration rate. (Acceleration rate is the same as the deceleration rate.) Either independent mode positioning or constant mode speed control is possible. For positioning, ACC(888) is used in combination with PULS(886). ACC(888) can also be executed during pulse output to change the target frequency or acceleration/deceleration rate, enabling smooth (sloped) speed changes.

This instruction is supported by CJ1M-CPU21/22/23 CPU Units only.

Ladder Symbol



P: Port specifier
M: Output mode
S: First word of settings table

Variations

Variations		
	Executed Each Cycle for ON Condition	ACC(888)
	Executed Once for Upward Differentiation	@ACC(888)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

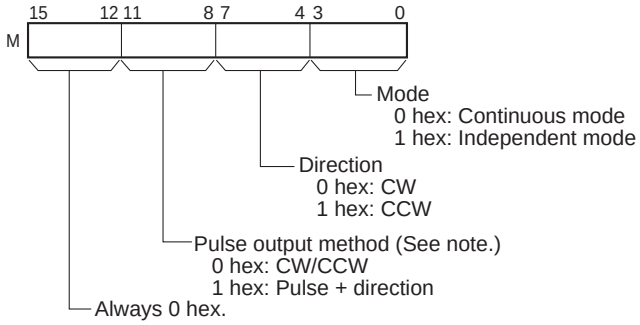
P: Port Specifier

The port specifier specifies the port where the pulses will be output.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1

M: Output Mode

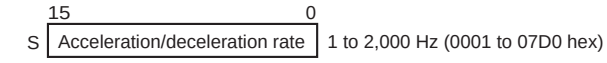
The content of M specifies the parameters for the pulse output as follows:



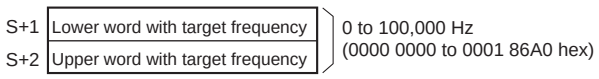
Note: Use the same pulse output method when using both pulse outputs 0 and 1.

S: First Word of Settings Table

The content of S to S+2 controls the pulse output as shown in the following diagrams.



Specify the increase or decrease in the frequency per pulse control period (4 ms).



Specify the frequency after acceleration in Hz.

Operand Specifications

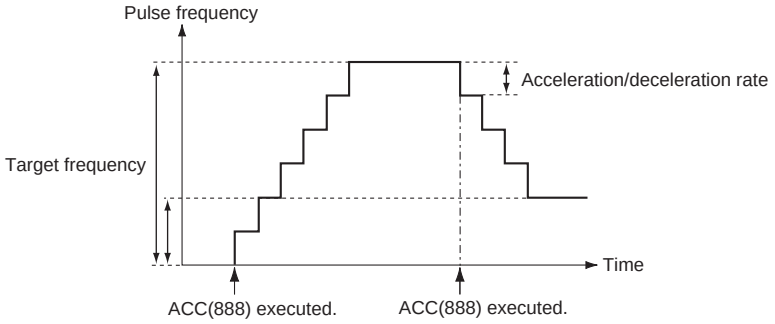
Area	P	M	S
CIO Area	---	---	CIO 0000 to CIO 6141
Work Area	---	---	W000 to W509
Holding Bit Area	---	---	H000 to H509
Auxiliary Bit Area	---	---	A448 to A957
Timer Area	---	---	T0000 to T4093
Counter Area	---	---	C0000 to C4093
DM Area	---	---	D00000 to D32765
EM Area without bank	---	---	---
EM Area with bank	---	---	---

Area	P	M	S
Indirect DM/EM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM/EM addresses in BCD	---	---	*D00000 to *D32767
Constants	See description of operand.	See description of operand.	---
Data Registers	---	---	---
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

ACC(888) starts pulse output on the port specified in P using the mode specified in M using the target frequency and acceleration/deceleration rate specified in S. The frequency is increased every pulse control period (4 ms) at the acceleration rate specified in S until the target frequency specified in S is reached.

Pulse output is started each time ACC(888) is executed. It is thus normally sufficient to use the differentiated version (@ACC(888)) of the instruction or an execution condition that is turned ON only for one scan.



In independent mode, pulse output stops automatically when the specified number of pulses has been output. In continuous mode, pulse output continues until it is stopped from the program.

An error will occur if an attempt is made to switch between independent and continuous mode during pulse output.

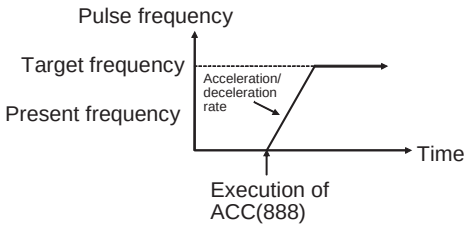
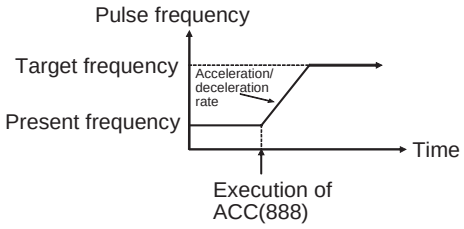
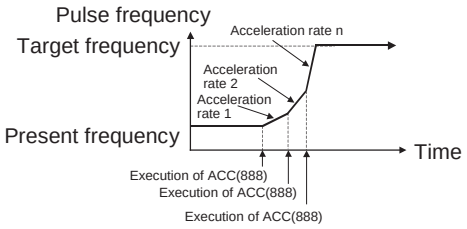
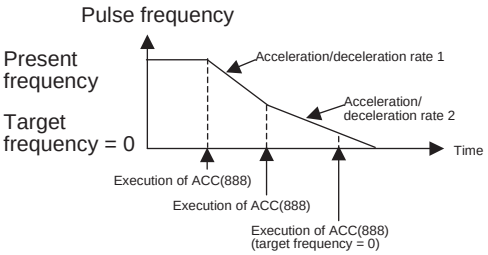
With the CJ1M CPU Units, PLS2(887) can be executed during pulse output for ACC(888) in either independent or continuous mode, and during acceleration, constant speed, or deceleration. (See note.) ACC(888) can also be executed during pulse output for PLS2(887) during acceleration, constant speed, or deceleration.

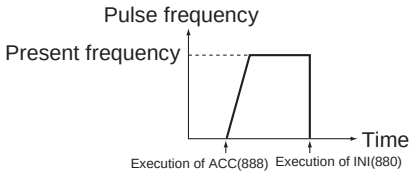
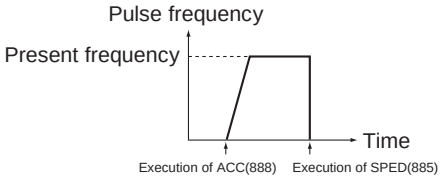
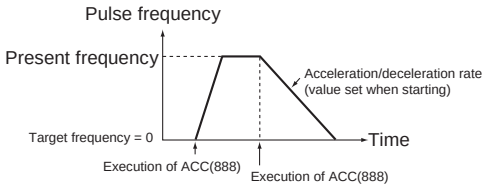
Note Executing PLS2(887) during speed control with ACC(888) (continuous mode) with the same target frequency as ACC(888) can be used to achieved interrupt feeding of a fixed distance. Acceleration will not be performed by PLS2(887) for this application, but if the acceleration rate is set to 0, the Error 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 deceleration rate is changed while decelerating. **Note** If the target frequency is set to 0 Hz, the current deceleration rate will be used.	ACC(888) (Continuous) ↓ ACC(888) (Continuous) ↓ACC(888) (Continuous, target frequency of 0 Hz)

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Stopping pulse output	To stop pulse output	Immediate stop		Immediately stops pulse output.	ACC(888) (Continuous) ↓ INI(880) (Continuous)
	To stop pulse output	Immediate stop		Immediately stops pulse output.	ACC(888) (Continuous) ↓ SPED(885) (Continuous, target frequency of 0)
	To stop pulse output smoothly	Decelerating to a stop		Decelerated pulse output to a stop. **Note** If the target frequency of the second ACC(888) instruction is 0 Hz, the deceleration rate from the first ACC(888) instruction will be used.	ACC(888) (Continuous) ↓ ACC(888) (Continuous, target frequency of 0)

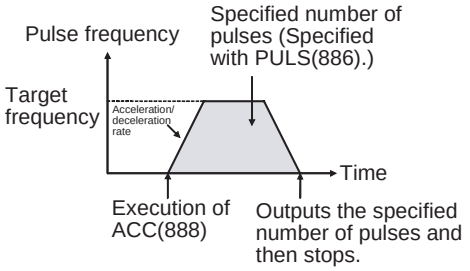
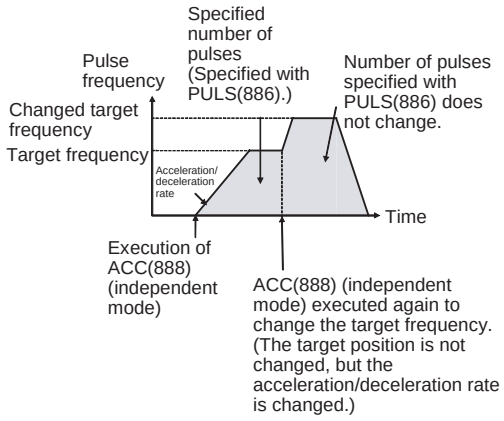
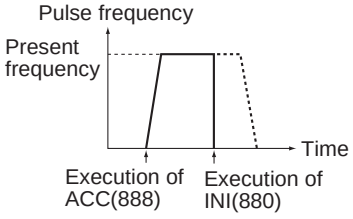
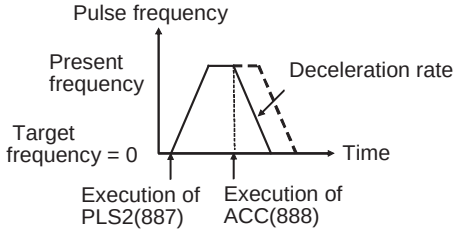
■ Independent Mode Positioning

When independent mode operation is started, pulse output will be continued until the specified number of pulses has been output.

The deceleration point is calculated from the number of output pulses and deceleration rate set in S and when that point is reached, the frequency is decreased every pulse control period (4 ms) at the deceleration rate specified in S until the specified number of points has been output, at which point pulse output is stopped.

Note

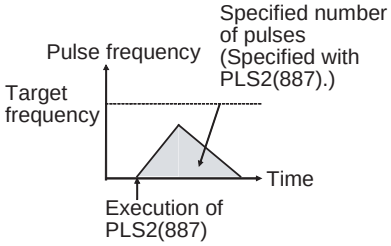
1. Pulse output will stop immediately if the CPU Unit is changed to PROGRAM mode.
2. The number of output pulses must be set each time output is restarted.
3. The number of output pulses must be set in advance with PULS(881). Pulses will not be output for ACC(888) if PULS(881) is not executed first.
4. The direction set in the ACC(888) operand will be ignored if the number of pulses is set with PULS(881) as an absolute value.

Operation	Purpose	Application	Frequency changes	Description	Procedure/ instruction
Starting pulse output	Simple trapezoidal control	Positioning with trapezoidal acceleration and deceleration (Same rate used for acceleration and deceleration; no starting speed) The number of pulses cannot be changed during positioning.		Accelerates and decelerates at the same fixed rate and stops immediately when the specified number of pulses has been output. (See note.) **Note** The target position (specified number of pulses) cannot be changed during positioning.	PULS(886) ↓ ACC(888) (Independent)
Changing settings	To change speed smoothly (with the same acceleration and deceleration rates)	Changing the target speed (frequency) during positioning (acceleration rate = deceleration rate)		ACC(888) can be executed during positioning to change the acceleration/deceleration rate and target frequency. The target position (specified number of pulses) is not changed.	PULS(886) ↓ ACC(888) or SPED(885) (Independent) ↓ ACC(888) (Independent)
Stopping pulse output	To stop pulse output. (Number of pulses setting is not preserved.)	Immediate stop		Pulse output is stopped immediately and the remaining number of output pulses is cleared.	PULS(886) ↓ ACC(888) (Independent) ↓ INI(880)
	To stop pulse output smoothly. (Number of pulses setting is not preserved.)	Decelerating to a stop		Decelerates the pulse output to a stop. **Note** If ACC(888) started the operation, the original acceleration/ deceleration rate will remain in effect. If SPED(885) started the operation, the acceleration/ deceleration rate will be invalid and the pulse output will stop immediately.	PULS(886) ↓ ACC(888) or SPED(885) (Independent) ↓ ACC(888) (Independent, independent, target frequency of 0) PLS2(887) ↓ ACC(888) (Independent, target frequency of 0)

Note Triangular Control

If the specified number of pulses is less than the number required to reach the target frequency and return to zero, the function will automatically reduce the acceleration/deceleration time and perform triangular control (acceleration

and deceleration only.) An error will not occur.

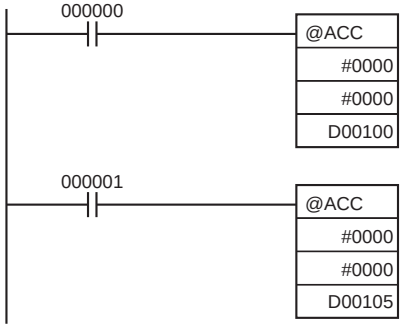


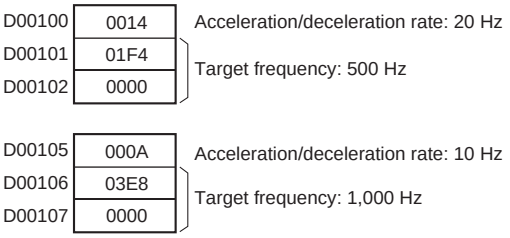
Flags

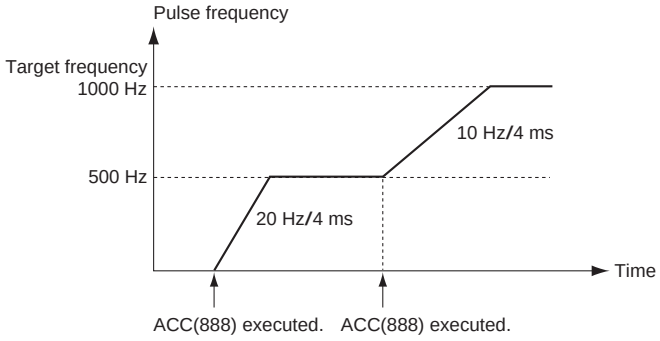
Name	Label	Operation
Error Flag	ER	ON if the specified range for P, M, or S is exceeded. ON if pulses are being output using ORG(889) for the specified port. ON if ACC(888) is executed to switch between independent and continuous mode for a port that is outputting pulses for SPED(885), ACC(888), or PLS2(887). ON if ACC(888) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task. ON if ACC(888) is executed for an absolute pulse output in independent mode but the origin has not been established.

Example

When CIO 000000 turns ON in the following programming example, ACC(888) starts pulse output from pulse output 0 in continuous mode in the clockwise direction using the CW/CCW method. Pulse output is accelerated at a rate of 20 Hz every 4 ms until the target frequency of 500 Hz is reached. When CIO 000001 turns ON, ACC(888) changes to an acceleration rate of 10 Hz every 4 ms until the target frequency of 1,000 Hz is reached.







3-21-9 ORIGIN SEARCH: ORG(889) (CJ1M-CPU21/22/23 Only)

Purpose

ORG(889) performs an origin search or origin return operation.

This instruction is supported by CJ1M-CPU21/22/23 CPU Units only.

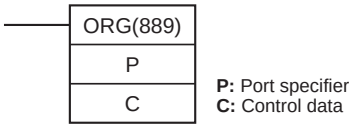
■ Origin Search

Pulses are output using the specified method to actually drive the motor and establish the origin based on origin proximity input and origin input signals.

■ Origin Return

The positioning system is returned to the pre-established origin.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	ORG(889)
	Executed Once for Upward Differentiation	@ORG(889)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

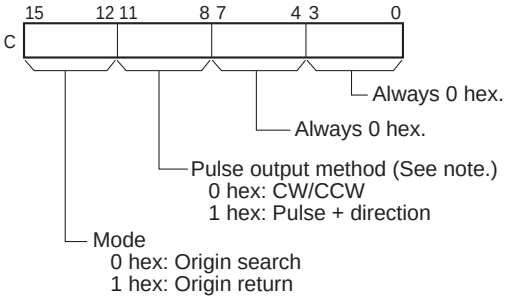
P: Port Specifier

The port specifier specifies the port where the pulses will be output.

P	Port
0000 hex	Pulse output 0
0001 hex	Pulse output 1

C: Control Data

The value of C determines the origin search method.



Note: Use the same pulse output method when using both pulse outputs 0 and 1.

Operand Specifications

Area	P	C
CIO Area	---	---
Work Area	---	---
Holding Bit Area	---	---
Auxiliary Bit Area	---	---
Timer Area	---	---

Area	P	C
Counter Area	---	---
DM Area	---	---
EM Area without bank	---	---
EM Area with bank	---	---
Indirect DM/EM addresses in binary	---	---
Indirect DM/EM addresses in BCD	---	---
Constants	See description of operand.	See description of operand.
Data Registers	---	---
Index Registers	---	---
Indirect addressing using Index Registers	---	---

Description

ORG(889) performs an origin search or origin return operation for the port specified in P using the method specified in C.

The following parameters must be set in the PLC Setup before ORG(889) can be executed. Refer to the *CJ-series Built-in I/O Operation Manual* for details.

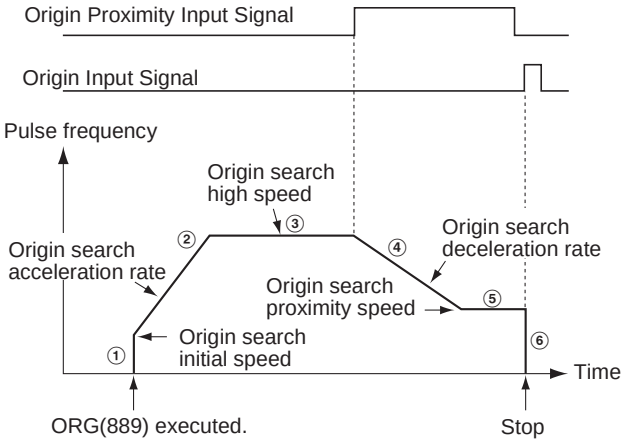
Origin search	Origin return
Origin Search Function Enable/Disable	Origin Search/Return Initial Speed
Origin Search Operating Mode	Origin Return Target Speed
Origin Search Operation Setting	Origin Return Acceleration Rate
Origin Detection Method	Origin Return Deceleration Rate
Origin Search Direction Setting	
Origin Search/Return Initial Speed	
Origin Search High Speed	
Origin Search Proximity Speed	
Origin Compensation	
Origin Search Acceleration Rate	
Origin Search Deceleration Rate	
Limit Input Signal Type	
Origin Proximity Input Signal Type	
Origin Input Signal Type	

An origin search or origin return is started each time ORG(889) is executed. It is thus normally sufficient to use the differentiated version (@ORG(889)) of the instruction or an execution condition that is turned ON only for one scan.

■ Origin Search (Bits 12 to 15 of C = 0 hex)

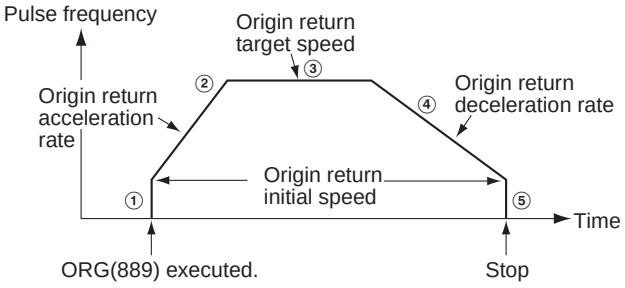
ORG(889) starts outputting pulses using the specified method at the Origin Search Initial Speed (1 in diagram). Pulse output is accelerated to the Origin Search High Speed using the Origin Search Acceleration Rate (2 in diagram). Pulse output is then continued at constant speed until the Origin Proximity Input Signal turns ON (3 in diagram), from which point pulse output is decelerated to the Origin Search Proximity Speed using the Origin Search Deceleration Rate (4 in diagram). Pulses are then output at constant speed until the Origin Input Signal turns ON (5 in diagram). Pulse output is stopped when the Origin Input Signal turns ON (6 in diagram).

When the origin search operation has been completed, the Error Counter Reset Output will be turned ON. The above operation, however, depends on the operating mode, origin detection method, and other parameters. Refer to the *CJ-series Built-in I/O Operation Manual* for details.



■ **Origin Return (Bits 12 to 15 of C = 1 hex)**

ORG(889) starts outputting pulses using the specified method at the Origin Return Initial Speed (1 in diagram). Pulse output is accelerated to the Origin Return Target Speed using the Origin Return Acceleration Rate (2 in diagram) and pulse output is continued at constant speed (3 in diagram). The deceleration point is calculated from the number of pulses remaining to the origin and the deceleration rate and when that point is reached, the pulse output is decelerated (4 in diagram) at the Origin Return Deceleration Rate until the Origin Return Start Speed is reached, at which point pulse output is stopped at the origin (5 in diagram).

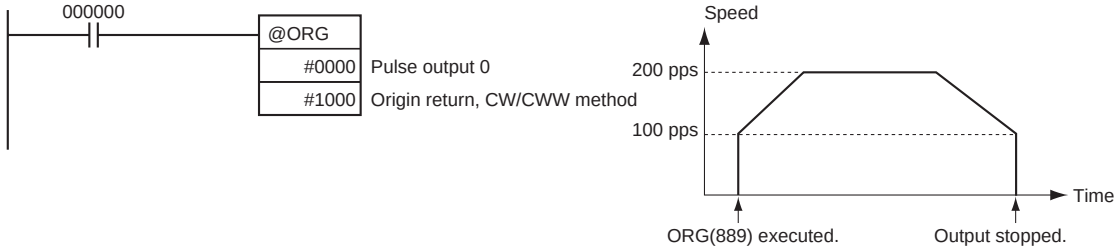


Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P or C is exceeded. ON if ORG(889) is specified for a port during pulse output for SPED(885), ACC(888), or PLS2(887). ON if ORG(889) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task. ON if the origin search or origin return parameters set in the PLC Setup are not within range. ON if the Origin Search High Speed is less than or equal to the Origin Search Proximity Speed or the Origin Search Proximity Speed is less than or equal to the Origin Search Initial Speed. ON if the Origin Return Target speed is less than or equal to the Origin Return Initial Speed. ON if an origin return operation is attempted when the origin has not been established.

Example

When CIO 000000 turns ON in the following programming example, ORG(889) starts an origin return operation for pulse output 0 by outputting pulses using the CW/CCW method. According to the PLC Setup, the initial speed is 100 pps, the target speed is 200 pps, and the acceleration and deceleration rates are 50 Hz/4 ms.



The PLC Setup parameters are as follows:

Parameter	Setting
Pulse Output 0 Starting Speed for Origin Search and Origin Return	0000 0064 hex: 100 pps
Pulse Output 0 Origin Return Target Speed	0000 00C8 hex: 200 pps
Pulse Output 0 Origin Return Acceleration Rate	0032 hex: 50 hex/4 ms
Pulse Output 0 Origin Return Deceleration Rate	0032 hex: 50 hex/4 ms

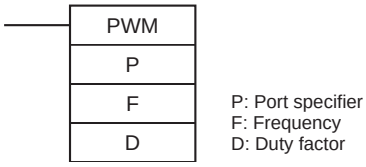
3-21-10 PULSE WITH VARIABLE DUTY FACTOR: PWM(891) (CJ1M-CPU21/22/23 Only)

Purpose

PWM(891) is used to output pulses with the specified duty factor from the specified port.

This instruction is supported by CJ1M-CPU21/22/23 CPU Units only.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	PWM(891)
	Executed Once for Upward Differentiation	@PWM(891)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

P: Port Specifier

The port specifier specifies the port where the pulses will be output.

P	Port
0000 hex	Pulse output 0 (duty factor: in increments of 1%)
0001 hex	Pulse output 1 (duty factor: in increments of 1%)
1000 hex (CJ1M CPU Unit Ver. 2.0 only)	Pulse output 0 (duty factor: in increments of 0.1%)
1001hex (CJ1M CPU Unit Ver. 2.0 only)	Pulse output 1 (duty factor: in increments of 0.1%)

F: Frequency

F specifies the frequency of the pulse output between 0.1 and 6,553.5 Hz (0.1 Hz units, 0001 to FFFF hex). The accuracy of the PMW(891) waveform that is actually output (ON duty +5%/–0%) applies only to 0.1 to 1,000.0 Hz due to limitations in the output circuits.

D: Duty Factor

D specifies the duty factor of the pulse output, i.e., the percentage of time that the output is ON. The value of D must be between the following range.

- Pre-Ver. 2.0 CJ1m CPU Units
  0% and 100% (1% units, 0000 to 0064 hex)
- Ver. 2.0 CJ1m CPU Units
  0.0% and 100.0% (0.1% units, 0000 to 03E8 hex)

Operand Specifications

Area	P	F	D
CIO Area	---	CIO 0000 to CIO 6143	CIO 0000 to CIO 6143
Work Area	---	W000 to W511	W000 to W511
Holding Bit Area	---	H000 to H511	H000 to H511
Auxiliary Bit Area	---	A448 to A959	A448 to A959
Timer Area	---	T0000 to T4095	T0000 to T4095
Counter Area	---	C0000 to C4095	C0000 to C4095
DM Area	---	D00000 to D32767	D00000 to D32767
EM Area without bank	---	---	---
EM Area with bank	---	---	---
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767	@ D00000 to @ D32767
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767	*D00000 to *D32767
Constants	See description of operand.	0000 to FFFF hex	0000 to 0064 hex
Data Registers	---	DR0 to DR15	DR0 to DR15
Index Registers	---	---	---
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

PWM(891) outputs the frequency specified in F at the duty factor specified in D from the port specified in P. PWM(891) can be executed during duty-factor pulse output to change the duty factor without stopping pulse output. Any attempts to change the frequency will be ignored.

Pulse output is started each time PWM(891) is executed. It is thus normally sufficient to use the differentiated version (@PWM(891)) of the instruction or an execution condition that is turned ON only for one scan.

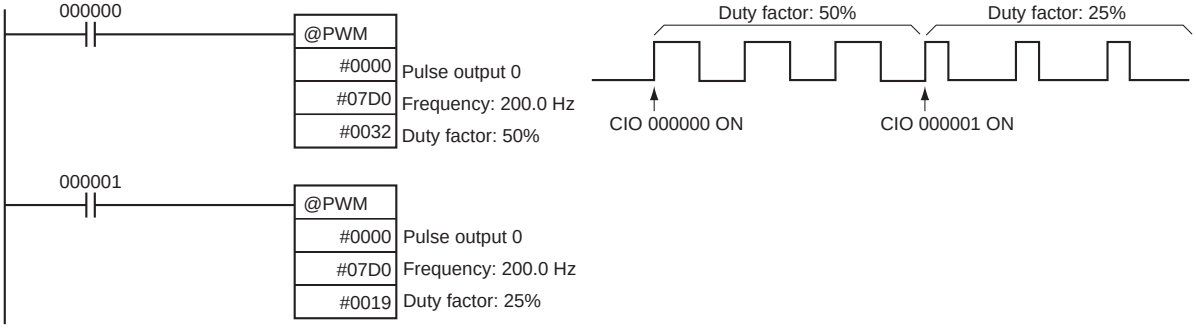
The pulse output will continue either until INI(880) is executed to stop it (C = 0003 hex: stop pulse output) or until the CPU Unit is switched to PROGRAM mode.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range for P, F, or D is exceeded. ON if pulses are being output using ORG(889) for the specified port. ON if PWM(891) is executed in an interrupt task when an instruction controlling pulse output is being executed in a cyclic task.

Example

When CIO 000000 turns ON in the following programming example, PWM(891) starts pulse output from pulse output 0 at 200 Hz with a duty factor of 50%. When CIO 000001 turns ON, the duty factor is changed to 25%.



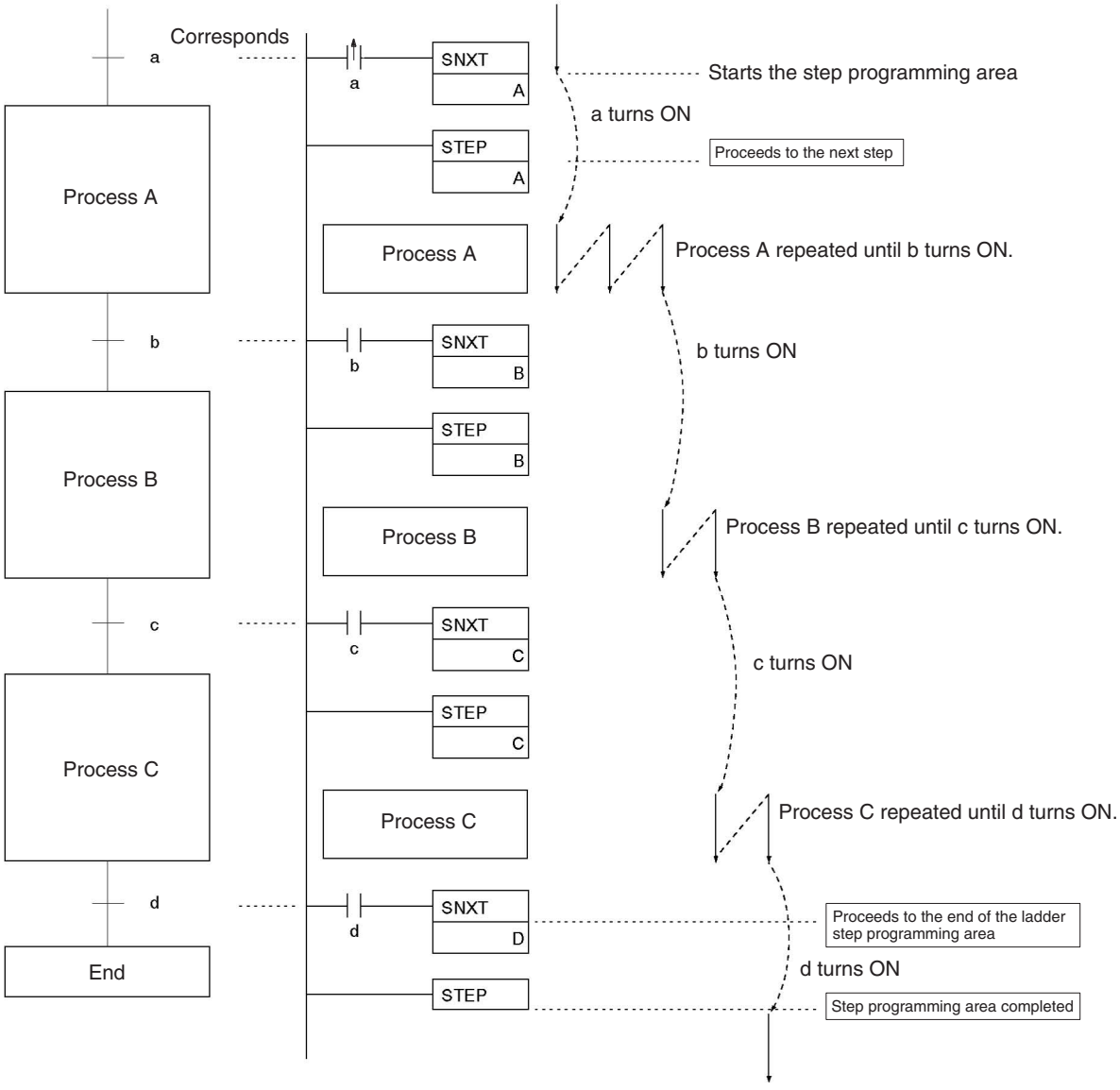
3-22 Step Instructions

This section describes Step Instructions, which are used to set up break points between sections in a large program so that the sections can be executed as units and reset upon completion.

Instruction	Mnemonic	Function code	Page
STEP DEFINE	STEP	008	909
STEP START	SNXT	009	909

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.	Corresponds
STEP(008): STEP DEFINE	Indicates the start of a step. Repeats the same step program until the conditions for progression to the next step are established.	Corresponds



Note Work bits are used as the control bits for A, B, C and D.

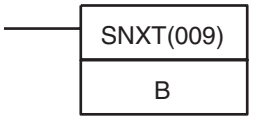
3-22-1 STEP DEFINE and STEP START: STEP(008)/SNXT(009)

Purpose

SNXT(009) is placed immediately before the STEP(008) instruction and controls step execution by turning the specified control bit ON. If there is another step immediately before SNXT(009), it also turns OFF the control bit of that process.

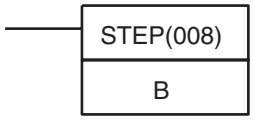
STEP(008) is placed immediately after the SNXT(009) instruction and before each process. It defines the start of each process and specified the control bit for it. It is also placed at the end of the step programming area after the last SNXT(009) to indicate the end of the step programming area. When it appears at the end of the step programming area, STEP(008) does not take a control bit.

Ladder Symbols



B: Bit

When defining the beginning of a step, a control bit is specified as follows:



B: Bit

When defining the end of a step a control bit is not specified as follows:

Variations

Variations		
	Executed Each Cycle for ON Condition	STEP(008)/ SNXT(009)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operand Specifications

Area	B
CIO Area	---
Work Area	W00000 to W51115
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

SNXT(009)

SNXT(009) is used in the following three ways:

1,2,3...

1. To start step programming execution.
2. To proceed to the next step control bit.
3. To end step programming execution.

The step programming area is from the first STEP(008) instruction (which always takes a control bit) to the last STEP(008) instruction (which never takes a control bit).

Starting Step Execution

SNXT(009) is placed at the beginning of the step programming area to start step execution. It turns ON the control bit specified for B for the next STEP(008) and proceeds to step B (all instructions after STEP(008) B). A differentiated execution condition must be used for the SNXT(009) instruction that starts step programming area execution, or step execution will last for only one cycle.

Proceeding to the Next Step

When SNXT(009) occurs in the middle of the step programming area, it is used to proceed to the next step. It turns OFF the previous control bit and turns ON the next control bit B, for the next step, thereby starting step B (all instructions after STEP(008) B).

Ending the Step Programming Area

When SNXT(009) is placed at the very end of the step programming area, it ends step execution and turns OFF the previous control bit. The control bit specified for B is a dummy bit. This bit will however be turned ON, so be sure to select a bit that will not cause problems.

STEP(008)

STEP(008) functionS in following 2 ways, depending on its position and whether or not a control bit has been specified.

1,2,3... 1. Starts a specific step.

2. Ends the step programming area (i.e., step execution).

Starting a Step

STEP(008) is placed at the beginning of each step with an operand, B, that serves as the control bit for the step.

The control bit B will be turned ON by SNXT(009) and the instruction in the step will be executed from the one immediately following STEP(008). A20012 (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) (CJ1-H-R CPU Units only)	PV	Cannot be read.
	Completion Flag	OFF (reset)
Bits or words specified for other instructions (see note)		Holds the previous status (but the instructions are not executed)

Note Indicates all other instructions, such as TTIM(087), TTIMX(555), MTIM(543), MTIMX(554), SET, REST, CNT, CNTX(546), CNTR(012), CNTRX(548), SFT(010), and KEEP(011).

The STEP(008) instruction must be placed at the beginning of each step. STEP(008) is placed at the beginning of a step area to define the start of the step.

Ending the Step Programming Area

STEP(008) is placed at the end of the step programming area without an operand to define the end of step programming When the control bit preceding a SNXT(009) instruction is turned OFF, step execute is stopped by SNXT(009).

Flags:STEP(008)

Name	Label	Operation
Error Flag	ER	ON when the specified bit B is not in the WR area. ON when STEP(008) is used in an interrupt program. OFF in all other cases.

Flags:SNXT(009)

Name	Label	Operation
Error Flag	ER	ON when the specified bit B is not in the WR area. ON when SNXT(009) is used in an interrupt program. OFF in all other cases.

Precautions

The control bit, B, must be in the Work Area for STEP(008)/SNXT(009).

A control bit for STEP(008)/SNXT(009) cannot be use anywhere else in the ladder diagram. If the same bit is used twice, as duplication bit error will occur.

If SBS(091) is used to call a subroutine from within a step, the subroutine outputs and instructions will not be interlocked when the control bit turns OFF.

Control bits within one section of step programming must be sequential and from the same word.

SNXT(009) will be executed only once, i.e., on the rising edge of the execution condition.

Input SNXT(009) at the end of the step programming area and make sure that the control bit is a dummy bit in the Work Area. If a control bit for a step is used in the last SNXT(009) in the step programming area, the corresponding step will be started when SNXT(009) is executed.

An error will occur and the Error Flag will turn ON if the operand B specified for SNXT(009) or STEP(008) is not in the Work Area or if the step program has been placed anywhere but in a cyclic task.

A20012 (Step Flag) is turned ON for one cycle when STEP(008) is executed. This flag can be used to conduct initialization once the step execution has started.

Placement Conditions for Step Programming Areas (STEP B to STEP)

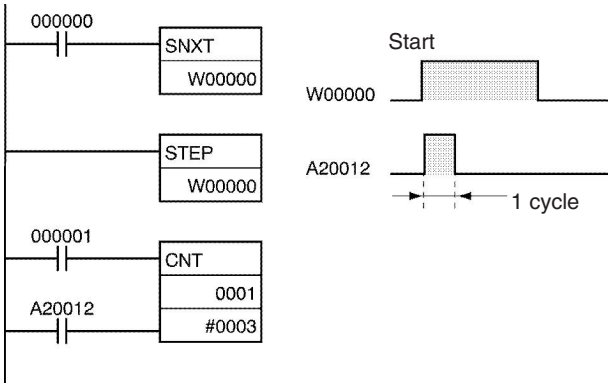
STEP(008) and SNXT(009) cannot be used inside of subroutines, interrupt programs, or block programs.

Be sure that two steps are not executed during the same cycle.

Instructions that Cannot be Used Within Step Programs

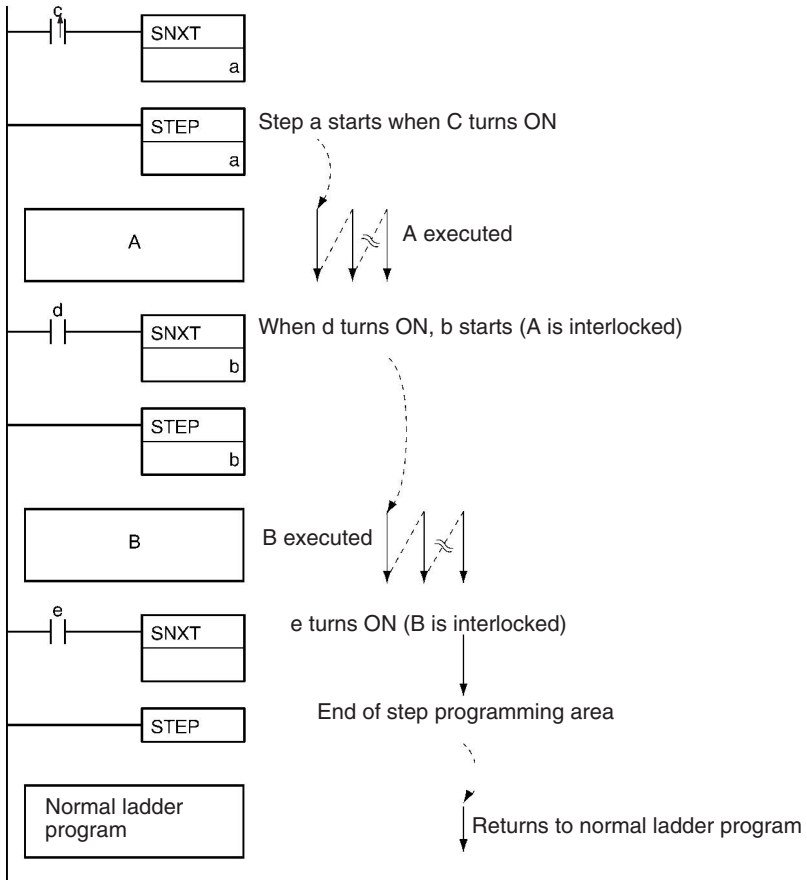
The instructions that cannot be used within step programs are listed in the following table.

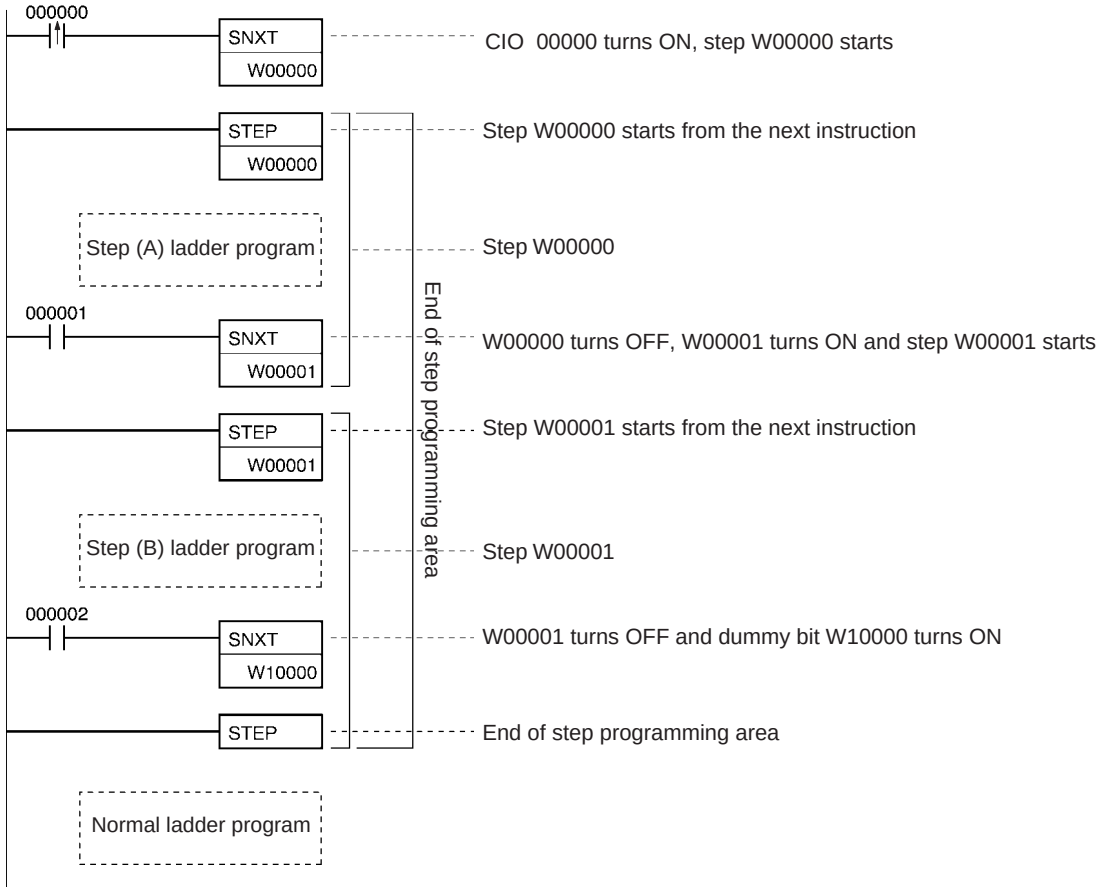
Function	Mnemonic	Name
Sequence Control Instructions	END(001)	END
	IL(002)	INTERLOCK
	ILC(003	INTERLOCK CLEAR
	JMP(004)	JUMP
	JME(005)	JUMP END
	CJP(510)	CONDITIONAL JUMP
	CJPN(511)	CONDITIONAL JUMP NOT
	JMP0(515)	MULTIPLE JUMP
	JME0(516)	MULTIPLE JUMP END
Subroutine Instructions	SBN(092)	SUBROUTINE ENTRY
	RET(093)	SUBROUTINE RETURN



Related Bits

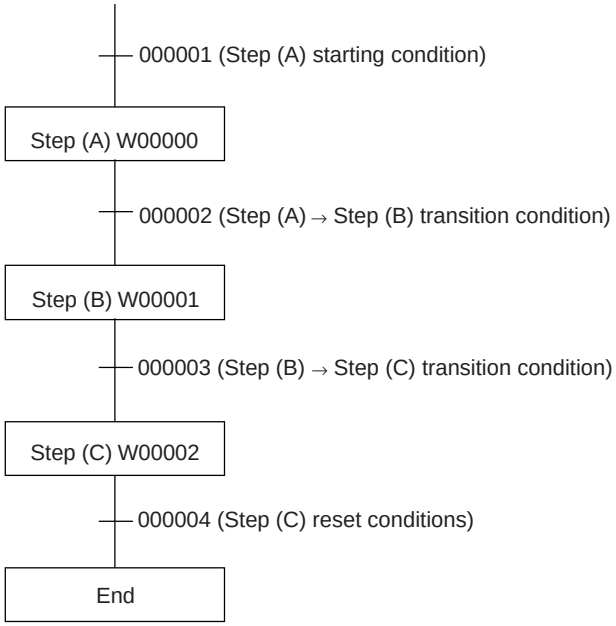
Name	Address	Details
Step Flag	A20012	ON for one cycle when a step program is started using STEP(008). Can be used to reset timers and perform other processing when starting a new step.

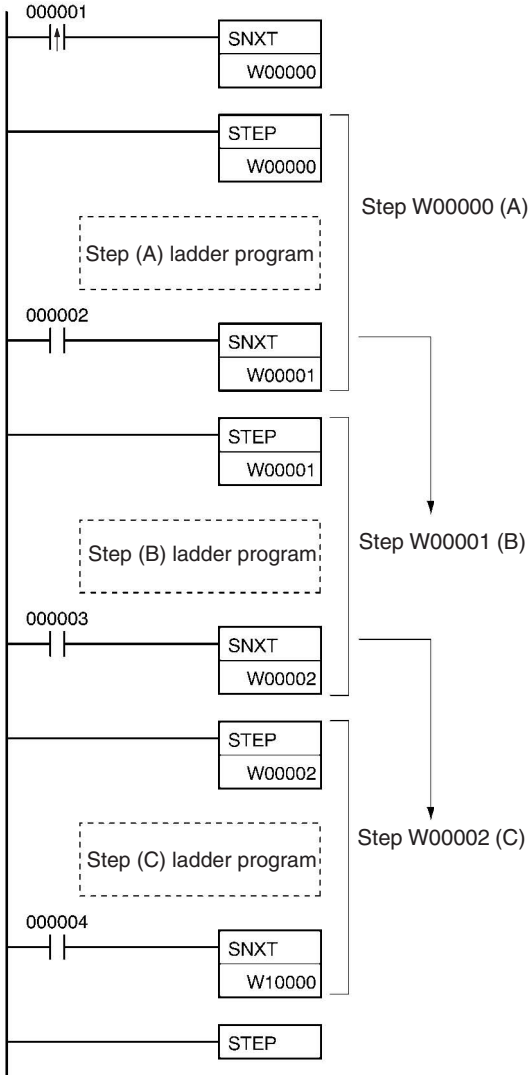

c
SNXT
a
STEP
a
Step a starts when C turns ON
A
A executed
d
SNXT
b
When d turns ON, b starts (A is interlocked)
STEP
b
B
B executed
e
SNXT
e turns ON (B is interlocked)
STEP
End of step programming area
Normal ladder program
Returns to normal ladder program



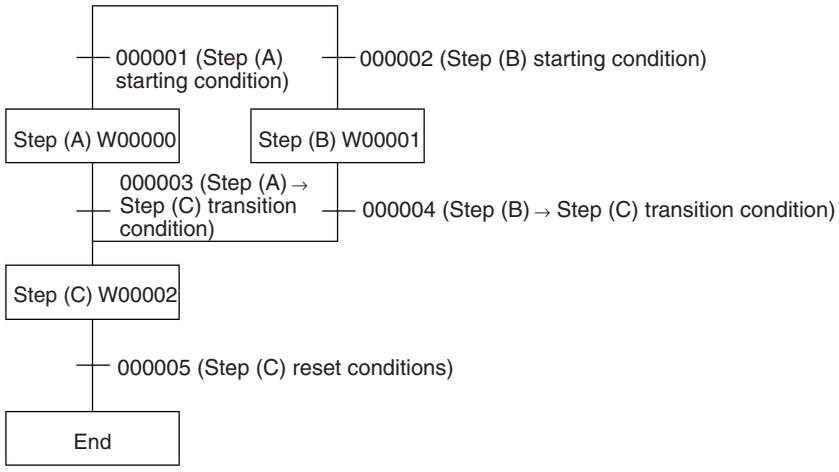
Examples

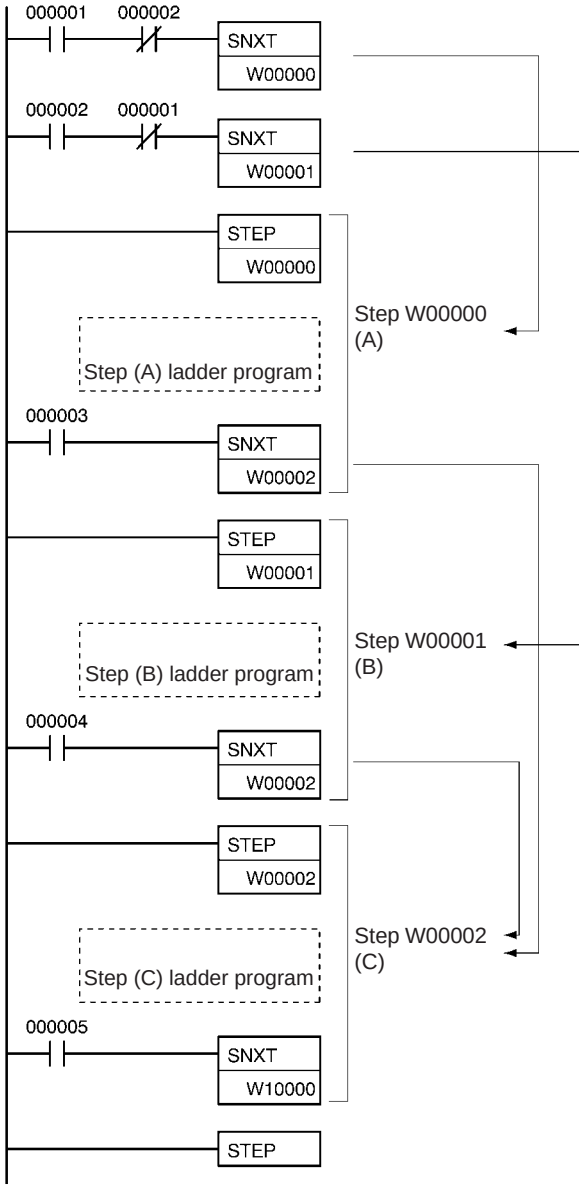
Sequential Control



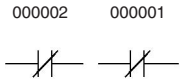


Branching Control



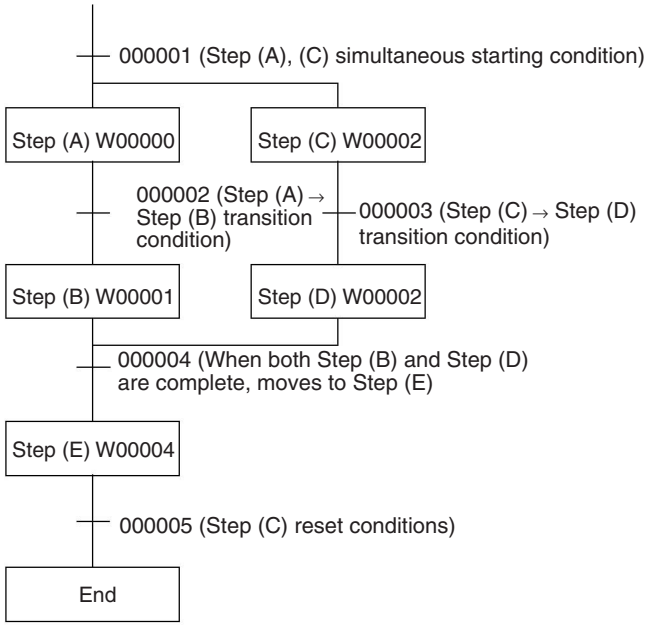


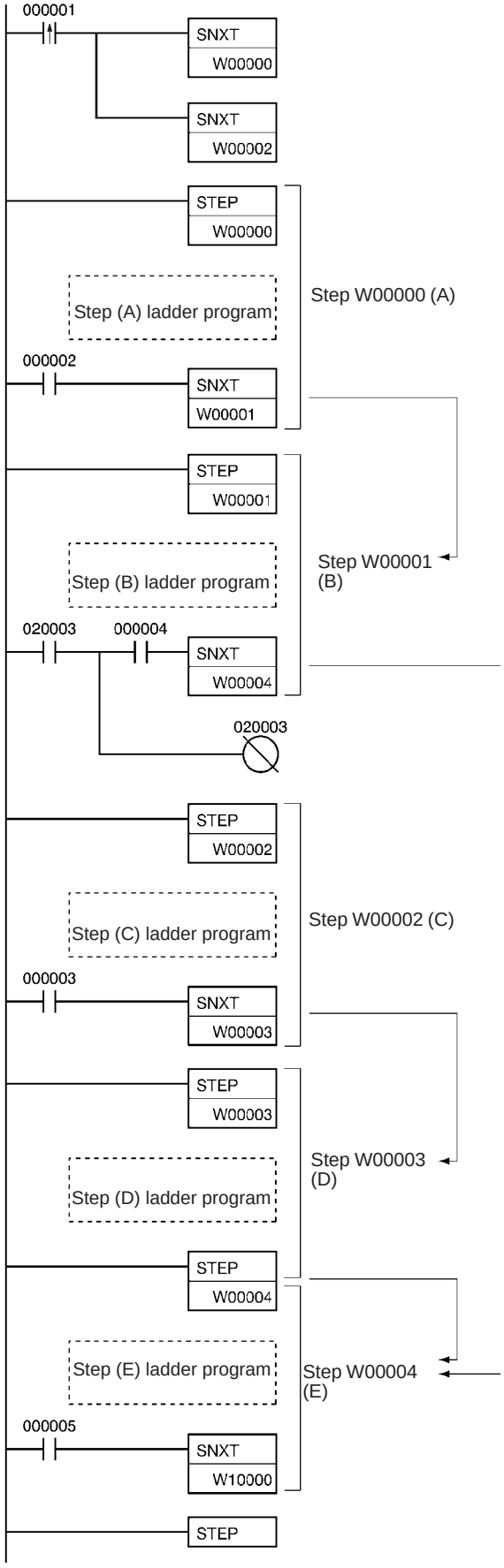
The above programming is used when steps A and B cannot be executed simultaneously. For simultaneous execution of A and B, delete the execution conditions illustrated below.



Note In the above example, where SNXT(009) is executed for W00002, the branching moves onto the next steps even though the same control bit is used twice. This is not picked up as an error in the program check using the CX-Programmer. A duplicate bit error will only occur in a step ladder program only when a control bit in a step instructions is also used in the normal ladder diagram.

Parallel Control



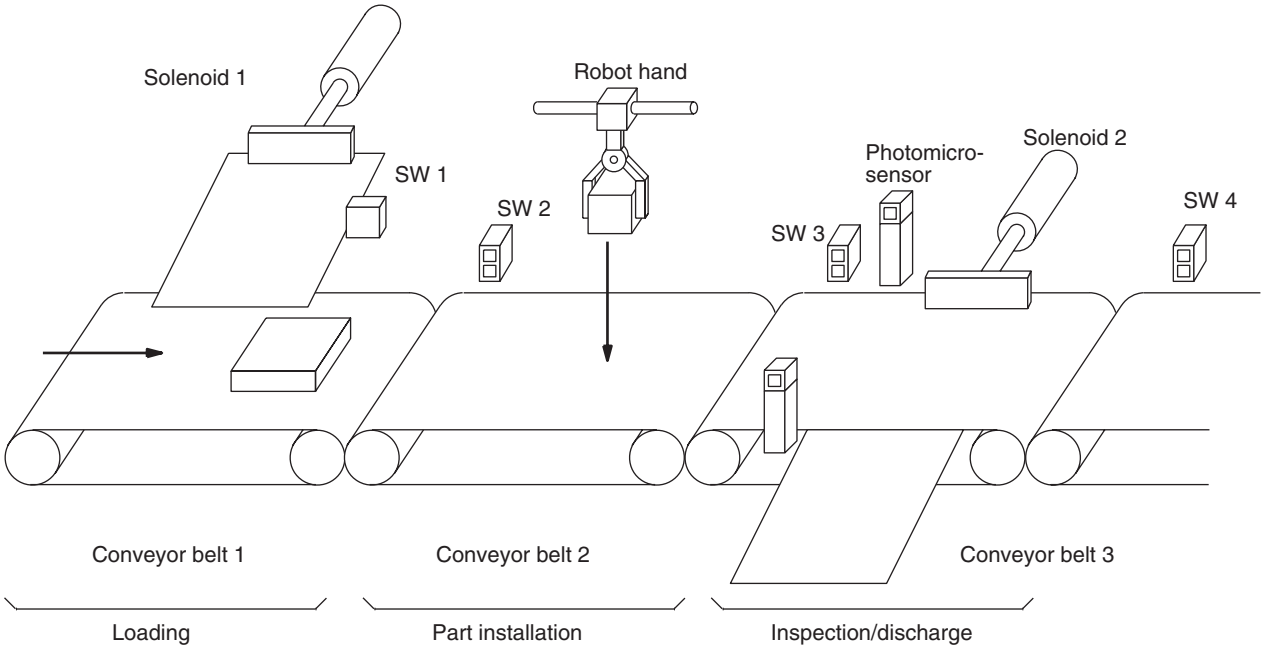

000001
SNXT
W00000
SNXT
W00002
STEP
W00000
Step (A) ladder program
Step W00000 (A)
000002
SNXT
W00001
STEP
W00001
Step (B) ladder program
Step W00001
(B)
020003
000004
SNXT
W00004
020003
STEP
W00002
Step (C) ladder program
Step W00002 (C)
000003
SNXT
W00003
STEP
W00003
Step (D) ladder program
Step W00003
(D)
STEP
W00004
Step (E) ladder program
Step W00004
(E)
000005
SNXT
W10000
STEP

Application Examples

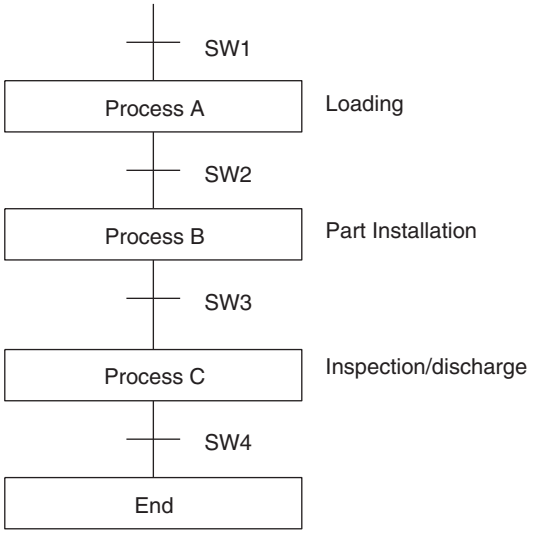
The following three examples demonstrate the three types of execution control possible with step programming. *Example 1* demonstrates sequential execution; *Example 2*, branching execution; and *Example 3*, parallel execution.

Example 1: Sequential Execution

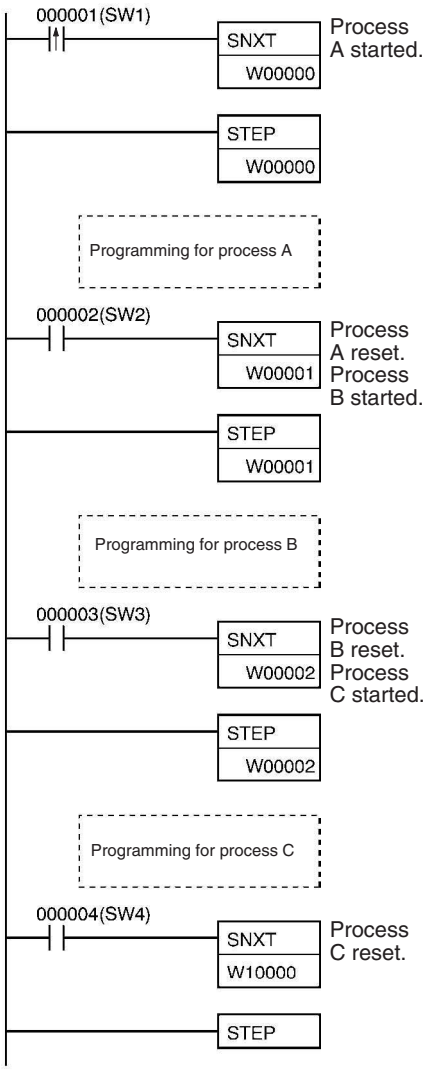
The following process requires that three processes, loading, part installation, and inspection/discharge, be executed in sequence with each process being reset before continuing on the next process. Various sensors (SW1, SW2, SW3, and SW4) are positioned to signal when processes are to start and end.



The following diagram demonstrates the flow of processing and the switches that are used for execution control.



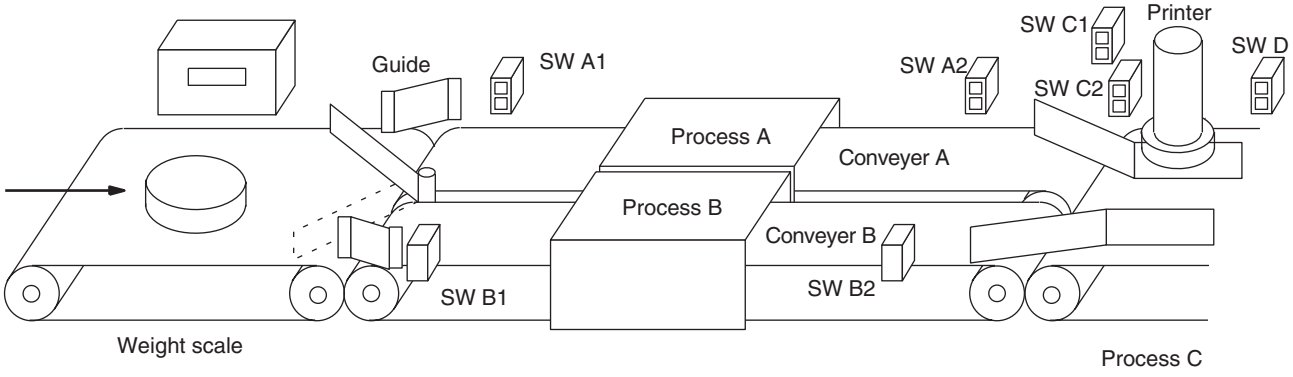
The program for this process, shown below, utilizes the most basic type of step programming: each step is completed by a unique SNXT(009) that starts the next step. Each step starts when the switch that indicates the previous step has been completed turns ON.



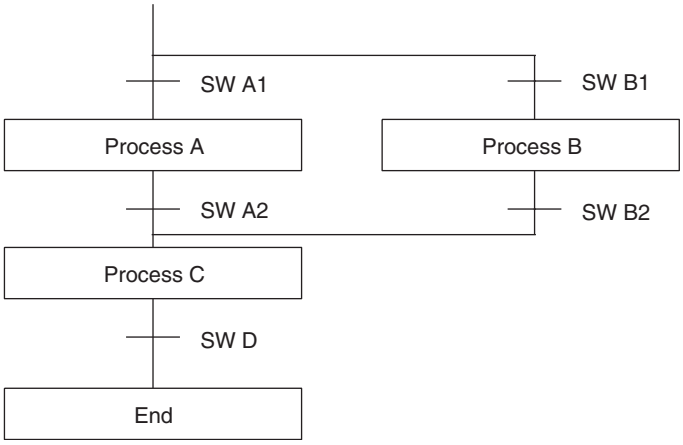
Address	Instruction	Operands
000000	@LD	000001
000001	SNXT(009)	W00000
000002	STEP(008)	W00000
	Process A	
000100	LD	000002
000101	SNXT(009)	W00001
000102	STEP(008)	W00001
	Process B	
000100	LD	000003
000101	SNXT(009)	W00002
000102	STEP(008)	W00002
	Process C	
000200	LD	000004
000201	SNXT(009)	W00003
000202	STEP(008)	W00003

Example 2: Branching Execution

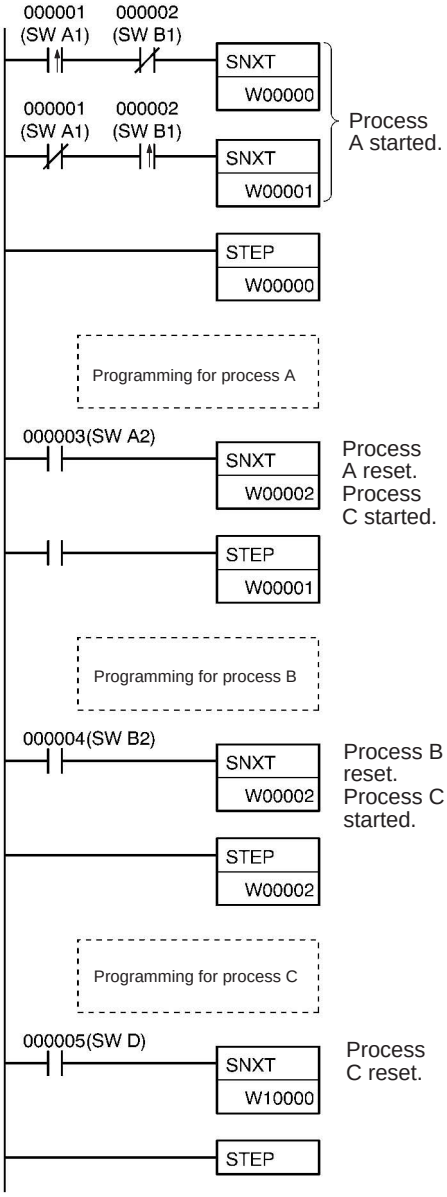
The following process requires that a product is processed in one of two ways, depending on its weight, before it is printed. The printing process is the same regardless of which of the first processes is used. Various sensors are positioned to signal when processes are to start and end.



The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, either process A or process B is used depending on the status of SW A1 and SW B1.



The program for this process, shown below, starts with two SNXT(009) instructions that start processes A and B. Because of the way CIO 000001 (SW A1) and CIO 000002 (SW B1) are programmed, only one of these will be executed with an ON execution condition to start either process A or process B. Both of the steps for these processes end with a SNXT(009) that starts the step (process C).

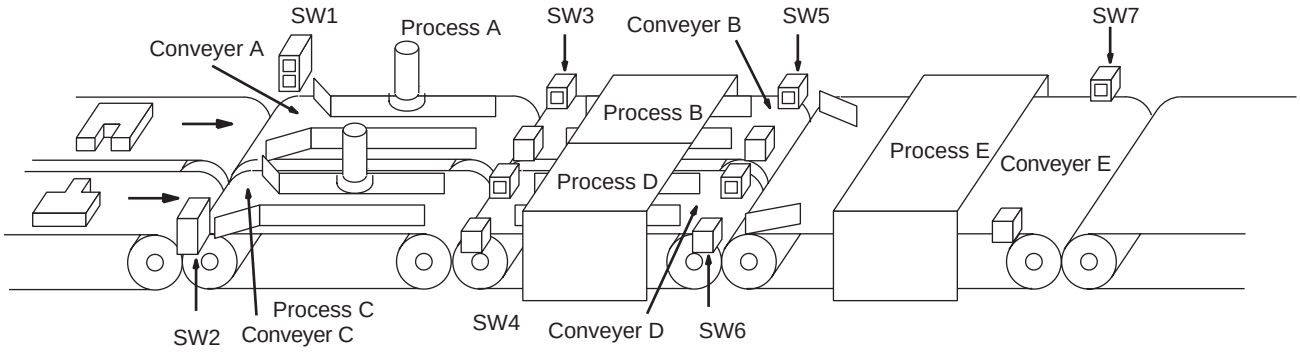


Address	Instruction	Operands
000000	@LD	000001
000001	AND NOT	000002
000002	SNXT(009)	010000
000003	LD NOT	000001
000004	@AND	000002
000005	SNXT(009)	010001
000006	STEP(008)	010000
Process A		
000100	LD	000003
000101	SNXT(009)	010002
000102	STEP(008)	010001
Process B		
000100	LD	000004
000101	SNXT(009)	010002
000102	STEP(008)	010002
Process C		
000200	LD	000005
000201	SNXT(009)	024614
000202	STEP(008)	---

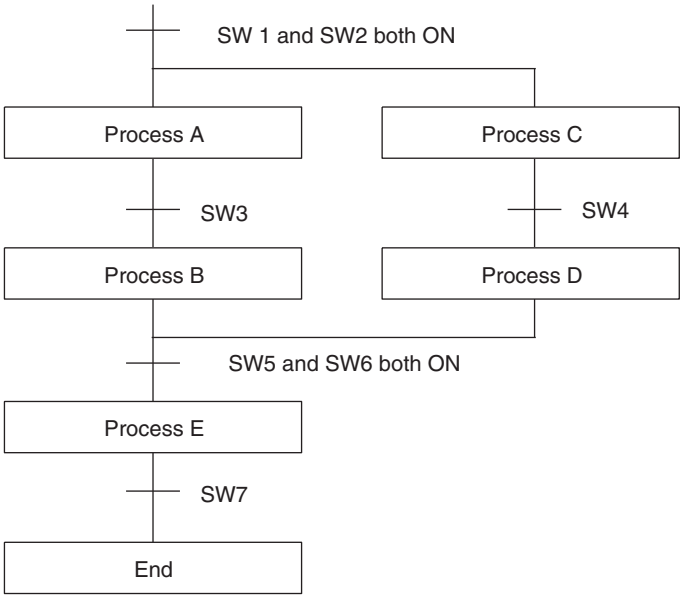
Note In the above programming, CIO 010002 is used in two SNXT(009) instructions. This will not produce a duplication error during the program check.

Example 3: Parallel Execution

The following process requires that two parts of a product pass simultaneously through two processes each before they are joined together in a fifth process. Various sensors are positioned to signal when processes are to start and end.

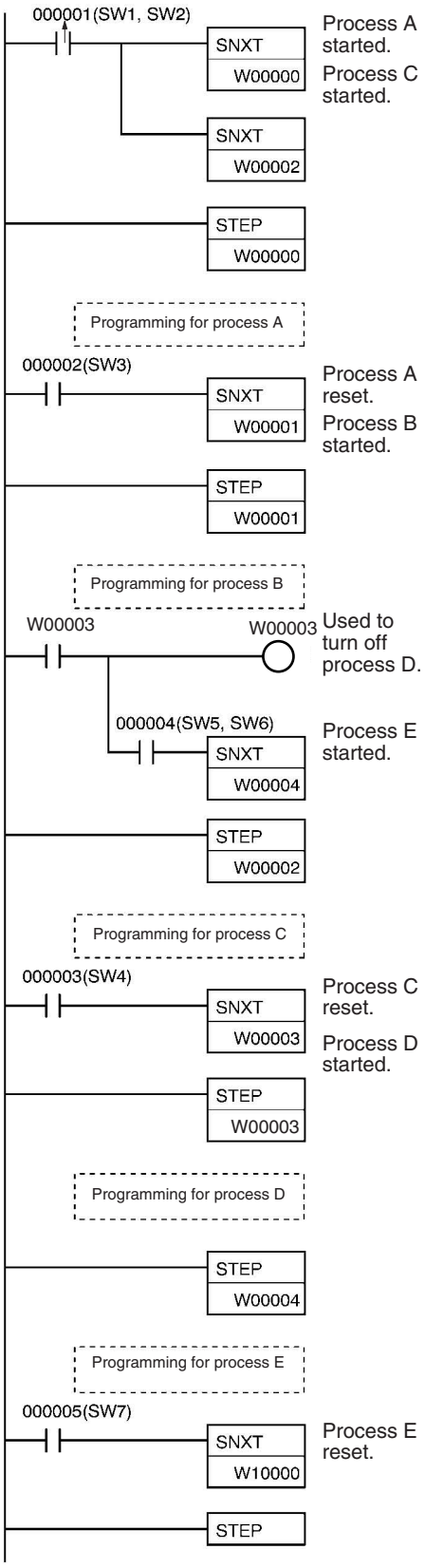


The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, process A and process C are started together. When process A finishes, process B starts; when process C finishes, process D starts. When both processes B and D have finished, process E starts.



The program for this operation, shown below, starts with two SNXT(009) instructions that start processes A and C. These instructions branch from the same instruction line and are always executed together, starting steps for both A and C. When the steps for both A and C have finished, the steps for process B and D begin immediately.

When both process B and process D have finished (i.e., when SW5 and SW6 turn ON), processes B and D are reset together by the SNXT(009) at the end of the programming for process B. Although there is no SNXT(009) at the end of process D, the control bit for it is turned OFF by executing SNXT(009) W00004. This is because the OUT for bit W00003 is in the step reset by SNXT(009) W00004, i.e., W00003 is turned OFF when SNXT(009) W00004 is executed. Process B is thus reset directly and process D is reset indirectly before executing the step for process E.



Address	Instruction	Operands
000000	@LD	000001
000001	SNXT(009)	W00000
000002	SNXT(009)	W00002
000003	STEP(008)	W00000
	Process A	
000100	LD	000002
000101	SNXT(009)	W00001
000102	STEP(008)	W00001
	Process B	
000100	LD	000003
000101	OUT	W00003
000101	AND	000004
000101	SNXT(009)	W00004
000102	STEP(008)	W00002
	Process C	
000200	LD	000003
000201	SNXT(009)	W00003
000202	STEP(008)	W00003
	Process D	
000300	STEP(008)	W00004
	Process E	
000400	LD	000005
000401	SNXT(009)	024613
000402	STEP(008)	---

3-23 Basic I/O Unit Instructions

This section describes instructions used with I/O Units.

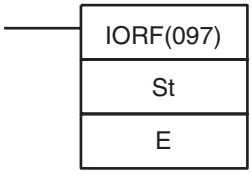
Instruction	Mnemonic	Function code	Page
I/O REFRESH	IORF	097	926
SPECIAL I/O UNIT I/O REFRESH	FIORF	225	929
CPU BUS UNIT I/O REFRESH	DLNK	226	932
7-SEGMENT DECODER	SDEC	078	937
INTELLIGENT I/O READ	IORD	222	962
INTELLIGENT I/O WRITE	IOWR	223	967
DIGITAL SWITCH INPUT	DSW	210	940
TEN KEY INPUT	TKY	211	945
HEXADECIMAL KEY INPUT	HKY	212	948
MATRIX INPUT	MTR	213	953
7-SEGMENT DISPLAY OUTPUT	7SEG	214	957

3-23-1 I/O REFRESH: IORF(097)

Purpose

Refreshes the specified I/O words.

Ladder Symbol



St: Starting word

E: End word

Variations

Variations		
	Executed Each Cycle for ON Condition	IORF(097)
	Executed Once for Upward Differentiation	@IORF(097)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

St: Starting Word
CIO 0000 to CIO 0999 (I/O Bit Area) or
CIO 2000 to CIO 2959 (Special I/O Unit Bit Area)

E: End Word
CIO 0000 to CIO 0999 (I/O Bit Area) or
CIO 2000 to CIO 2959 (Special I/O Unit Bit Area)

Note St and E must be in the same memory area.

Operand Specifications

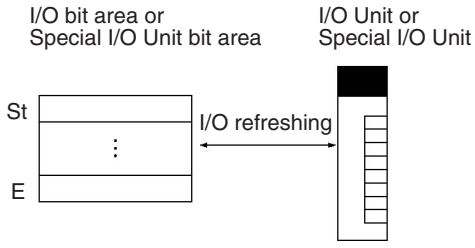
Area	St	E
CIO Area	CIO 0000 to CIO 0999 CIO 2000 to CIO 2959	
Auxiliary Area	---	
Holding Bit Area	---	
Special Bit Area	---	
Timer Area	---	

Area	St	E
Counter Area	---	
DM Area	---	
EM Area without bank	---	
EM Area with bank	---	
Indirect DM/EM addresses in binary	---	
Indirect DM/EM addresses in BCD	---	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to IR15 –2048 to +2047, IR0 to IR15 DR0 to DR15, IR0 to IR15, IR0 to IR15+(++) ,–(– –) IR0 to IR15	

Description

IORF(097) refreshes the I/O words between St and E, inclusively. IORF(097) is used to refresh words allocated to Basic I/O Units or Special I/O Units mounted on the CPU Rack or Expansion Racks. IORF(097) cannot be used to refresh words in both areas at the same time (i.e., with the same instruction). 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 is specified for words in the Special I/O Unit bit area, all 10 words allocated to the Unit will be refreshed as long as the first word of the 10 words allocated to the Unit is included in the specified range of words.



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.

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)

IORF(097) can be used in interrupt tasks, allowing high-speed response for the specific I/O words refreshed in the interrupt task. (See Precautions.)

Comparison with FIORF(225) and DLNK(226)

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

Applicable Units

The following Units can be refreshed with IORF(097). These Unit can be refreshed only when they are on the CPU Rack or an Expansion Rack. They cannot be refreshed if they are on Slave Racks.

CS-series Basic I/O Units, C200H Basic I/O Units (CS Series only), C200H Group-2 High-density I/O Units (CS Series only), CJ-series Basic I/O Units, and Special I/O Units (including High-density Units. All words allocated to the Units can be refreshed.)

Note The Units that can be refreshed with IORF(097) are not necessarily the same as the Units that can be refreshed with immediate refreshing specifications (!).

Flags

Name	Label	Operation
Error Flag	ER	ON if St is greater than E. ON if St and E are in different memory areas. With the CS1D CPU Units: ON if the active and standby CPU Units could not be synchronized. OFF in all other cases.

Precautions

An error will occur if words in both the I/O Bit Area (CIO 0000 to CIO 0999) and the Special I/O Unit Bit Area (CIO 2000 to CIO 2959) are specified for the same instruction.

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.

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.

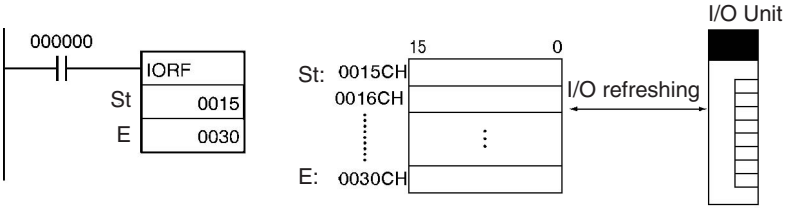
- Words allocated to the same Special I/O Unit were already refreshed by IORF(097) or FIORF(225).
- Words allocated to the same Special I/O Unit were read or written by IORD(222) or IOWR(223).

When cyclic refreshing of a Special I/O Unit is disabled, execute IORF(097) or FIORF(225) (CJ1-H-R CPU Units only) 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.

Examples

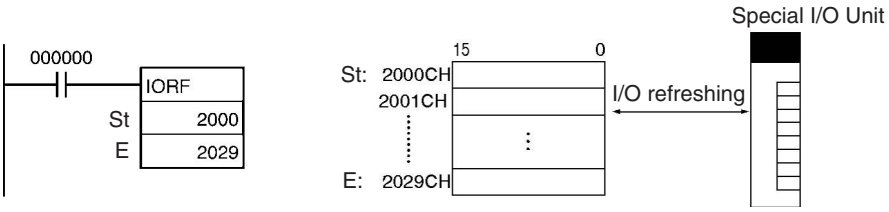
Refreshing Words in the I/O Bit Area

The following example shows how to refresh 16 words from CIO 0015 to CIO 0030 when CIO 000000 turns ON.



Refreshing Words in the Special I/O Unit Bit Area

The following example shows how to refresh 30 words from CIO 2000 to CIO 2029 when CIO 000000 turns ON.



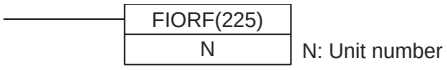
3-23-2 SPECIAL I/O UNIT I/O REFRESH: FIORF(225)

Purpose

Performs I/O refreshing immediately for the specified Special I/O Unit's allocated CIO Area and DM Area words.t with the specified unit number.

This instruction is supported by the CJ1-H-R CPU Units only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	FIORF(225)
	Executed Once for Upward Differentiation	@FIORF(225)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operands

N: Unit number

Specifies the Special I/O Unit's unit number (0000 to 005F hex or 0 to 95 decimal).

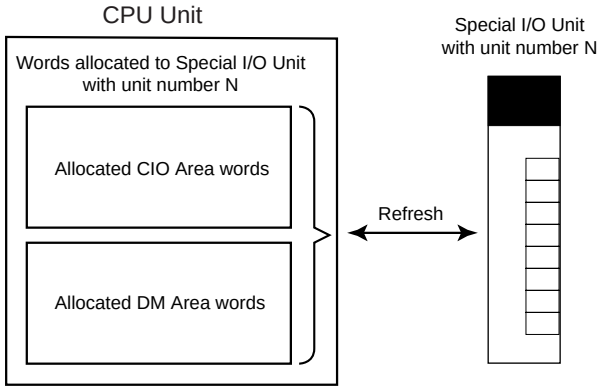
Note If the Special I/O Unit uses more than one unit number, specify the lowest unit number.

Operand Specifications

Area	N
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	#0000 to #005F (binary) or 0 to 95 (decimal)
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

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.



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 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

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.

Purpose

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.

Units Refreshed by FIORF(225)

Unit type (See note.)	Refreshable by FIORF(255)
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

Note This table applies to Units mounted in a CPU Rack or an Expansion Rack. It does not apply to Units mounted in a SYSMAC Bus Slave Rack.

Flags

Name	Label	Operation
Error Flag	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	=	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.

Precautions

I/O refreshing by FIORF(225) will be stopped if an I/O Bus Error occurs while during I/O refreshing.

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.

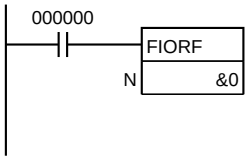
When cyclic refreshing of the specified Special I/O Unit is enabled in the PLC Setup (the corresponding Special I/O Unit Cyclic Refreshing Disable Bit is OFF), a non-fatal Duplicate Refresh Error will occur and the Interrupt Task Error Flag (A40213) will go ON in the following cases.

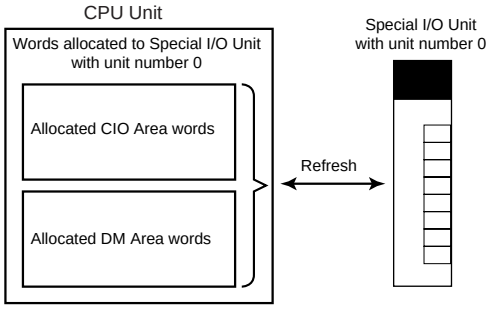
- Words allocated to the same Special I/O Unit were already refreshed by IORF(097) or FIORF(225).
- Words allocated to the same Special I/O Unit were read or written by IORD(222) or IOWR(223).

When cyclic refreshing of a Special I/O Unit is disabled, execute IORF(097) or FIORF(225) (CJ1-H-R CPU Units only) 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.

Operation Examples

When CIO 000000 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.





3-23-3 CPU BUS UNIT I/O REFRESH: DLNK(226)

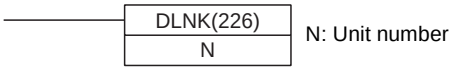
Purpose

Performs I/O refreshing immediately for the CPU Bus Unit with the specified unit number. The following data is refreshed.

- The words allocated to the CPU Bus Unit in the PLC's CPU Bus Unit Areas (25 words in the CIO Area and 100 words in the DM Area)
- Specific data refreshing for Units such as Units that support data links

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	DLNK(226)
	Executed Once for Upward Differentiation	@DLNK(226)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

N: Unit number
Specifies the CPU Bus Unit's unit number (0000 to 000F hex or 0 to 15 decimal).

Operand Specifications

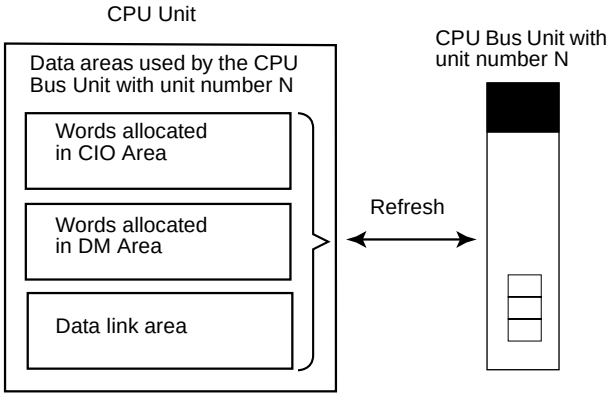
Area	N
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	#0000 to #000F (binary) or 0 to 15 (decimal)
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

DLNK(226) performs immediate I/O refreshing for the CPU Bus Unit with the specified unit number. The data listed below is refreshed. Refer to the *Precautions* below for details on the execution conditions to use for immediate refreshing.

1. The words allocated to the CPU Bus Unit in the PLC's CPU Bus Unit Areas (25 words in the CIO Area and 100 words in the DM Area)
2. Data specific the CPU Bus Unit such as data link data or DeviceNet Remote I/O Communications data (refreshed together with the data in the CPU Bush Unit Areas)

CPU Bus Unit	Data refreshing specific to the Unit
Controller Link Unit or SYSMAC Link Unit	Data link refreshing
DeviceNet Unit (Does not include C200H DeviceNet Master Units.)	Remote I/O communications refreshing



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):

1,2,3...

1. When exchanging data through a data link or DeviceNet remote I/O communications, the data exchange is not performed with the other Units at the same time that DLNK(226) is executed. The data exchange will be performed when the network communications cycle reaches the Unit in question and data is exchanged with that Unit. Consequently, the actual data exchange may be delayed by as much as the communications cycle time of the network.
2. DLNK(226) cannot perform I/O refreshing with a CPU Bus Unit if that Unit is currently exchanging data. If DLNK(226) is executed too frequently, I/O refreshing will not be performed. We recommend allowing a delay between executions of DLNK(226) that is longer than the communications cycle time.

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified unit number is not between 0000 and 000F hex (between 0 and 15 decimal). ON if the PLC does not have a CPU Bus Unit with the specified unit number. With the CS1D CPU Units: ON if the active and standby CPU Units could not be synchronized. OFF in all other cases.
Equals Flag	=	OFF if the I/O refreshing could not be performed because the CPU Bus Unit was refreshing data. OFF if there was 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.

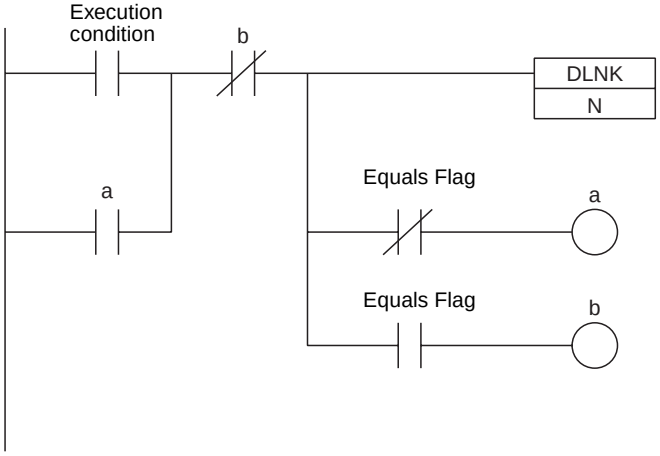
Precautions

I/O refreshing will not be performed if a CPU Bus Unit Error (A40207) or CPU Bus Unit Setup Error (A40203) has occurred in the specified CPU Bus Unit.

I/O refreshing will be stopped if an I/O Bus Error occurs while I/O refreshing is being performed by DLNK(226).

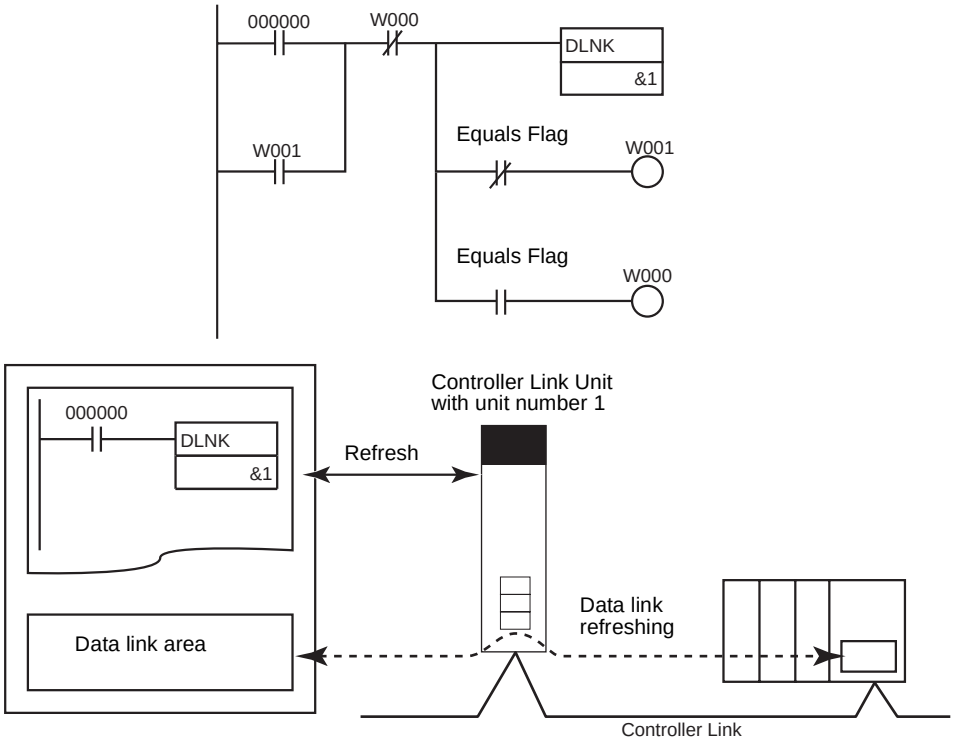
DLNK(226) refreshes data between the CPU Unit and specified CPU Bus Unit. Some time is required for the data exchange with the CPU Bus Unit (for example, a data link with a Controller Link Unit).

If the specified CPU Bus Unit is exchanging data, DLNK(226) will not be executed and the Equals Flag will be turned OFF. We recommend programming the execution conditions shown below so that the execution of DLNK(226) will be retried automatically.



Example

When CIO 000000 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 W001 to be turned ON so that the instruction execution will be retried in the next cycle. When the I/O refreshing is completed normally, the Equals Flag will be turned ON and the instruction will not be retried in the next cycle.

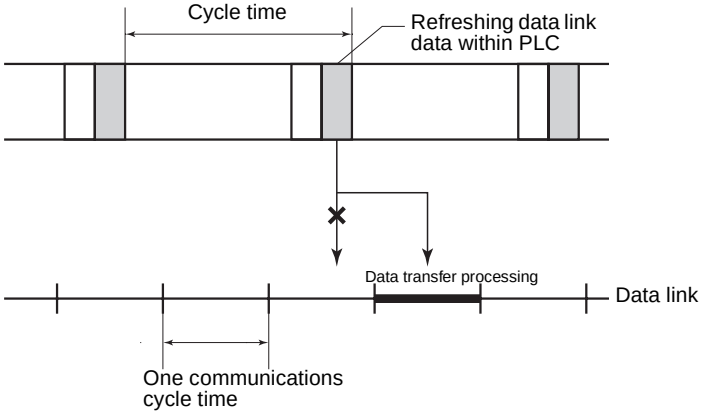


The actual timing for data link area refreshing in this example is as follows:

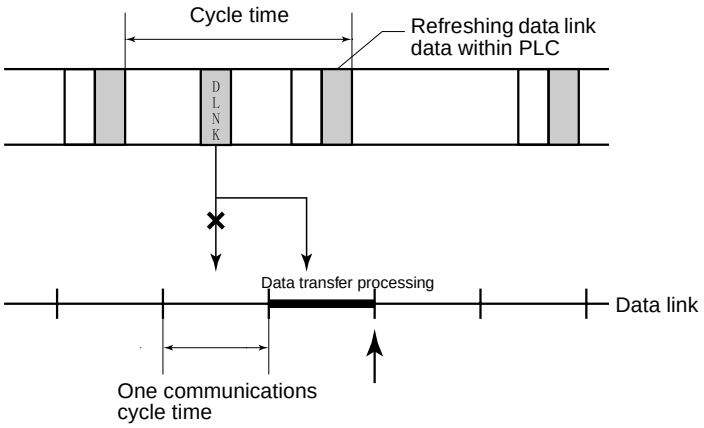
- When transmitting: Data is transmitted over the network the next time that the token right is acquired. (The transmitted data is delayed up to 1 communications cycle time max.)
- When receiving: The data that is input was received from the network the last time that the token right was acquired. (The data received is delayed up to 1 communications cycle time max.)

Examples of Data Transfer Processing:

- Transferring Data from the Previous I/O Refreshing



• Transferring Data with Execution of DLNK(226)

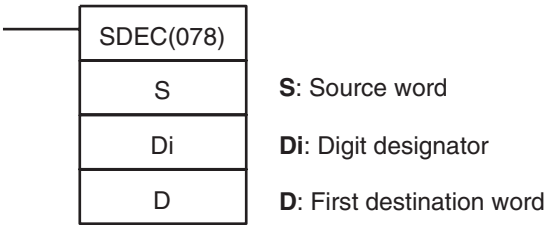


3-23-4 7-SEGMENT DECODER: SDEC(078)

Purpose

Converts the hexadecimal contents of the designated digit(s) into 8-bit, 7-segment display code and places it into the upper or lower 8-bits of the specified destination words.

Ladder Symbol



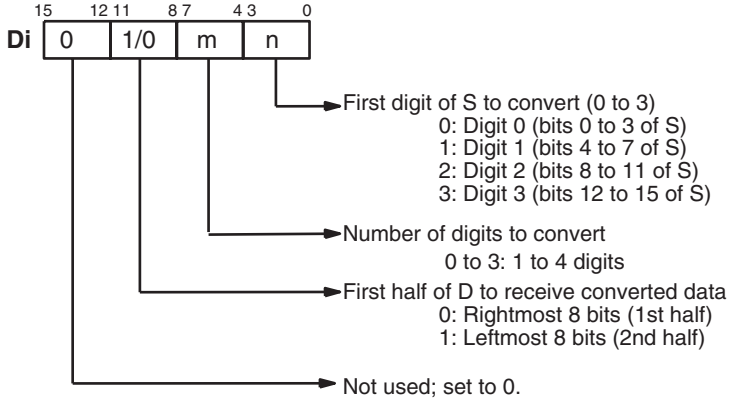
Variations

Variations		
	Executed Each Cycle for ON Condition	SDEC(078)
	Executed Once for Upward Differentiation	@SDEC(078)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands: Digit Designator

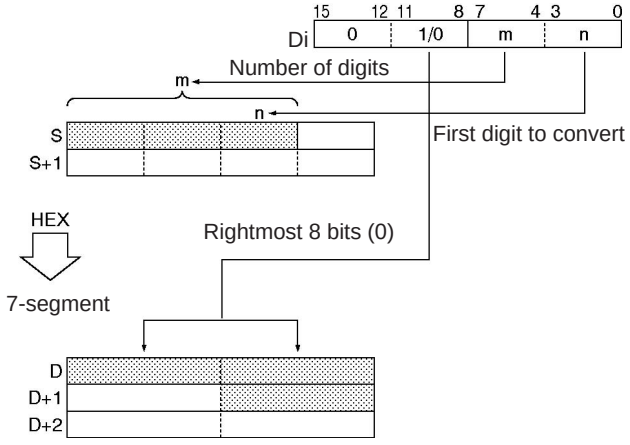


Operand Specifications

Area	S	Di	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@D00000 to @D32767 @E00000 to @E32767 @En_00000 to @En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	---
Data Registers	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SDEC(078) regards the data specified by S as 4-digit hexadecimal data, converts the digits specified in S by Di (first digit and number of digits) to 7-segment data and outputs the results to D in the bits specified in Di.



Flags

Name	Label	Operation
Error Flag	ER	ON if settings in Di are not within the specified ranges. OFF in all other cases.

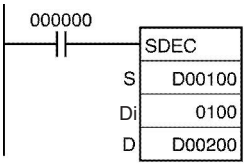
Precautions

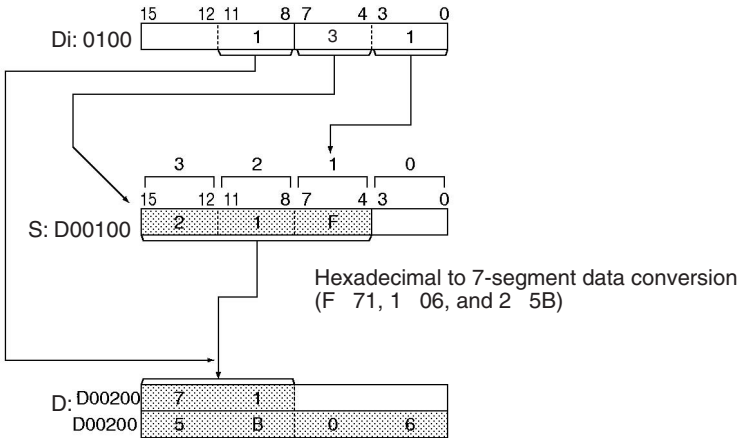
If more than one digit is specified for conversion in Di, digits are converted in order toward the most-significant digit. Digit 0 is the next digit after digit 3.

Results are stored in D in order from the specified portion toward higher-address words. If just one of the bytes in a destination word receives converted data, the other byte is left unchanged.

Examples

When CIO 000000 turns ON in the following example, the contents of the 3 digits beginning with digit 1 in D00100 will be converted from hexadecimal data to 7-segment data, and the results will be output to the upper byte of D00200 and both bytes of D00201. The specifications of the bytes to be converted and the location of the output bytes are made in CIO 0100.

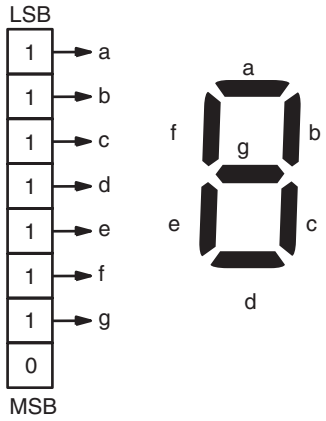




7-segment Data

The following table shows the data conversions from a hexadecimal digit (4 bits) to 7-segment code (8 bits).

Original data					Converted code (segments)									Display Original data
Digit	Bits				–	g	f	e	d	c	b	a	Hex	
0	0	0	0	0	0	0	1	1	1	1	1	1	3F	0
1	0	0	0	1	0	0	0	0	0	1	1	0	06	1
2	0	0	1	0	0	1	0	1	1	0	1	1	5B	2
3	0	0	1	1	0	1	0	0	1	1	1	1	4F	3
4	0	1	0	0	0	1	1	0	0	1	1	0	66	4
5	0	1	0	1	0	1	1	0	1	1	0	1	6D	5
6	0	1	1	0	0	1	1	1	1	1	0	1	7D	6
7	0	1	1	1	0	0	1	0	0	1	1	1	27	7
8	1	0	0	0	0	1	1	1	1	1	1	1	7F	8
9	1	0	0	1	0	1	1	0	1	1	1	1	6F	9
A	1	0	1	0	0	1	1	1	0	1	1	1	77	A
B	1	0	1	1	0	1	1	1	1	1	0	0	7C	b
C	1	1	0	0	0	0	1	1	1	0	0	1	39	C
D	1	1	0	1	0	1	0	1	1	1	1	0	5E	d
E	1	1	1	0	0	1	1	1	1	0	0	1	79	E
F	1	1	1	1	0	1	1	1	0	0	0	1	71	F



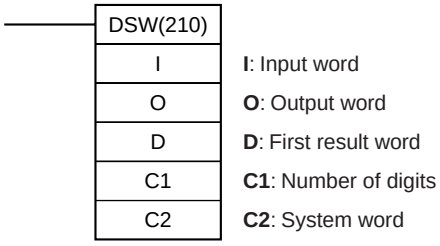
3-23-5 DIGITAL SWITCH INPUT – DSW(210)

Purpose

Reads the value set on a external digital switch (or thumbwheel switch) connected to an I/O Unit and stores the 4-digit or 8-digit value in the specified words.

This instruction is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



I: Input word
O: Output word
D: First result word
C1: Number of digits
C2: System word

Variations

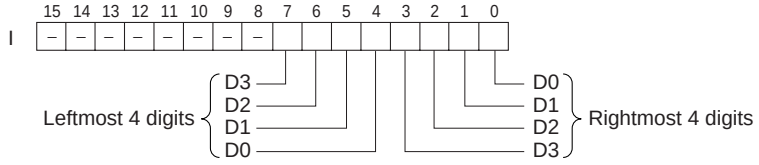
Variations		DSW(210)
	Executed Each Cycle for ON Condition	
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

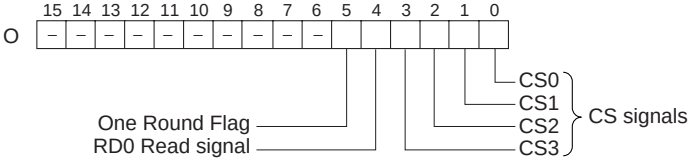
Operands

I: **Input Word (Data Line D0 to D3 Inputs)**
Specify the input word allocated to the Input Unit and connect the digital switch's D0 to D3 data lines to the Input Unit as shown in the following diagram.

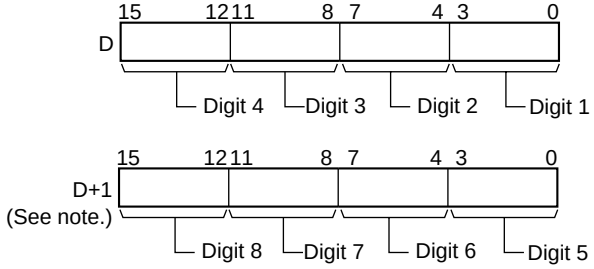


O: Output Word (CS/RD Control Signal Outputs)

Specify the output word allocated to the Output Unit and connect the digital switch's control signals (CS and RD signals) to the Output Unit as shown in the following diagram.

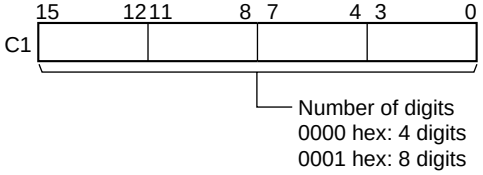


D: First Result Word
Specifies the leading word address where the external digital switch's set values will be stored.

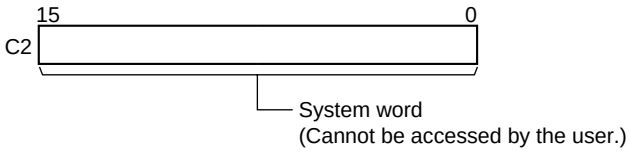


Note: Only when C1 = 0001 hex to read 8 digits.

C1: Number of Digits
Specifies the number of digits that will be read from the external digital switch. Set C1 to 0000 hex to read 4 digits or 0001 hex to read 8 digits.



C2: System Word
Specifies a work word used by the instruction. This word cannot be used in any other application.



Operand Specifications

Area	I	O	D	C1	C2
CIO Area	CIO 0000 to CIO 6143			---	CIO 0000 to CIO 6143
Work Area	W000 to W511			---	W000 to W511
Holding Bit Area	H000 to H511			---	H000 to H511
Auxiliary Bit Area	A000 to A959	A448 to A953		---	A448 to A959
Timer Area	T0000 to T4095			---	T0000 to T4095
Counter Area	C0000 to C4095			---	C0000 to C4095
DM Area	D00000 to D32767			---	D00000 to D32767
EM Area without bank	E00000 to E32767			---	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)			---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)			---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)			---	---
Constants	---			0000 or 0001 hex	---
Data Registers	DR0 to DR15				DR0 to DR15
Index Registers	---				
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15				,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

DSW(210) outputs control signals to bits 00 to 04 of O, reads the specified number of digits (either 4-digit or 8-digit, specified in C1) of digital switch data line data from I, and stores the result in D and D+1. (If 4 digits are read, the result is stored in D. If 8 digits are read, the result is stored in D and D+1.)

DSW(210) reads the 4-digit or 8-digit switch data once every 16 cycles, and then starts over and continues reading the data. The One Round Flag (bit 05 of O) is turned ON once every 16 CPU Unit cycles.

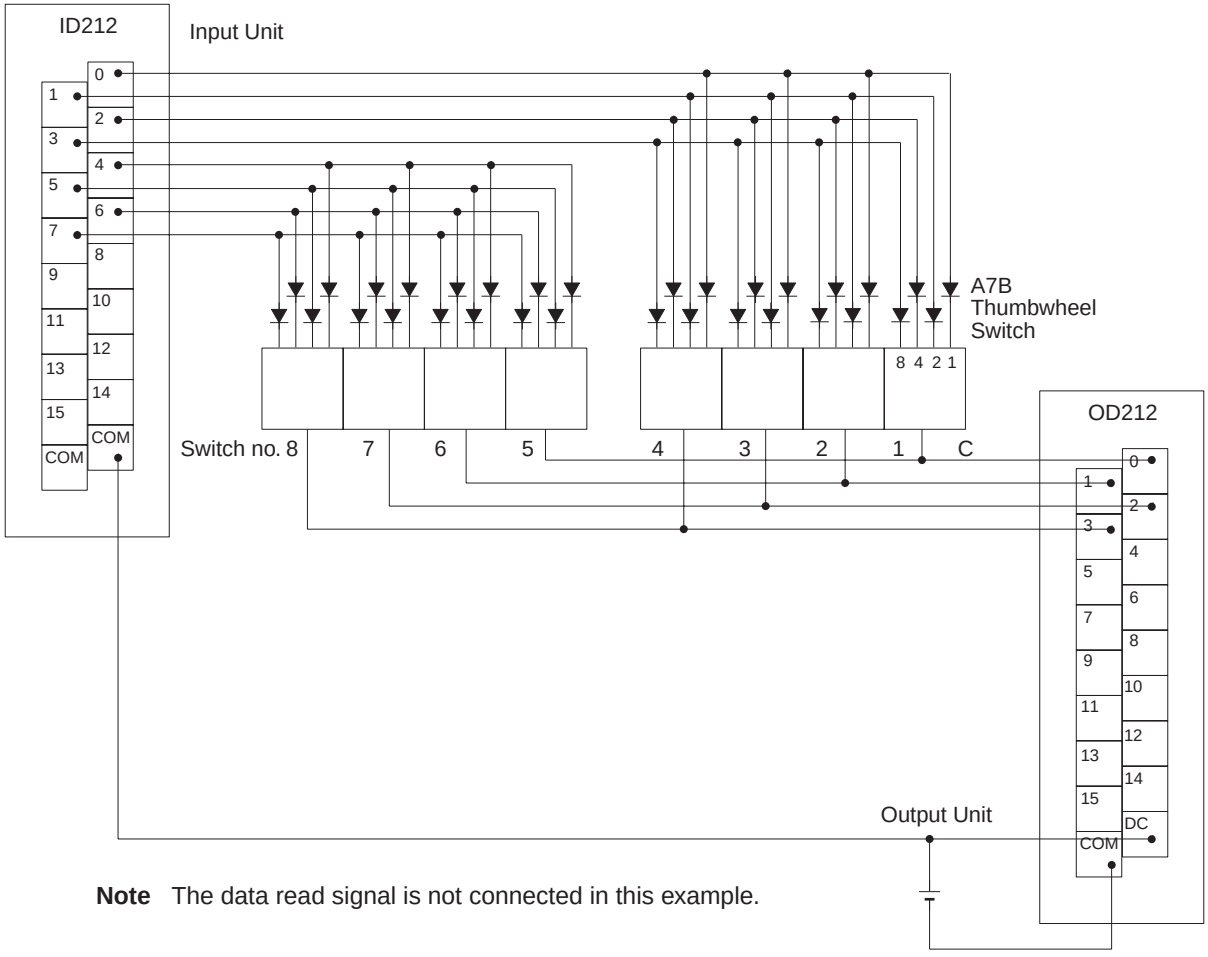
DSW(210) reads the 4-digit or 8-digit data once in 16 cycles, and then starts over and reads the data again in the next 16 cycles.

When executed, DSW(210) begins reading the switch data from the first of the sixteen cycles, regardless of the point at which the last instruction was stopped.

There is no restriction on the number of times that DSW(210) can appear in the program (unlike the C200HX/HG/HE and CQM1H Series).

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.

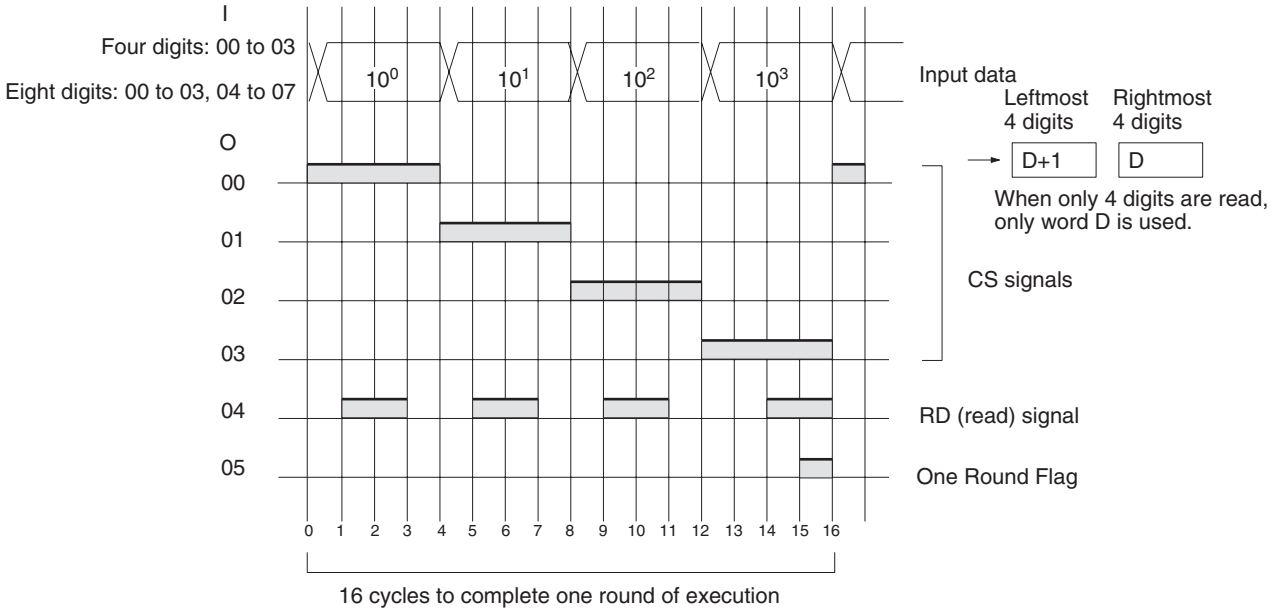


Note The data read signal is not connected in this example.

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



Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

Do not read or write the system word (C2) from any other instruction. DSW(210) will not operate correctly if the system word is accessed by another instruction. The system word is not initialized by DSW(210) in the first cycle when program execution starts. If DSW(210) is being used from the first cycle, clear the system word from the program.

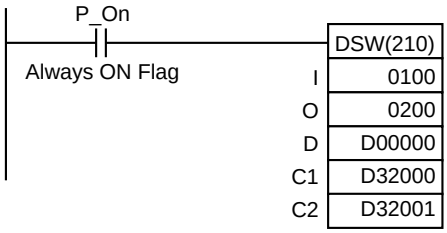
DSW(210) will not operate correctly if I/O refreshing is not performed with the Input Unit and Output Unit connected to the digital switch or thumbwheel switch after DSW(210) is executed. Consequently, set the input time constant for the Input Units used for the data line input word to a value that is shorter than the cycle time, or 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)

Example

In this example, DSW(210) is used to read an 8-digit number from a digital switch and outputs the resulting value constantly to D00000 and D00001. The digital switch is connected through CIO 0100 (allocated to a CS1W-ID211 16-point DC Input Unit) and CIO 0200 (allocated to a CS1W-OD211 16-point Transistor Output Unit).

Since 8 digits of data are being read, C1 (D32000 in this case) is set to 0001 hex. D32001 is used as the system word.



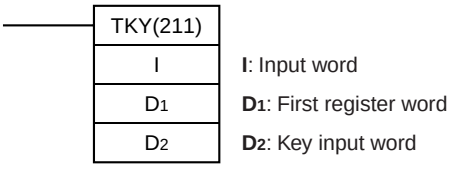
3-23-6 TEN KEY INPUT – TKY(211)

Purpose

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.

This instruction is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	TKY(211)
	Executed Once for Upward Differentiation	@TKY(211)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

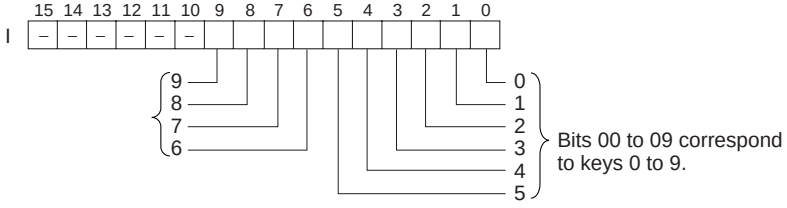
Applicable Program Areas

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

Operands

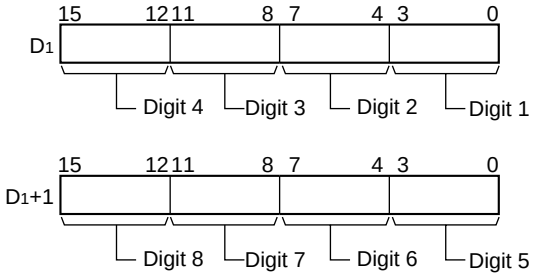
I: **Input Word (Data Line Inputs)**

Specify the input word allocated to the Input Unit and connect the ten-key keypad's 0 to 9 data lines to the Input Unit as shown in the following diagram.



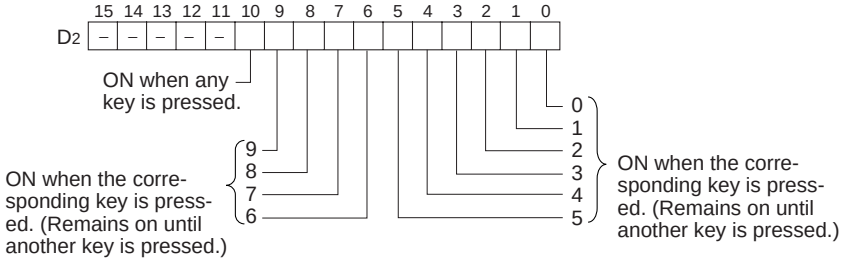
D_1: First Register Word

Specifies the leading word address where the ten-key keypad's numeric input (up to 8 digits) will be stored.



D_2: Key Input Word

Bits 00 to 10 of D_2 indicate key inputs. When one of the keys on the keypad (0 to 9) has been pressed, the corresponding bit of D_2 (0 to 9) is turned ON. Bit 10 of D_2 will be ON while any key is being pressed.



Note TKY(211) does not require a system word, unlike other I/O instructions such as HKY(212).

Operand Specifications

Area	I	D_1	D_2
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W510	W000 to W511
Holding Bit Area	H000 to H511	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to A959	A448 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	DR0 to DR15	---	DR0 to DR15

Area	I	D_1	D_2
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

TKY(211) reads numeric data from input word I, which is allocated to a ten-key keypad connected to an Input Unit, and stores up to 8 digits of BCD data in register words D_1 and D_1+1. In addition, each time that a key is pressed, the corresponding bit in D_2 (0 to 9) will be turned ON and remains ON until another key is pressed. Bit 10 of D_2 will be ON while any key is being pressed and OFF when no key is being pressed.

The two-word register in D_1 and D_1+1 operates as an 8-digit shift register. When a key is pressed on the ten-key keypad, the corresponding BCD digit is shifted into the least significant digit of D_1. The other digits of D_1, D_1+1 are shifted left and the most significant digit of D_1+1 is lost.

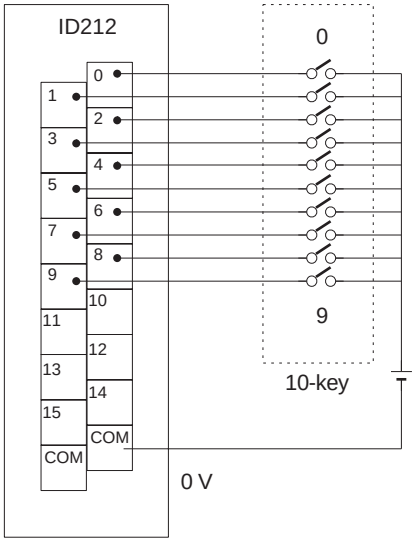
When executed, TKY(211) begins reading the key input data from the first cycle, regardless of the point at which the last instruction was stopped.

When one of the keypad keys is being pressed, input from the other keys is disabled.

There is no restriction on the number of times that TKY(211) can appear in the program (unlike the C200HX/HG/HE and CQM1H Series).

External Connections

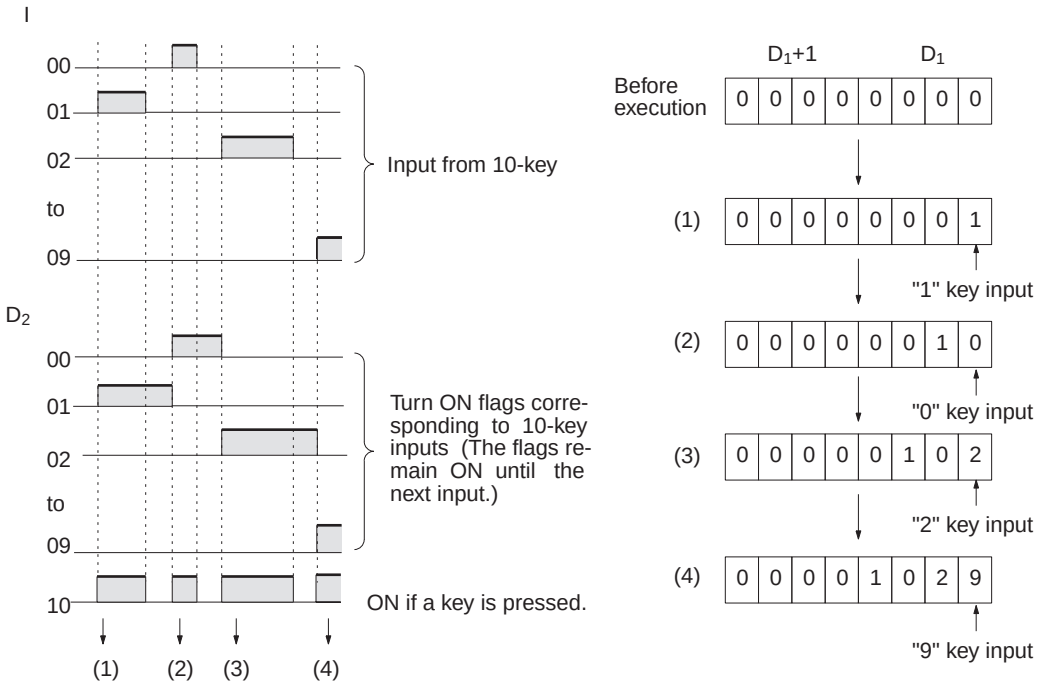
Connect the ten-key keypad so that the switches for keys 0 through 9 are input to contacts 0 through 9 of the Input Unit, as shown in the following diagram.



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



Flags

Name	Label	Operation
Error Flag	ER	OFF

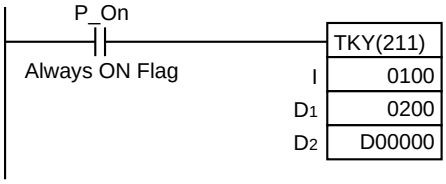
Precautions

TKY(211) will not operate correctly if I/O refreshing is not performed Input Unit connected to the ten-key keypad after TKY(211) is executed. Consequently, set the input time constant for the Input Units used for the data line input word to a value that is shorter than the cycle time, or 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)

Example

In this example, TKY(211) reads key inputs from a ten-key keypad and stores the inputs in CIO 200 and CIO 201. The ten-key keypad is connected to CIO 0100 (allocated to a CS1W-ID211 16-point DC Input Unit).



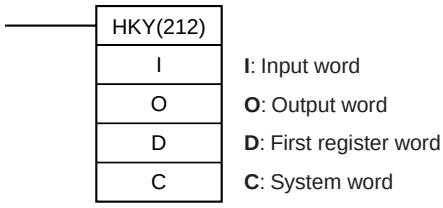
3-23-7 HEXADECIMAL KEY INPUT – HKY(212)

Purpose

Reads numeric data from a hexadecimal keypad connected to an Input Unit and Output Unit and stores up to 8 digits of hexadecimal data in the specified words.

This instruction is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	HKY(212)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

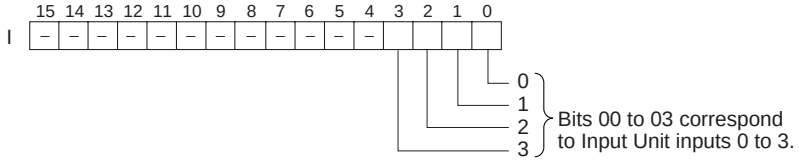
Applicable Program Areas

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

Operands

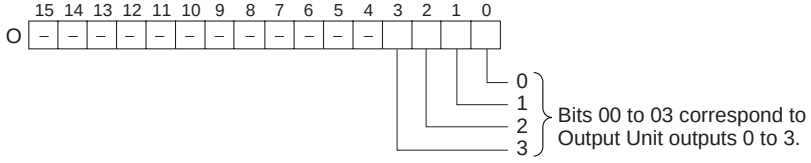
I: **Input Word (Data Line D0 to D3 Inputs)**

Specify the input word allocated to the Input Unit and connect the hexadecimal keypad's D0 to D3 data lines to the Input Unit as shown in the following diagram.



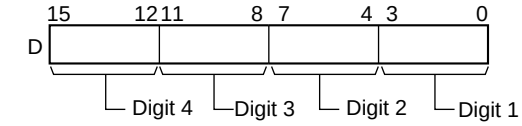
O: Output Word (Selection Signal Outputs)

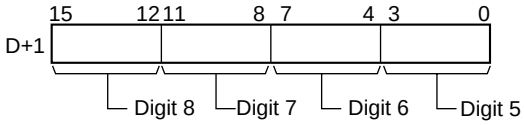
Specify the output word allocated to the Output Unit and connect the hexadecimal keypad's selection signals to the Output Unit as shown in the following diagram.

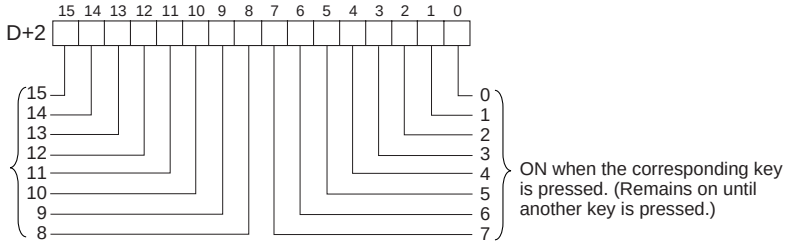


D: First Register Word

Specifies the leading word address where the hexadecimal keypad's numeric input (up to 8 digits) will be stored. (In addition, each time that a key is pressed, the corresponding bit in D+2 (0 to F) will be turned ON and remains ON until another key is pressed.)

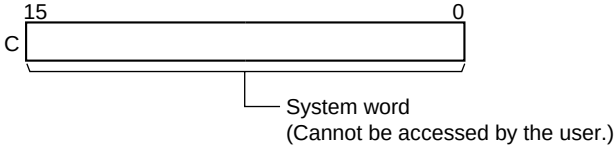






C: System Word

Specifies a work word used by the instruction. This word cannot be used in any other application.



Operand Specifications

Area	I	O	D	C
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6141	CIO 0000 to CIO 6143
Work Area	W000 to W511		W000 to W509	W000 to W511
Holding Bit Area	H000 to H511		H000 to H509	H000 to H511
Auxiliary Bit Area	A000 to A957	A448 to A959	A448 to A957	A448 to A959
Timer Area	T0000 to T4095		T0000 to T4093	T0000 to T4095
Counter Area	C0000 to C4095		C0000 to C4093	C0000 to C4095
DM Area	D00000 to D32767		D00000 to D32765	D00000 to D32767
EM Area without bank	E00000 to E32767		E00000 to E32765	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32765 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)			

Area	I	O	D	C
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)			
Constants	---			
Data Registers	DR0 to DR15		---	DR0 to DR15
Index Registers	---			
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

HKY(212) outputs the selection signals to bits 00 to 03 of O, reads the data in order from bits 00 to 03 of I, and stores up to 8 digits of hexadecimal data in register words D and D+1.

HKY(212) inputs each digit in 3 to 12 cycles, and then starts over and continues inputting. In addition, each time that a key is pressed, the corresponding bit in D+2 (0 to F) will be turned ON and remains ON until another key is pressed.

HKY(212) determines which key is pressed by identifying which input is ON when a given selection signal is ON, so it can take anywhere from 3 to 12 cycles for one hexadecimal digit to be read. After the key input is read, HKY(212) starts over and reads another digit in the next 3 to 12 cycles.

When executed, HKY(212) begins reading the key input data from the first selection signal, regardless of the point at which the last instruction was stopped.

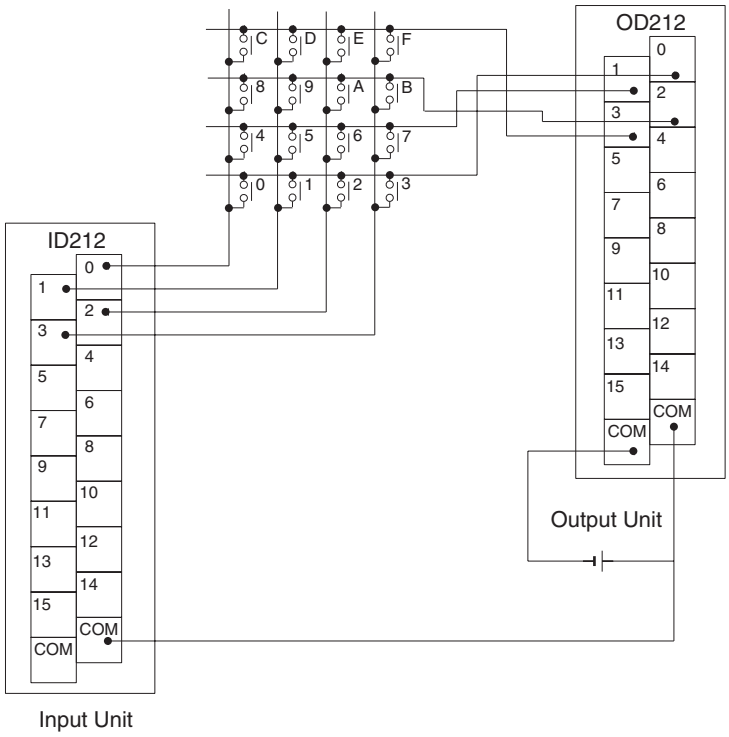
The two-word register in D_1 and D_1+1 operates as an 8-digit shift register. When a key is pressed on the ten-key keypad, the corresponding hexadecimal digit is shifted into the least significant digit of D_1. The other digits of D_1, D_1+1 are shifted left and the most significant digit of D_1+1 is lost.

When one of the keypad keys is being pressed, input from the other keys is disabled.

There is no restriction on the number of times that HKY(212) can appear in the program (unlike the CQM1H Series).

External Connections

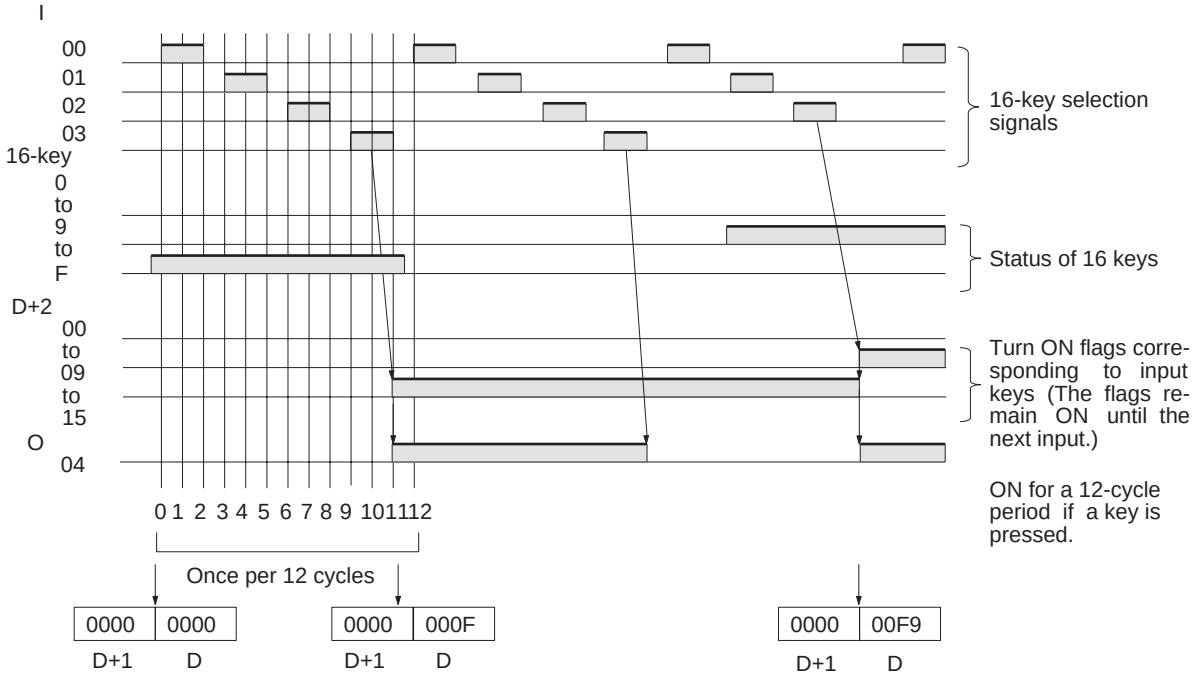
Connect the hexadecimal keypad to Input Unit contacts 0 to 3 and Output Unit contacts 0 to 3, as shown in the following diagram.



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



Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

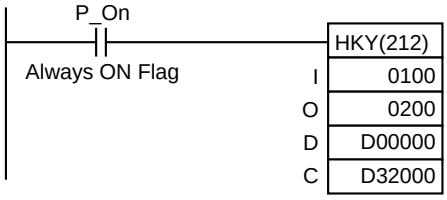
Do not read or write the system word (C) from any other instruction. HKY(212) will not operate correctly if the system word is accessed by another instruction. The system word is not initialized by HKY(212) in the first cycle when program execution starts. If HKY(212) is being used from the first cycle, clear the system word from the program.

HKY(212) will not operate correctly if I/O refreshing is not performed with the Input Unit and Output Unit connected to the hexadecimal keypad after HKY(212) is executed. Consequently, set the input time constant for the Input Units used for the data line input word to a value that is shorter than the cycle time, or 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)

Example

In this example, HKY(212) reads up to 8 digits of hexadecimal data from a hexadecimal keypad and stores the data in D00000 and D00001. The hexadecimal keypad is connected through CIO 0100 (allocated to a CS1W-ID211 16-point DC Input Unit) and CIO 0200 (allocated to a CS1W-OD211 16-point Transistor Output Unit). D32000 is used as the system word.



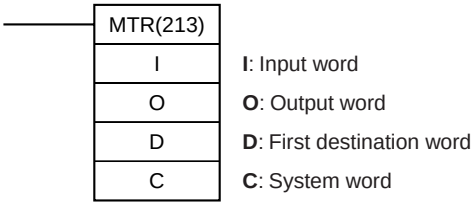
3-23-8 MATRIX INPUT: MTR(213)

Purpose

Inputs up to 64 signals from an 8 × 8 matrix connected to an Input Unit and an Output Unit (using 8 input points and 8 output points) and stores that 64-bit data in the 4 destination words.

This instruction is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	MTR(213)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

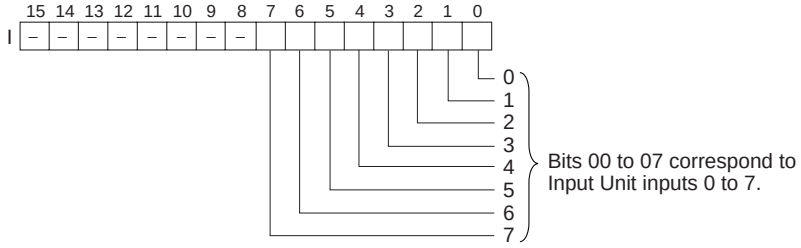
Applicable Program Areas

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

Operands

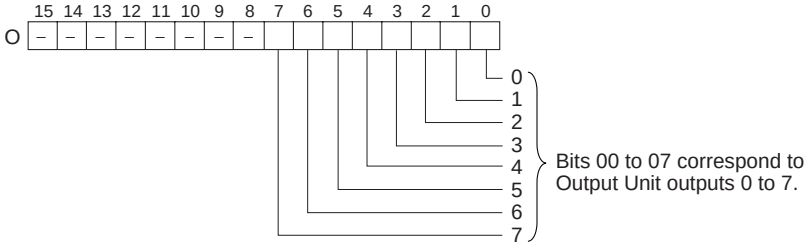
I: **Input Word**

Specify the input word allocated to the Input Unit and connect the 8 input signal lines to the Input Unit as shown in the following diagram.



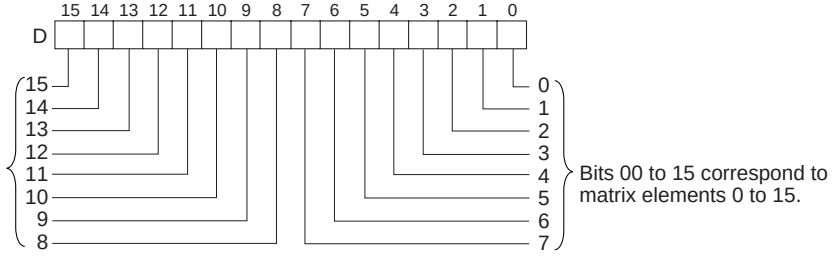
O: Output Word (Selection Signal Outputs)

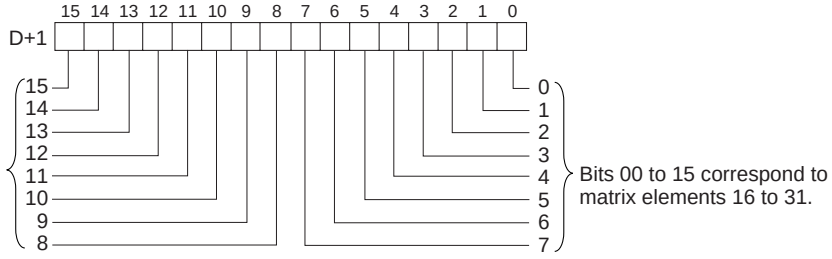
Specify the output word allocated to the Output Unit and connect the 8 selection signals to the Output Unit as shown in the following diagram.

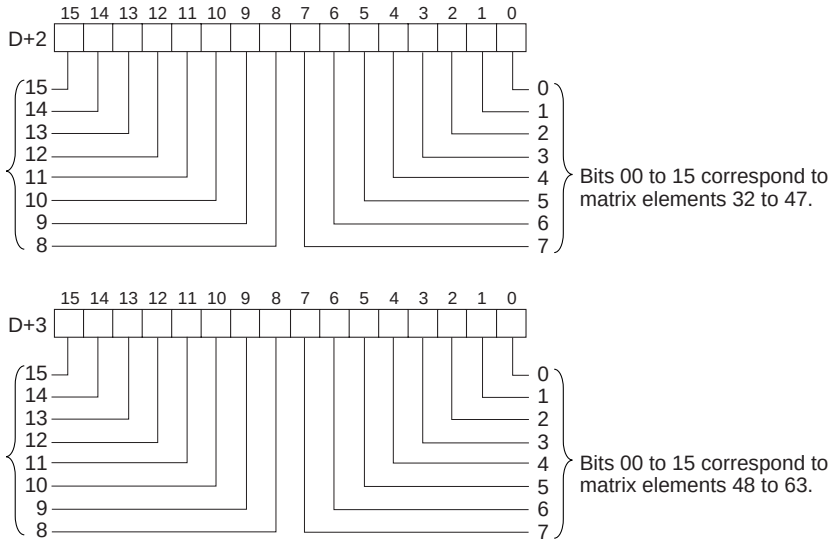


D: First Register Word

Specifies the leading word address of the 4 words that contain the data from the 8×8 matrix.

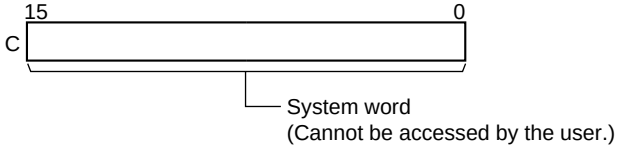






C: System Word

Specifies a work word used by the instruction. This word cannot be used in any other application.



Operand Specifications

Area	I	O	D	C
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 614	CIO 0000 to CIO 6143
Work Area	W000 to W511		W000 to W508	W000 to W511
Holding Bit Area	H000 to H511		H000 to H508	H000 to H511
Auxiliary Bit Area	A000 to A959	A448 to A959	A448 to A956	A448 to A959
Timer Area	T0000 to T4095		T0000 to T4092	T0000 to T4095
Counter Area	C0000 to C4095		C0000 to C4092	C0000 to C4095
DM Area	D00000 to D32767		D00000 to D32764	D00000 to D32767
EM Area without bank	E00000 to E32767		E00000 to E32764	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32764 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)			
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)			
Constants	---			
Data Registers	DR0 to DR15		---	DR0 to DR15

Area	I	O	D	C
Index Registers	---			
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

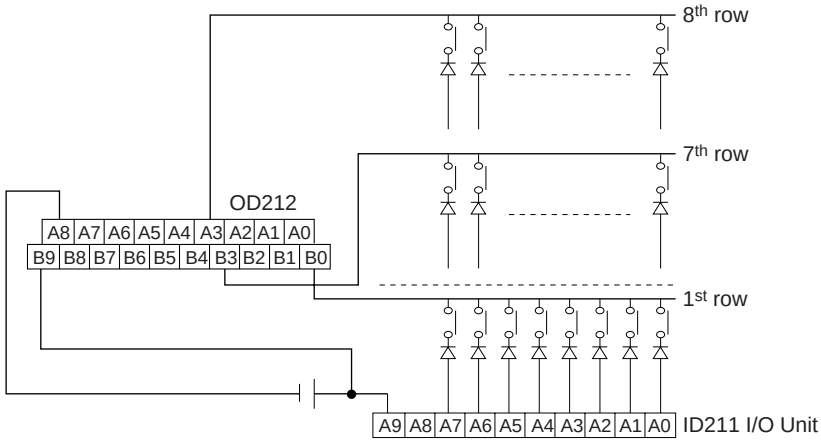
MTR(213) outputs the selection signals to bits 00 to 07 of O, reads the data in order from bits 00 to 07 of I, and stores the 64 bits of data in the 4 words D through D+3. MTR(213) reads the status of the 64-bit matrix every 24 CPU Unit cycles. The One Round Flag (bit 08 of O) is turned ON for one cycle in every 24 cycles after each of the selection signals has been turned ON.

When executed, MTR(213) begins reading the matrix status from the beginning of the matrix, regardless of the point at which the last instruction was stopped.

There is no restriction on the number of times that MTR(213) can appear in the program (unlike the C200HX/HG/HE and CQM1H Series).

External Connections

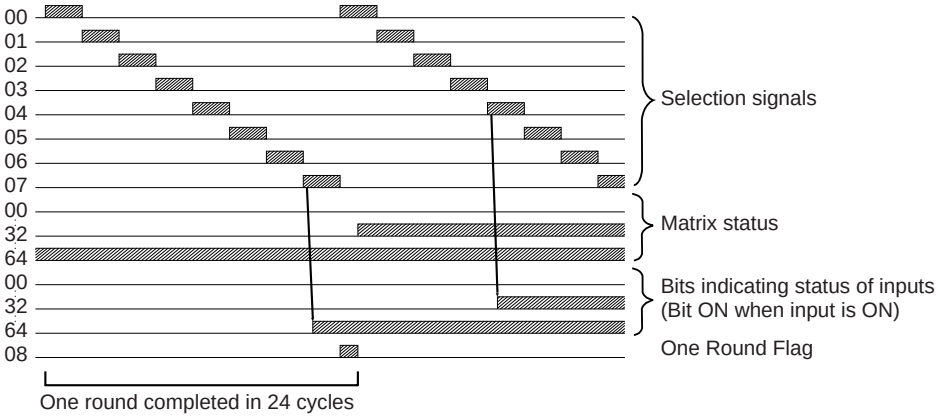
Connect the hexadecimal keypad to Input Unit contacts 0 to 3 and Output Unit contacts 0 to 3, as shown in the following diagram.



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



Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

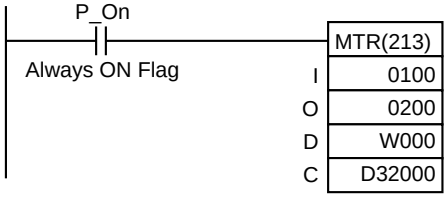
Do not read or write the system word (C) from any other instruction. MTR(213) will not operate correctly if the system word is accessed by another instruction. The system word is not initialized by MTR(213) in the first cycle when program execution starts. If MTR(213) is being used from the first cycle, clear the system word from the program.

MTR(213) will not operate correctly if I/O refreshing is not performed with the Input Unit and Output Unit connected to the external matrix after MTR(213) is executed. Consequently, set the input time constant for the Input Units used for the data line input word to a value that is shorter than the cycle time, or 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)

Example

In this example, MTR(213) reads the 64 bits of data from the 8 × 8 matrix and stores the data in W000 to W003. The 8 × 8 matrix is connected through CIO 0100 (allocated to a CS1W-ID211 16-point DC Input Unit) and CIO 0200 (allocated to a CS1W-OD211 16-point Transistor Output Unit). D32000 is used as the system word.



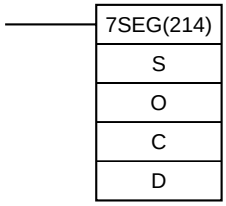
3-23-9 7-SEGMENT DISPLAY OUTPUT – 7SEG(214)

Purpose

Converts the source data (either 4-digit or 8-digit BCD) to 7-segment display data, and outputs that data to the specified output word.

This instruction is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



Variations

Variations		7SEG(214)
	Executed Each Cycle for ON Condition	
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

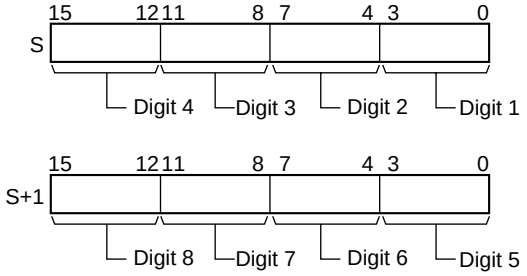
Applicable Program Areas

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

Operands

S: Source Word

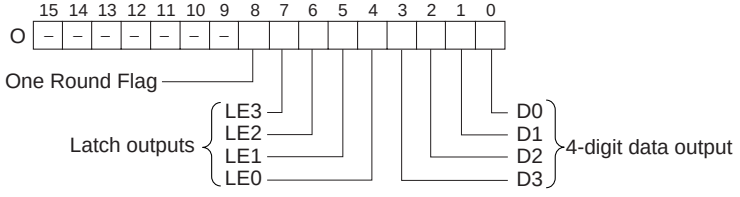
Specify the first source word containing the data that will be converted to 7-segment display data.



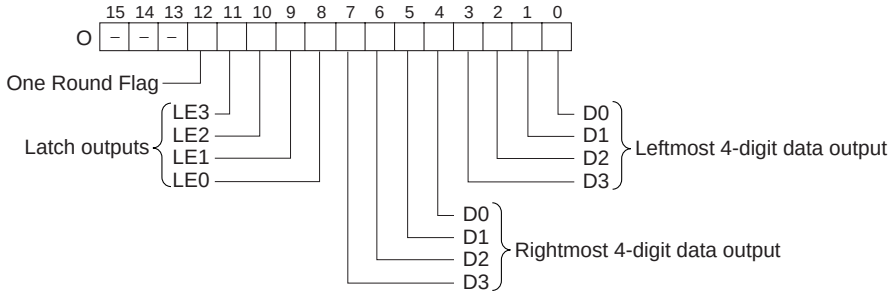
O: Output Word (Data and Latch Outputs)

Specify the output word allocated to the Output Unit and connect the 7-segment display to the Output Unit as shown in the following diagram.

- Converting 4 digits



- Converting 8 digits



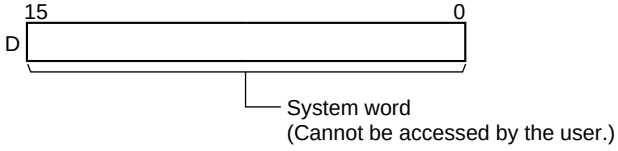
C: Control Data

The value of C indicates the number of digits of source data and the logic for the Input and Output Units, as shown in the following table. (The logic refers to the transistor output's NPN or PNP logic.)

Source data	Display's data input logic	Display's latch input logic	C
4 digits (S)	Same as Output Unit	Same as Output Unit	0000
		Different from Output Unit	0001
	Different from Output Unit	Same as Output Unit	0002
		Different from Output Unit	0003
8 digits (S, S+1)	Same as Output Unit	Same as Output Unit	0004
		Different from Output Unit	0005
	Different from Output Unit	Same as Output Unit	0006
		Different from Output Unit	0007

D: System Word

Specifies a work word used by the instruction. This word cannot be used in any other application.



Operand Specifications

Area	S	O	C	D
CIO Area	CIO 0000 to CIO 6143		---	CIO 0000 to CIO 6143
Work Area	W000 to W511		---	W000 to W511
Holding Bit Area	H000 to H511		---	H000 to H511
Auxiliary Bit Area	A000 to A959	A448 to A959	---	A448 to A959
Timer Area	T0000 to T4095		---	T0000 to T4095
Counter Area	C0000 to C4095		---	C0000 to C4095
DM Area	D00000 to D32767		---	D00000 to D32767
EM Area without bank	E00000 to E32767		---	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		---	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)			
Constants	---	---	0000 to 0007	---
Data Registers	---	DR0 to DR15	---	DR0 to DR15

Area	S	O	C	D
Index Registers	---			
Indirect addressing using Index Registers	IR0 to IR15, –2048 to +2047, IR0 to IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

7SEG(214) reads the source data, converts it to 7-segment display data, and outputs that data (as leftmost 4 digits D0 to D3, rightmost 4 digits D0 to D3, latch output signals LE0 to LE3) to the 7-segment display connected to the output indicated by O. The value of C indicates the number of digits of source data (either 4-digit or 8-digit) and the logic for the Input and Output Units.

7SEG(214) displays the 4-digit or 8-digit data in 12 cycles, and then starts over and continues displaying the data.

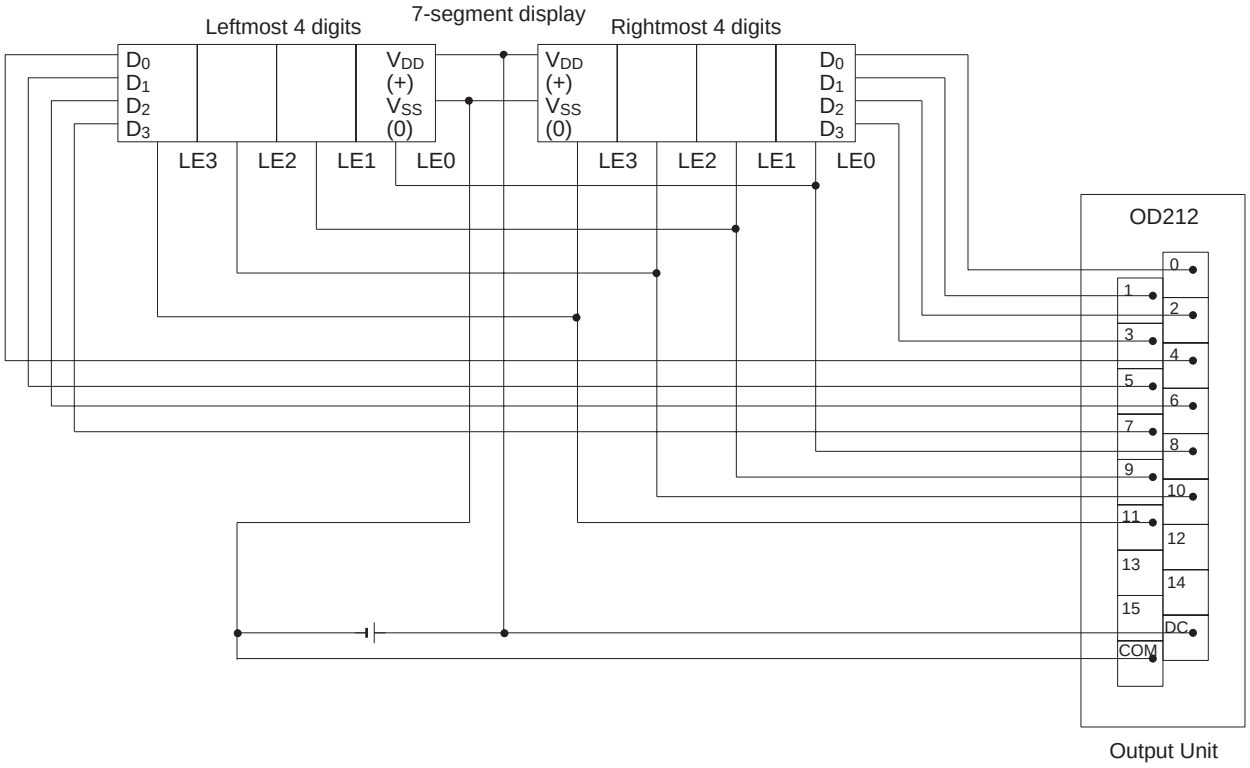
The One Round Flag (bit 08 of O when converting 4 digits, bit 12 of O when converting 8 digits) is turned ON for one cycle in every 12 cycles after 7SEG(214) has turned ON each of the latch output signals. After the 7-segment data is output in 12 cycles, 7SEG(214) starts over and converts the present contents of the source word(s) in the next 12 cycles.

When executed, 7SEG(214) begins on latch output 0 at the beginning of the round, regardless of the point at which the last instruction was stopped.

Even if the connected 7-segment display has fewer than 4 digits or 8 digits in its display, 7SEG(214) will still output 4 digits or 8 digits of data.

External Connections

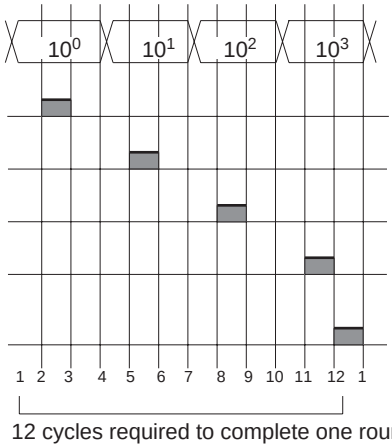
Connect the 7-segment display to the Output Unit as shown in the following diagram. This example shows an 8-digit display. With a 4-digit display, the data outputs (D0 to D3) would be connected to outputs 0 to 3 and the latch outputs (LE0 to LE3) would be connected to outputs 4 to 7. Output point 12 (for 8-digit display) or output point 8 (for 4-digit display) will be turned ON when one round of data has been output, but it is not necessary to connect them unless required by the application.



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	
Latch output 0	04	08	
Latch output 1	05	09	
Latch output 2	06	10	
Latch output 3	07	11	
One Round Flag	08	12	

Note 0 to 3: Data output for word S
4 to 7: Data output for word S+1

Flags

Name	Label	Operation
Error Flag	ER	OFF

Precautions

Do not read or write the system word (D) from any other instruction. 7SEG(214) will not operate correctly if the system word is accessed by another instruction. The system word is not initialized by 7SEG(214) in the first cycle when program execution starts. If 7SEG(214) is being used from the first cycle, clear the system word from the program.

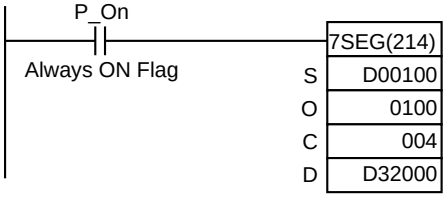
7SEG(214) will not operate correctly if I/O refreshing is not performed with the Output Unit connected to the 7-segment display after 7SEG(214) is executed. Consequently, do not connect the external matrix to the following Units.

- 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)

Example

In this example, 7SEG(214) converts the 8 digits of BCD data in D00100 and D00101 and outputs the data through CIO 0100 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 0004. D32000 is used as the system word, D.



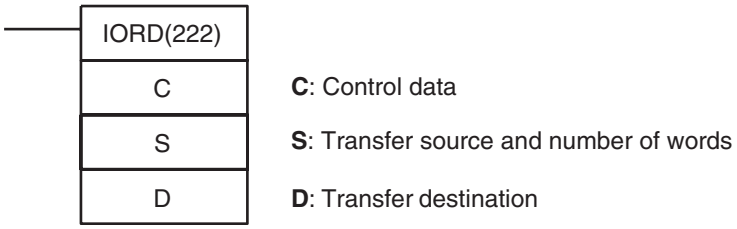
3-23-10 INTELLIGENT I/O READ: IORD(222)

Purpose

Reads the contents of memory area of a Special I/O Unit or CPU Bus Unit (see note).

Note There are restrictions in functionality for CPU Bus Units. Refer to *Restrictions* later in this section.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	IORD(222)
	Executed Once for Upward Differentiation	@IORD(222)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

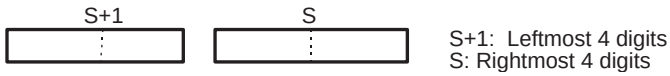
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Depends on Special I/O Unit or CPU Bus Unit.
S: Special I/O Unit: 0000 to 005F hex
(to specify unit numbers 0 to 95)

CPU Bus Unit: 8000 to 800F hex
(to specify unit numbers 0 to F hex)

S+1: Number of words to transfer
(0001 to 0080 Hex, depends on Special I/O Unit or CPU Bus Unit)

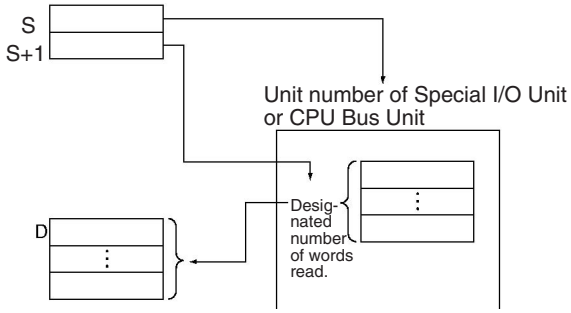


Operand Specifications

Area	C	S	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W510	W000 to W511
Holding Bit Area	H000 to H511	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A958	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	Specified values only	---
Data Registers	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

IORD(222) reads the number of words designated in S+1 from the memory area of the Special I/O Unit or CPU Bus Unit whose unit number is designated by S and outputs the data to D. 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.



Restrictions The following restrictions apply to reading from a CPU Bus Unit.

■ **Restrictions on the CPU Unit**

CS1-H CPU Units

Reading from a CPU Bus Unit is possible only for the following models of CPU Unit and only for CPU Units manufactured on or after 18 April 2003 (lot number 030418 or later).

- CS1G-CPU□□H
- CS1H-CPU□□H

The manufacturing date can be confirmed using the lot number given on the side or bottom of the CPU Unit. Lot numbers indicate the manufacturing date as follows:

YYMMDD nnnn

YY = Rightmost two digits of the year, MM = Month as a numeric value, DD = Day of month, nnnn = Serial number

CJ1-H, CJ1M, and CS1D CPU Units

Reading from a CPU Bus Unit is possible only for CPU Unit Ver. 2.0 or later.

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.

■ **Restrictions on the CX-Programmer**

Unit numbers for CPU Bus Units can be specified for S with CX-Programmer version 3.0 or higher.

Flags

Name	Label	Operation
Error Flag	ER	ON if the number of words to transfer (S) is outside the range of 0001 to 0080 hex. ON if the unit number (S) is outside the range of 0000 to 005F hex or 8000 to 800F hex. ON if 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. ON if a Special I/O Unit with a Special I/O Unit setting error or a Special I/O Unit error is designated. ON if a CPU Bus Unit with a CPU Bus Unit setting error or a CPU Bus Unit error 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	=	ON if reading operation is completed normally. OFF if reading operation is not completed normally.

Precautions

The Equals Flag will turn ON if the reading operation is completed normally.

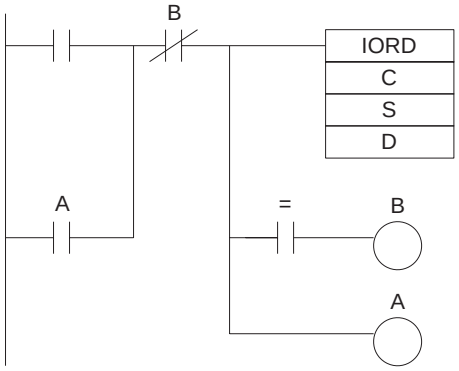
The Equals Flag will turn OFF if the reading operation cannot be completed normally due to the Special I/O Unit or CPU Bus Unit being busy.

Whenever any of the following occur, an error will occur and the Error Flag will turn ON.

- The number of words to transfer (S) is outside the range of 0001 to 0080 (hex).
- The unit number (S) is outside the range of 0000 to 005F hex or 8000 to 800F hex.
- The designated Special I/O Unit is on SYSMAC BUS.
- A Special I/O Unit or CPU Bus Unit not affected by IORD(222) is designated.
- A Special I/O Unit with a Special I/O Unit setting error or a Special I/O Unit error is designated.
- A CPU Bus Unit with a CPU Bus Unit setting error or a CPU Bus Unit error is designated.

When IORD(222) is executed, the execution results are reflected in the condition flags. In particular, the Equals Flag turns ON when reading is completed. Input the condition flags such as the Equals Flag with output branching from the same input conditions as the IORD(222) instruction.

If the Special I/O Unit or CPU Bus Unit is busy, the reading operation will not be executed. Use the Equals Flag to create a self-maintaining program, as shown below, so that IORD(222) will be executed with each cycle until the reading operation is executed.



When the input condition is met, self maintenance is performed by output A and IORD(222) is executed with each cycle until the Equals Flag turns ON. When the reading is completed and the Equals Flag turns ON, output B turns ON and the self maintenance is cleared.

Be sure to place condition flags directly after IORD(222) instructions, and not after any other instructions. If a condition flag is placed after another instruction, it will be affected by the execution results of that instruction.

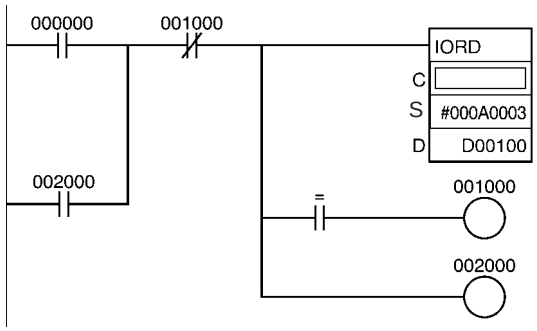
IORD(222) can be used in an interrupt task, which allows high-speed processing of specific I/O data with an interrupt. If IORD(222) is used in an interrupt task, always disable cyclic refreshing of the specified Special I/O Unit by turning ON the corresponding Special I/O Unit Cyclic Refreshing Disable 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 (A40213) will go ON in the following cases.

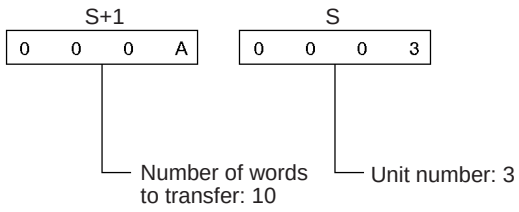
- Words allocated to the same Special I/O Unit were already refreshed by IORF(097) or FIORF(225) (CJ1-H-R CPU Units only).
- Words allocated to the same Special I/O Unit were read or written by IORD(222) or IOWR(223).

Example

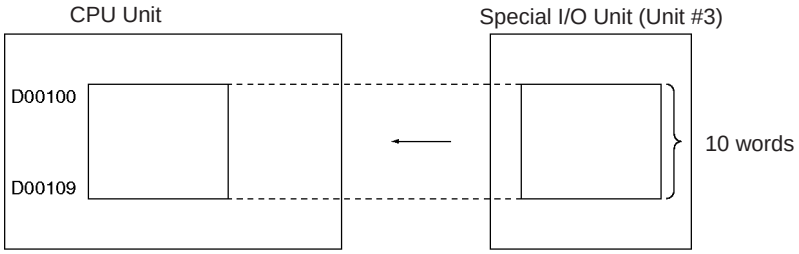
In this example, IORD(222) is used to read data.



When CIO 000000 is turned ON, 10 words are read from the Special I/O Unit with unit number 3, and are stored in D00100 to D00109.



The control code (C) varies depending on the Special I/O Unit.



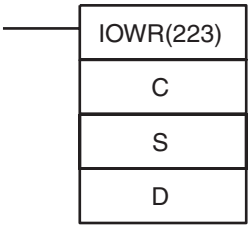
3-23-11 INTELLIGENT I/O WRITE: IOWR(223)

Purpose

Outputs the contents of the CPU Unit's I/O memory area to a Special I/O Unit or CPU Bus Unit (see note).

Note There are restrictions in functionality for CPU Bus Units. Refer to *Restrictions* later in this section.

Ladder Symbol



C: Control data

S: Transfer source and number of words

D: Transfer destination and number of words

Variations

Variations		
	Executed Each Cycle for ON Condition	IOWR(223)
	Executed Once for Upward Differentiation	@IOWR(223)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

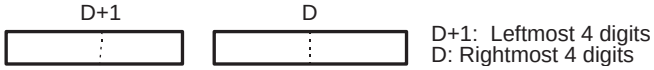
Applicable Program Areas

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

Operands

C: Depends on Special I/O Unit or CPU Bus Unit.
D: Special I/O Unit: 0000 to 005F hex
(to specify unit numbers 0 to 95)
CPU Bus Unit: 8000 to 800F hex
(to specify unit numbers 0 to F hex)

D+1: Number of words to transfer
(0000 to 0080 Hex, depends on Special I/O Unit or CPU Bus Unit)

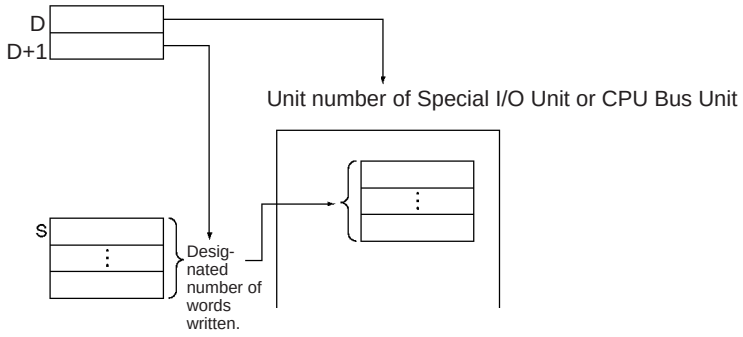


Operand Specifications

Area	C	S	D
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W511		W000 to W510
Holding Bit Area	H000 to H511		H000 to H510
Auxiliary Bit Area	A000 to A959		A000 to A958
Timer Area	T0000 to T4095		T0000 to T4094
Counter Area	C0000 to C4095		C0000 to C4094
DM Area	D00000 to D32767		D00000 to D32766
EM Area without bank	E00000 to E32767		E00000 to E32766
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32766 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)		Specified values only
Data Registers	DR0 to DR15	---	---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

IOWR(223) writes the designated number of words (D) from the first source word (designated by S) onwards and outputs them to the Special I/O Unit or CPU Bus Unit that has the unit number designated by D. Only Special I/O Units or CPU Bus Units mounted on CPU Racks or Expansion I/O Racks can be designated.



Restrictions The following restrictions apply to reading from a CPU Bus Unit.

■ **Restrictions on the CPU Unit**

CS1-H CPU Units

Writing to a CPU Bus Unit is possible only for the following models of CPU Unit and only for CPU Units manufactured on or after 18 April 2003 (lot number 030418 or later).

- CS1G-CPU□□H
- CS1H-CPU□□H

The manufacturing date can be confirmed using the lot number given on the side or bottom of the CPU Unit. Lot numbers indicate the manufacturing date as follows:

YYMMDD nnnn

YY = Rightmost two digits of the year, MM = Month as a numeric value, DD = Day of month, nnnn = Serial number

CJ1-H, CJ1M, and CS1D CPU Units

Writing to a CPU Bus Unit is possible only for CPU Unit Ver. 2.0 or later.

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.

■ **Restrictions on the CX-Programmer**

Unit numbers for CPU Bus Units can be specified for S with CX-Programmer version 3.0 or higher.

Flags

Name	Label	Operation
Error Flag	ER	ON if the number of words to transfer (D) is outside the range of 0001 to 0080 hex. ON if the unit number (D) is outside the range of 0000 to 005F hex or 8000 to 800F hex. ON if S is designated by a constant when the number of words to be transferred (D+1) is not 0001 hex. ON if 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. ON if a Special I/O Unit with a Special I/O Unit setting error or a Special I/O Unit error is designated. ON if a CPU Bus Unit with a CPU Bus Unit setting error or a CPU Bus Unit error 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	=	ON if writing operation is completed normally. OFF if writing operation is not completed normally.

Precautions

When “0001” is designated for the number of words to be transferred (D+1), the data for S can be designated by a constant. If a constant is designated for S when the number of words to be transferred is not “0001,” an error will occur and the Error Flag will turn ON.

The Equals Flag will turn ON if the writing operation is completed normally.

The Equals Flag will turn OFF if the writing operation cannot be completed normally due to the Special I/O Unit or CPU Bus Unit being busy.

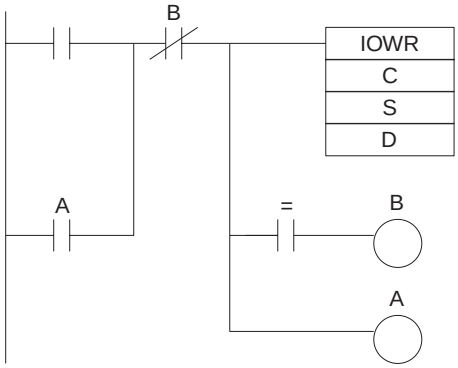
Whenever any of the following occur, an error will occur and the Error Flag will turn ON.

- There is an I/O Unit verification error, a Special I/O Unit setting error, a Special I/O Unit setting error, or a Special I/O Unit error at the Special I/O Unit.
- There is an I/O Unit verification error, a CPU Bus Unit setting error, a CPU Bus Unit setting error, or a CPU Bus Unit error at the CPU Bus Unit.
- The number of words to transfer (D) is outside the range of 0001 to 0080 (hex).
- The unit number (D) is outside the range of 0000 to 005F hex or 8000 to 800F hex.
- The designated Special I/O Unit is on SYSMAC BUS.
- A Special I/O Unit or CPU Bus Unit not affected by IOWR(223) is designated.
- A Special I/O Unit with a Special I/O Unit setting error or a Special I/O Unit error is designated.
- A CPU Bus Unit with a CPU Bus Unit setting error or a CPU Bus Unit error is designated.

When IOWR(223) is executed, the execution results are reflected in the condition flags. In particular, the Equals Flag turns ON when reading is completed. Input the condition flags such as the Equals Flag with output branching from the same input conditions as the IOWR(223) instruction.

If the Special I/O Unit or CPU Bus Unit is busy, the writing operation will not be executed. Use the Equals Flag to create a self-maintaining program, as

shown below, so that IOWR(223) will be executed with each cycle until the writing operation is executed.



When the input condition is met, self maintenance is performed by output A and IOWR(223) is executed with each cycle until the Equals Flag turns ON. When the writing is completed and the Equals Flag turns ON, output B turns ON and the self maintenance is cleared.

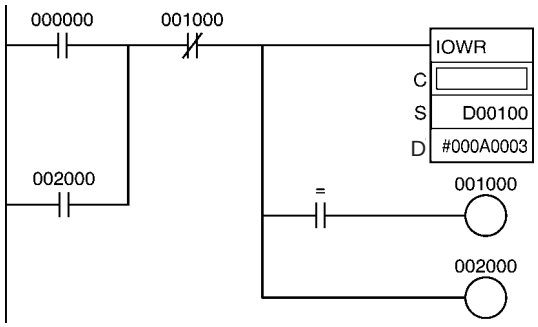
Be sure to place condition flags directly after IOWR(223) instructions, and not after any other instructions. If a condition flag is placed after another instruction, it will be affected by the execution results of that instruction.

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.

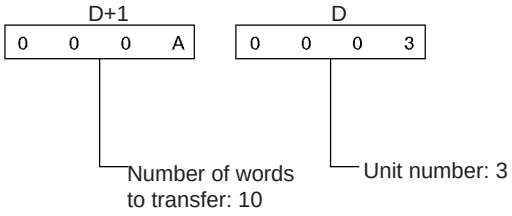
When cyclic refreshing of the specified Special I/O Unit is enabled in the PLC Setup (the corresponding Special I/O Unit Cyclic Refreshing Disable Bit is OFF), a non-fatal Duplicate Refresh Error will occur and the Interrupt Task Error Flag (A40213) will go ON in the following cases.

- Words allocated to the same Special I/O Unit were already refreshed by IORF(097) or FIORF(225) (CJ1-H-R CPU Units only).
- Words allocated to the same Special I/O Unit were read or written by IORD(222) or IOWR(223).

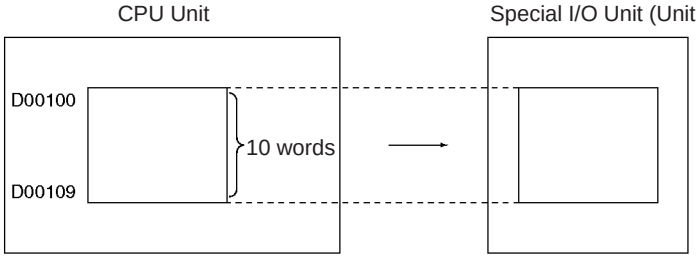
Example In this example, IOWR(223) is used to write data.



When CIO 000000 is turned ON, the 10 words in D00100 to D00109 are written to the Special I/O Unit.



The control code (C) varies depending on the Special I/O Unit.



3-24 Serial Communications Instructions

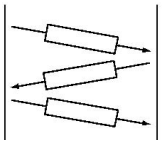
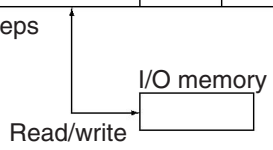
This section describes instructions used for serial communications.

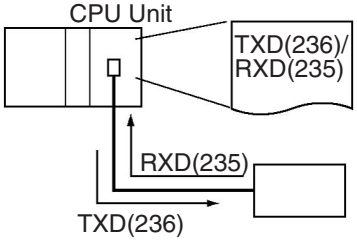
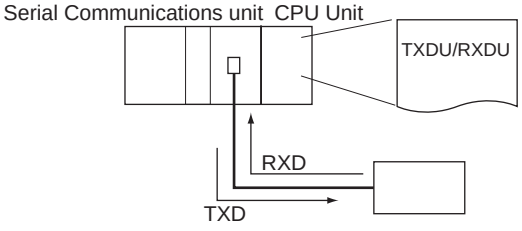
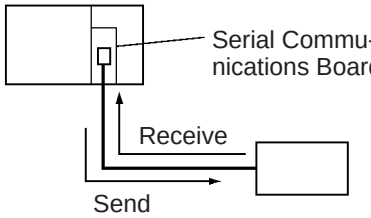
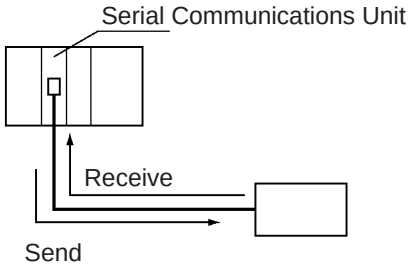
Instruction	Mnemonic	Function code	Page
PROTOCOL MACRO	PMCR	260	974
TRANSMIT	TXD	236	983
RECEIVE	RXD	235	993
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	256	1005
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	255	1013
CHANGE SERIAL PORT SETUP	STUP	237	1021

3-24-1 Serial Communications

There are two types of serial communications instruction. The TXD(236), RXD(235), TXDU(256), and RXDU(255) instructions send and receive data in no-protocol (custom) communications with an external device. PMCR(260) sends and receives data using user-defined protocols with an external device. The difference is shown in the following tables.

Note
1. The TXD(236) and RXD(235) instructions transfer data only through the CPU Unit's built-in serial port or a serial port on a Serial Communications Board (Ver. 1.2 or later).
2. The TXDU(256) and RXDU(255) instructions transfer data only through a Serial Communications Unit (Ver. 1.2 or later).

Instructions	Communications frames	Function
TXD(236), RXD(235), TXDU(256), and RXDU(255)	Any of the following can be used. No Start or End Code: Data Start and End Code: ST | Data | ED Only Start Code: ST | Data CR+LF End Code: Data | CR | LF Only End Code: Data | ED Start and CR+LF End Code: ST | Data | CR | LF	Sends or receives data in one direction only. A send delay can be set.
PMCR(260)	The following type of frames (messages) can be created to meet the requirements of the external device. Header | Address | Data | Error check | Terminator Communications steps can be created.  	Up to 16 steps can be defined for sending and receiving. Steps can be changed and retry processing performed based on responses. Communications monitoring times can be set. Symbols can be read/written for the PLC. Repeat symbols can be used. Other.

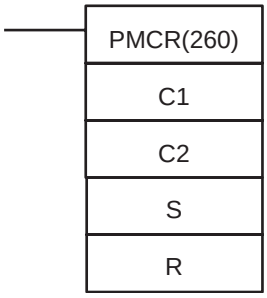
Instructions	Mode	Communications ports	
TXD(236) and RXD(235)	No-protocol (custom)	Serial port in CPU Unit or Serial Communications Board  TXD(236) and RXD(235) use serial ports on the CPU Unit or Serial Communications Boards (Ver. 1.2 or later).	
TXDU(256) and RXDU(255)	No-protocol (custom)	Serial Port of Serial Communications Unit (Version 1.2 or later) 	
PMCR(260)	Protocol macro	Serial Communications Board (CS Series only) 	Serial Communications Unit 

3-24-2 PROTOCOL MACRO: PMCR(260)

Purpose

Calls and executes a communications sequence registered in a Serial Communications Board (CS Series only) or Serial Communications Unit.

Ladder Symbol



C1: Control word 1

C2: Control word 2

S: First send word

R: First receive word

Variations

Variations		
	Executed Each Cycle for ON Condition	PMCR(260)
	Executed Once for Upward Differentiation	@PMCR(260)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

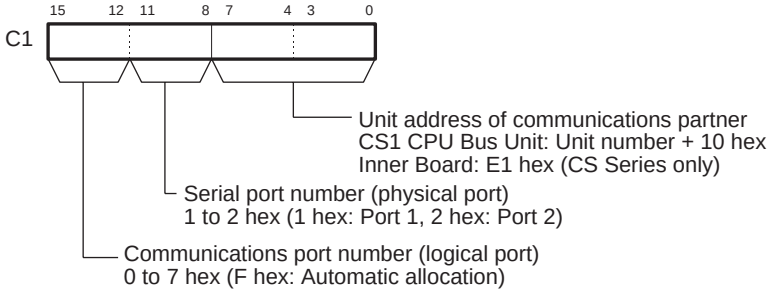
Applicable Program Areas

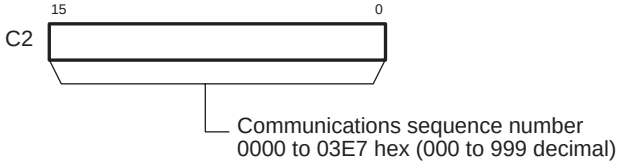
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C1: Control Word 1 and C2: Control Word 2

The contents of the two control words are shown below.





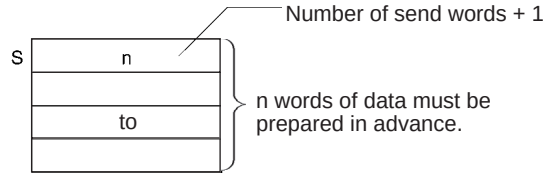
Note Refer to *Automatic Allocation of Communications Ports* on page 1032 for details on using automatic allocation of the communications port number (logical port).

S: First Send Word and Send Area

The first word of the words required to send data is specified. S contains the number of words to be sent +1 (i.e., including the S word) and send data starts in S+1. Between 0000 and 00FA hex (0 and 250 decimal) words can be sent.

If there is no operand specified in the execution sequence, such as a direct or linked word, specify the constant #0000 for S. If a word address or register is

specified, the data in the word or register must always be 0000. An error will occur and the Error Flag will turn ON if any other constant or a word address is given and PMCR(260) will not be executed.



R: First Receive Word and Receive Area

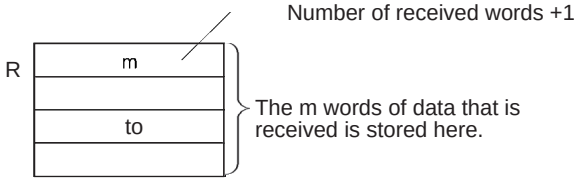
Received data is automatically stored in words starting with R+1 and the number of words received plus R (i.e., including R) is automatically written to R between 0000 and 00FA hex (0 and 250 decimal).

Setting Before Executing PMCR

Set the data specified by m (beginning with D) as the initial data for the receive buffer (backup data for receive failure). Data m can be set to 0002 to 00FA (hex) (2 to 255). If 0000 (hex) or 0001 (hex) is specified for m, the initial value of the receive buffer will be cleared to 0.

Always set a word address for R even if there is no receive data. If a constant is set, an error will occur, the Error Flag will turn ON, and PMCR(260) will not be executed. If there is no receive data, R will not be used and can be used for other purposes.

If there is no operand specified in the execution sequence, such as a direct or linked word, specify the constant #0000 for R. If a word address or register is specified, the data in the word or register must always be 0000.



Operand Specifications

Area	C1	C2	S	R
CIO Area	CIO 0000 to CIO 6143			
Work Area	W000 to W511			
Holding Bit Area	H000 to H511			
Auxiliary Bit Area	A000 to A447 A448 to A959			A448 to A959
Timer Area	T0000 to T4095			
Counter Area	C0000 to C4095			
DM Area	D00000 to D32767			
EM Area without bank	E00000 to E32767			
EM Area with bank	En_00000 to En_32767 (n = 0 to C)			
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)			

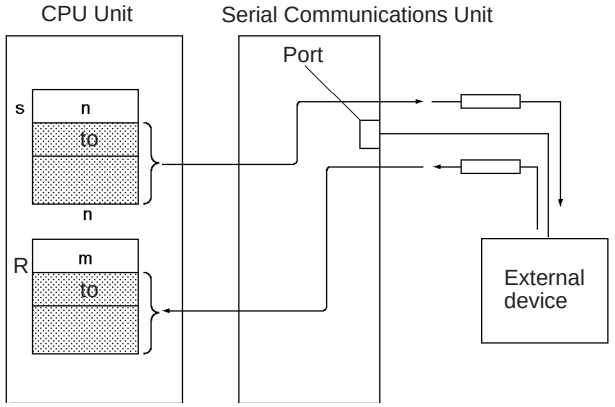
Area	C1	C2	S	R
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)			
Constants	Specified values only	0000 to 03E7Hex (0 to 999)	#0000 (binary)	
Data Registers	DR0 to DR15		---	
Index Registers	---			
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

PMCR(260) will execute the communications sequence specified in C2 using the logical port specified in bits 12 to 15 of C1 and the physical port specified in bits 8 to 11 of C1 for the unit address specified in bits 0 to 7 of C1.

If a symbol is specified as the operand for a send message, the number of send words specified in S and beginning from S+1 will be used as the send area. If a symbol is specified as the operand for a receive message, receive data is placed in memory staring with R+1 and the number of words received is automatically written to R if the transmission is successful.

If the transmission fails, the data (R+1 onward) set before PMCR(260) was executed will be read from the receive buffer and stored in the R+1 onward again.



Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the specified logical port when PMCR(260) is executed. ON if C1 is not within the specified ranges. (Error flag will not turn ON if the C2 data is outside the specified ranges. The end code will be stored in the Communications Port Completion Code (A203 to A210) of the auxiliary area.) ON if the number of words of S or R exceeds 249 (when words are specified). OFF in all other cases.

Precautions

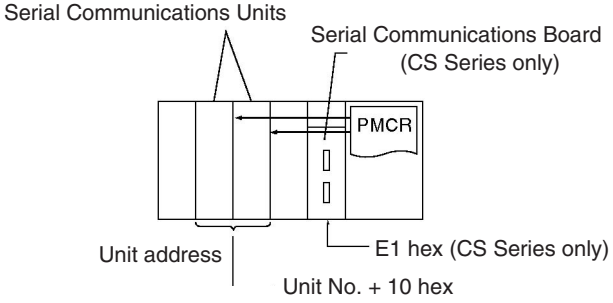
The data in the send area specified with S is actually sent using the symbol read option, R( ), in a send message.

Data is actually received to the receive area specified by R using the symbol write option, W( ), in a receive message.

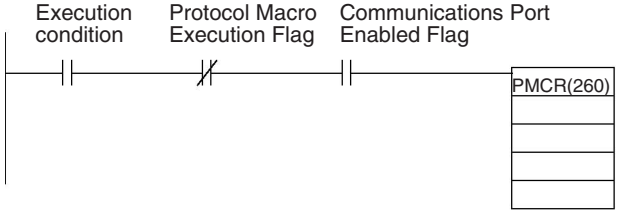
Refer to the *CX-Protocol Operation Manual* (W344) for procedures for designating symbols R( ) and W( ).

PMCR(260) can be executed for a serial communications port on a Serial Communications 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.

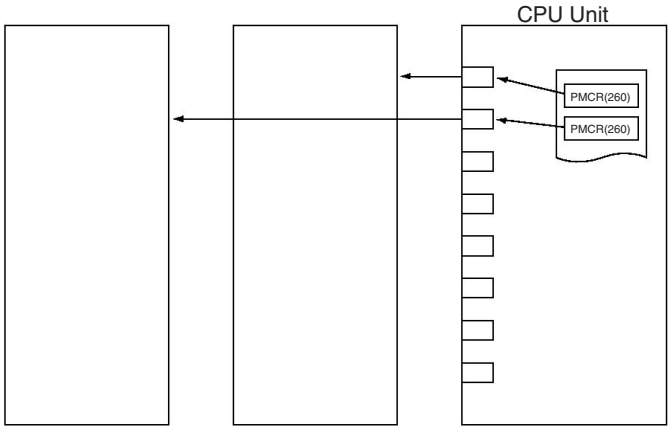
Unit/Board	Unit address
Serial Communications Board (CS Series only)	E1 hex
Serial Communications Unit	Unit number + 10 hex



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 N.O. input in the execution condition for PMCR(260), as shown in the above diagram.



The Error Flag will turn ON in the following cases.

- The corresponding Communications Port Enable Flag is OFF for the specified logical port (0 to 7) when PMCR(260) is executed.
- C1 is not within the specified ranges.

Designation of Receive Area

Before executing PMCR(260), users must set backup data in the receive area for receive processing failure. Once the PMCR(260) is executed, the data in the receive buffer is automatically stored in the receive area. One example of the backup data application is as follows: A certain value (backup data) is set in advance so that the present value will not be read as zero when transmission failure occurs while protocol is being executed for reading the present value of a controller.

Related Flags and Words

The following flags and words can be used as required when executing PMCR(260).

Auxiliary Area

Name	Address	Contents
Communications Port Enabled Flag	A20200 to A20207	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.

Name	Address	Contents
Communications Port Error Flag	A21900 to A21907	ON when an error occurs in network communications. Bits 00 to 07 correspond to logical ports 0 to 7, respectively. Flag status will be maintained until the next network communications start. The flag will turn OFF when communications start again even if an error occurred for the last execution.
Communications Port Completion Codes	A203 to A210	Contains the completion code stored when network communications are performed. Words A203 to A210 correspond to logical ports 0 to 7, respectively. The completion code will be 00 while the communications instruction is being executed. The new response code will be stored when execution has been completed. The contents of these words is cleared when operation is started.

Communications Responses

Code	Contents
1106 (hex)	No corresponding program number Specified Send/Receive Sequence No. that has not been registered. Modify the Send/Receive Sequence No. or add the number using the CX-Programmer.
2201 (hex)	Not operable due to protocol execution Since one protocol macro has already been executed, no further execution is accepted. Add NC condition to program for the Protocol Macro Execution Flag.
2202 (hex)	Not operable due to stoppage Since the protocol is being switched, no further execution is accepted. Add NC condition to program for the Serial Setting Change Flag.
2401 (hex)	No registration table An error has occurred in the protocol macro data or data is being transmitted. Transmit the protocol macro data using the CX-Programmer.
Others	Refer to the *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 190915	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 191915	

CPU Bus Unit Area

n = 1500 + 25 x unit number

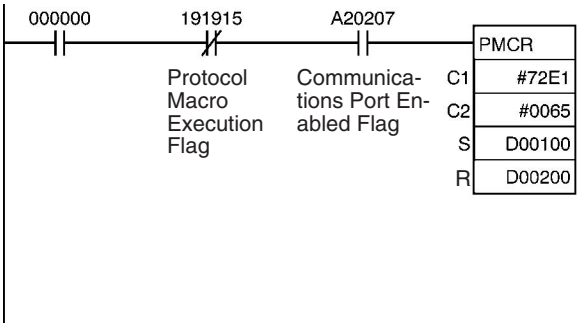
Name	Address	Contents
Port 1 Protocol Macro Execution Flag	Bit 15 of CIO n+9	ON when PMCR(260) is executed. The flag will remain OFF if execution fails. The flag will turn OFF when the communications sequence has been completed (either an end or abort).
Port 2 Protocol Macro Execution Flag	Bit 15 of CIO n+19	

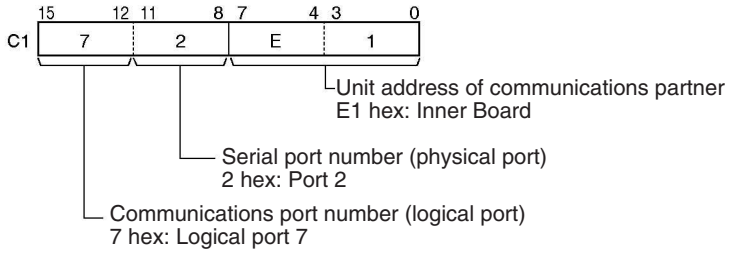
Examples

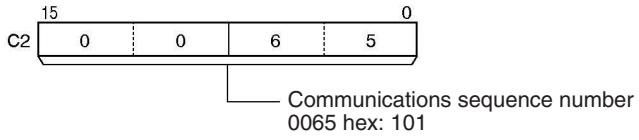
When CIO 0000 is ON in the following example, communications sequence No. 101 (0065 hex) will be executed as long as the Communications Port Enabled Flag for port 7 (A20207) is ON and the Port 1 Protocol Macro Execution Flag (CIO 190915) is OFF.

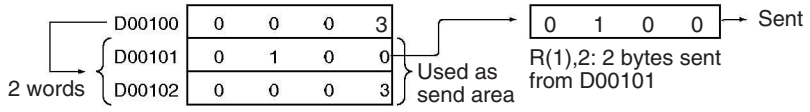
If an operand is specified for the symbol in a send message, 2 words of data starting from D00101 will be used as the send area (because the contents of D00100 is #0003).

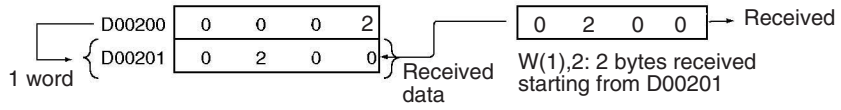
If an operand is specified for the symbol in a receive message, 2 words of data will be stored starting from D00201 and the number of words received +1 will be written to D00200.











Note As shown above, the symbol read option, R( ), in the send message or the symbol write option, W( ), actually sends/receives data.

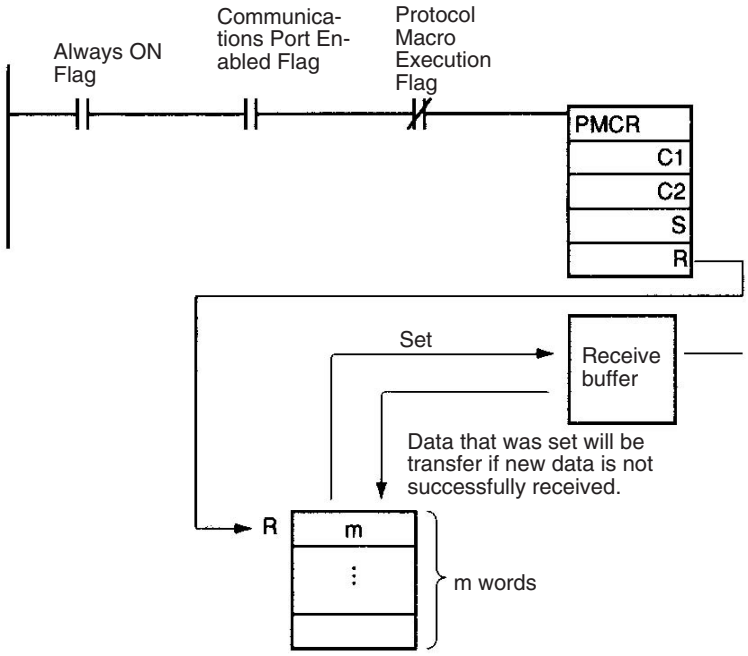
Holding the Receive Area

The receive buffer is cleared to all zeros immediately before a communications sequence is executed for PMCR(260). If programming such as that shown below is used to periodically read PV data or other values and data cannot be read due to a reception error or other cause, the data being read will be cleared until the next successful read.

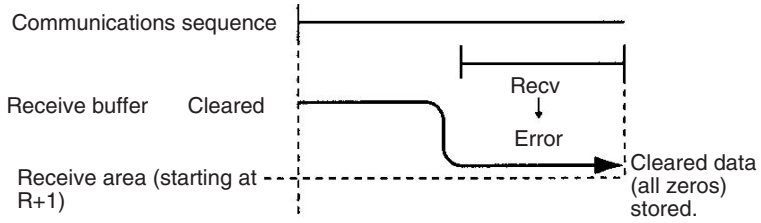
A function is provided to maintain the data in the receive area even when a reception error occurs. If this function is used, data will be transferred from the first m words of the receive area to the receive buffer after the buffer is cleared to all zeros but before the communications sequence is executed. This prevents the receive area from being temporarily cleared to all zeros by writing the most recent receive data when new receive data is not successfully obtained.

Specify the number of words of the receive area to be maintained as the value m. If 0 or 1 is specified, the holding function will be disabled and the receive area will be cleared to all zeros.

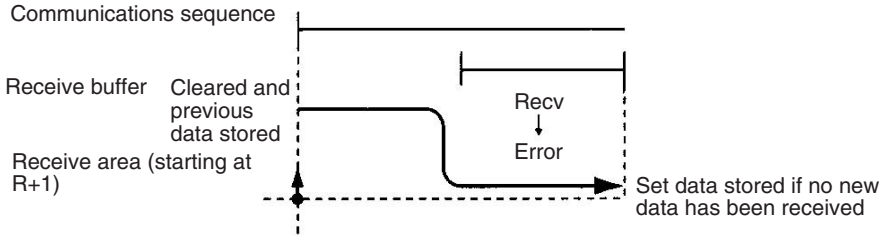
The following programming example shows the instructions used to constantly or periodically execute PMCR(260) to read data through a single receive operation.



Receive Area Not Held



Receive Area Held

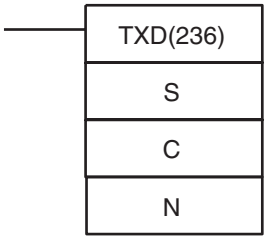


3-24-3 TRANSMIT: TXD(236)

Purpose

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).

Ladder Symbol



S: First source word

C: Control word

N: Number of bytes
0000 to 0100 hex (0 to 256)

Variations

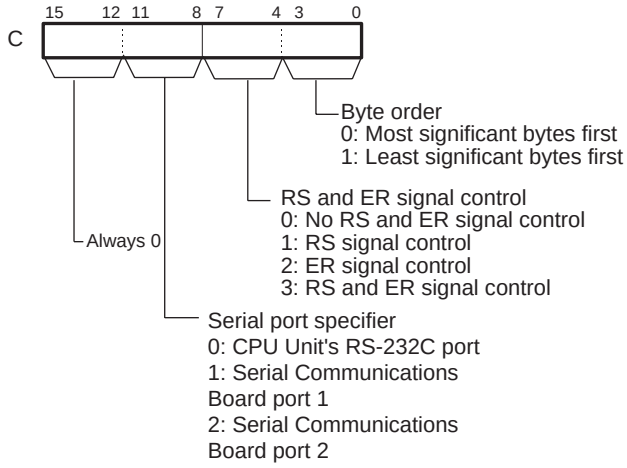
Variations	**Executed Each Cycle for ON Condition**	TXD(236)
	Executed Once for Upward Differentiation	@TXD(236)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

The contents of the control word, C, is as shown below.



Operand Specifications

Area	S	C	N
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A447 A448 to A959		
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		

Area	S	C	N
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	#0000 to #0100 (binary) or &0 to &256 (decimal)
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

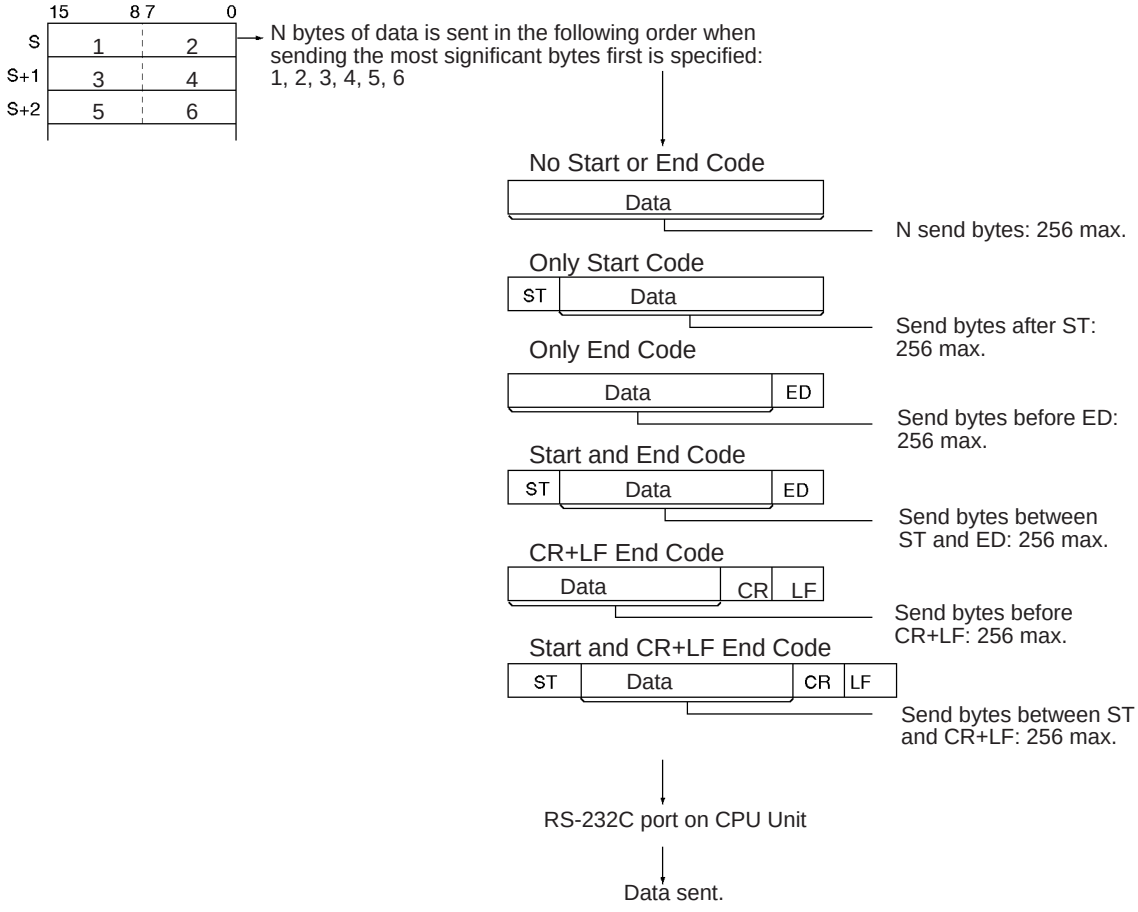
TXD(236) reads N bytes of data from words S to S+(N÷2)–1 and outputs the raw data in no-protocol mode from the CPU Unit's built-in RS-232C port or one of the Serial Communications Board's serial ports. (The output port is specified with bits 8 to 11 of C.)

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).

Data can be sent only when the port's Send Ready Flag is ON. 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.

Up to 259 bytes can be sent, including the send data (N = 256 bytes max.), the start code, and the end code.

The following diagram shows the order in which data is sent and the contents of the send frame for various start and end code settings.



Flags

Name	Label	Operation
Error Flag	ER	ON if the 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. (See note.) ON if a TXD(236) was executed for a serial port on a Serial Communications Board that was being restarted. **Note** The Error (ER) Flag will turn ON immediately after another TXD(236) or RXD(235) instruction in the interrupt task. OFF in all other cases.

Precautions

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.

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).

- Start code: None or 00 to FF hex.
- End code: None, CR+LF, or 00 to FF hex.

The data will be sent with any start and/or end codes specified in the PLC Setup 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 can be sent only when the port's Send Ready Flag is ON. (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.)

Data is sent in the order specified in C.

Nothing will be sent if 0 is specified for N.

If RS signal control is specified in C, bit 15 of S will be used as the RS signal.

If ER signal control is specified in C, bit 15 of S will be used as the ER signal.

If RS and ER signal control is specified in C, bit 15 of S will be used as the RS signal and bit 14 of S will be used as the ER signal.

If 1, 2, or 3 hex is specified for RS and ER signal control in C, TXD(236) will be executed regardless of the status of the Send Ready Flag (A39205, A35605, or A35613 depending on the port being used).

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 (A42404) and the Error Flag will turn ON.

An error will occur and the Error Flag will turn ON in the following cases.

- The CPU Unit's RS-232C port is specified, but no-protocol mode is not set for the port in the PLC Setup.
- One of the Serial Communications Board's serial ports is specified, but no-protocol mode is not set for the port in the allocated DM Setup Area.
- 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).
- The value of C is not within range.
- The value for N is not between 0000 and 0100 hex.
- A send was attempted when the Send Ready Flag was OFF. (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.)
- TXD(236) or RXD(235) was being executed for the Serial Communications Board in the cyclic task, the cyclic task was interrupted, and another TXD(236) or RXD(235) instruction was executed for the Serial Communications Board in the interrupt task.
- TXD(236) was executed for a serial port on a Serial Communications Board that was being restarted.

Note 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.)

Related Flags and Words

The following PLC Setup settings and Auxiliary Area flag can be used as required when executing TXD(236).

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: None 1: Use end code. 2: Use CR+LF.
	0 to 7	No-protocol Mode Number of bytes of Data	00: 256 bytes 01 to FF: 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: None 1: Use start code.
		8 to 11	No-protocol Mode End Code Specifier	0: None 1: Use end code. 2: Use CR+LF.

Auxiliary Area

Send Ready Flags

Port	Address	Contents
CPU Bus Unit's built-in RS-232C Port	A39205	ON when data can be sent in the no-protocol mode.
Serial Communications Board port 1	A35605	
Serial Communications Board port 2	A35613	

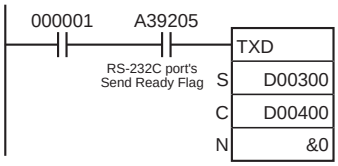
Inner Board Flags for Serial Communications Board (Ports 1 and 2)

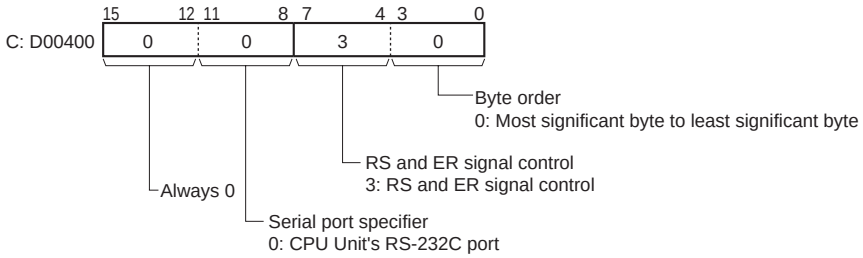
Name	Address	Contents
Inner Board Service Disabled Flag	A42404	ON when TXD(236) is executed for a Serial Communications Board that does not support no-protocol mode (a Board without a version number).

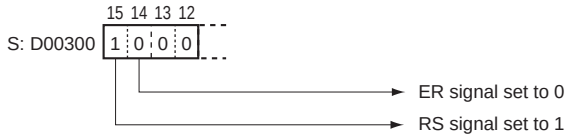
Examples

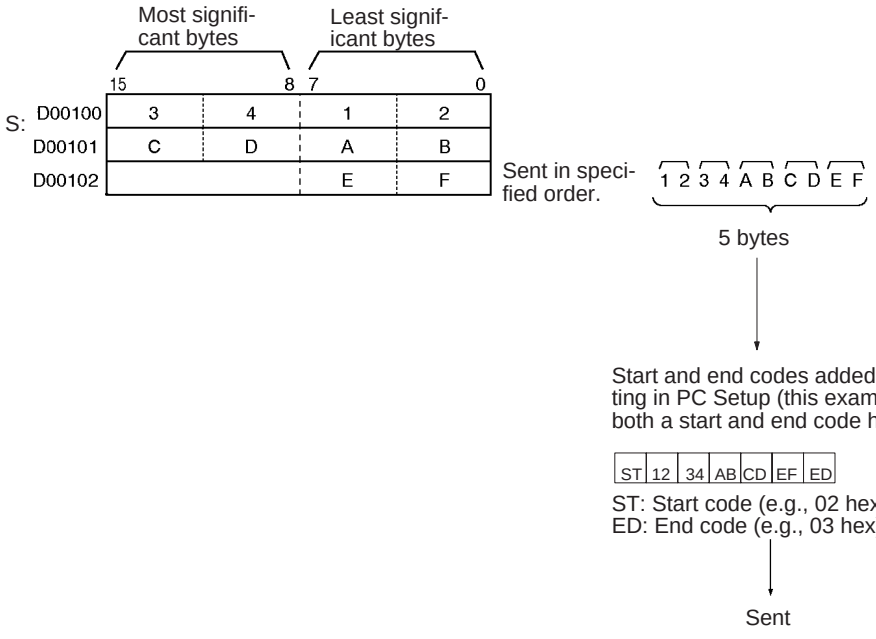
■ Example 1: Sending Data

When CIO 000001 and the RS-232C port's Send Ready Flag (A39205) are ON in the following example, the RS signal is set according to the status of D00300 bit 15 and the ER signal is set according to the status of D00300 bit 14.



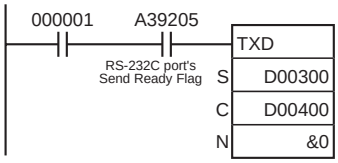


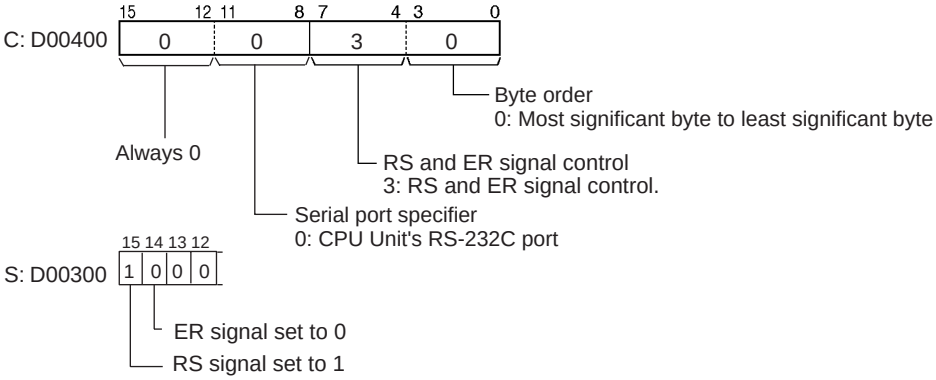




■ Example 2: Performing Signal Control

When CIO 000001 and the RS-232C port's Send Ready Flag (A39205) are ON in the following example, the RS signal is set according to the status of D00300 bit 15 and the ER signal is set according to the status of D00300 bit 14.

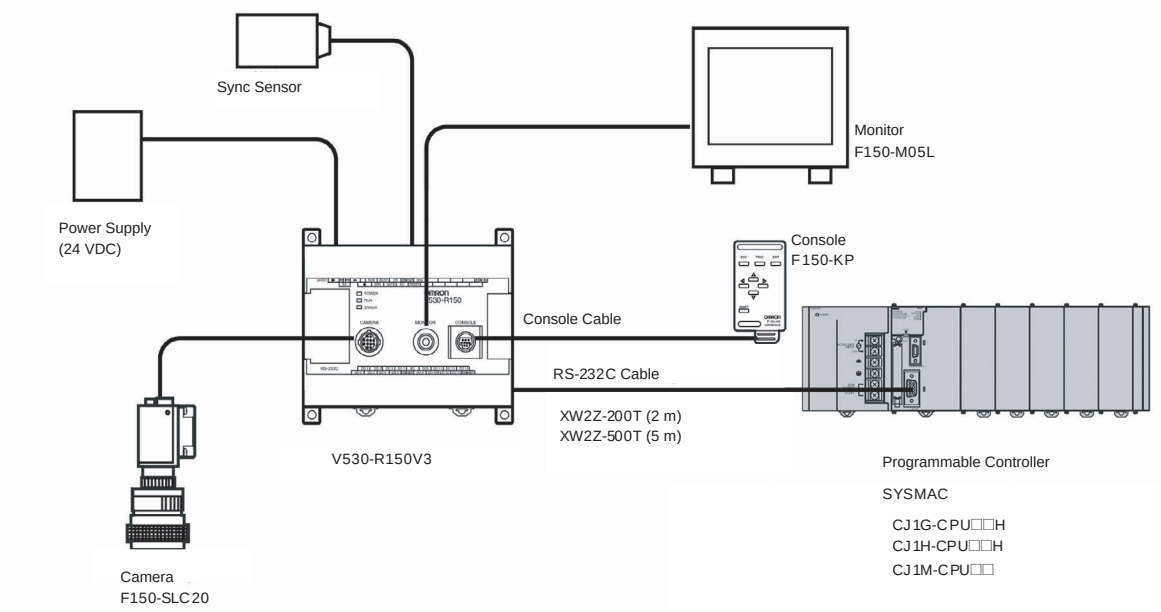




■ Example 3: Sending Data to a Code Reader

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 as given in the following table. These are the default settings.

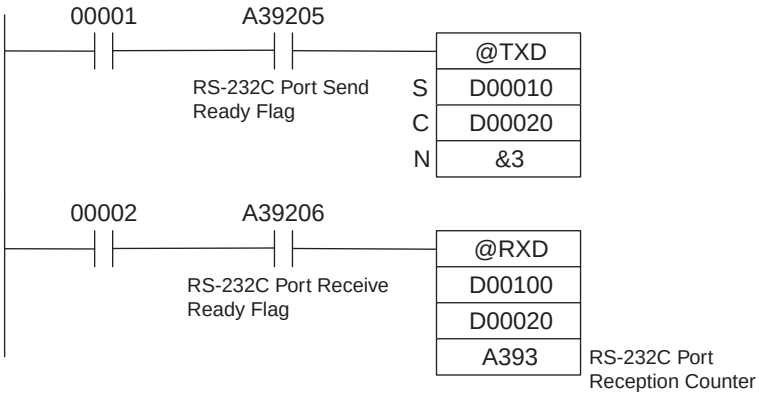
Item	Setting
Communications mode	No-protocol
Baud rate	38,400 bps

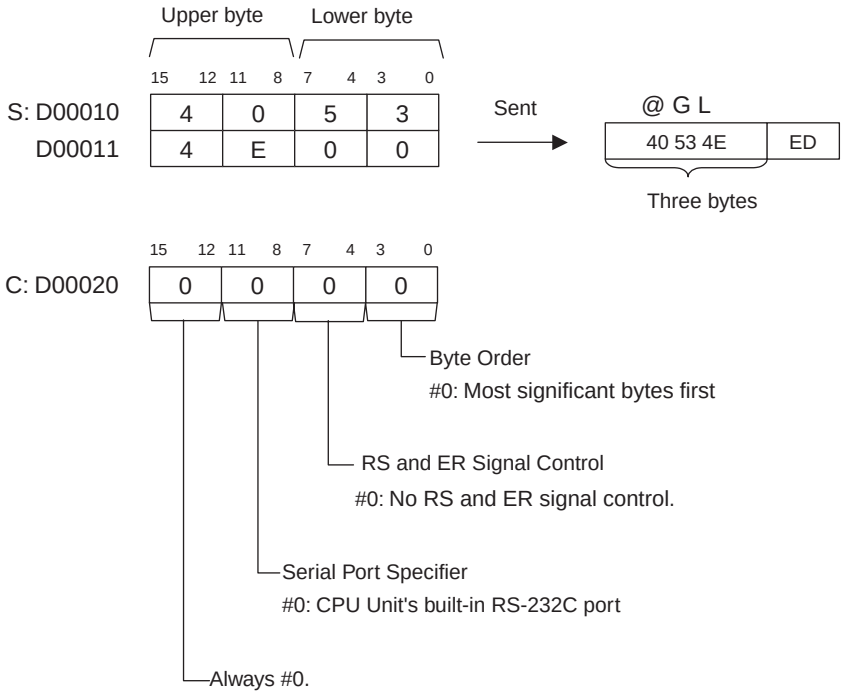
Item	Setting
Data bit length	8 bits
Parity	None
Stop bits	1
Start code	None
End code	#000D (CR)

Set the PLC communications settings to the same values in the PLC Setup. Only the end code needs to be set.

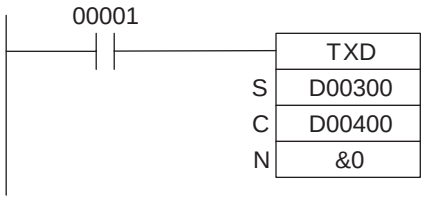
Programming Example

If CIO 000001 turns ON while the RS-232C Port Send Ready Flag (A39205) is ON, three bytes of data starting from the upper byte of D00010 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.

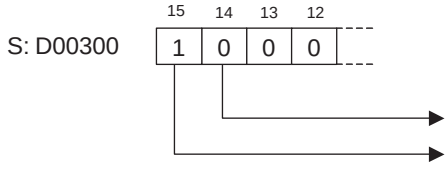


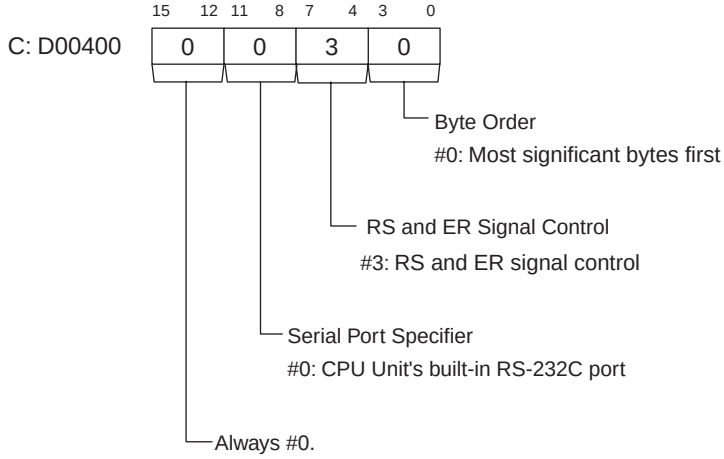


Controlling Signals



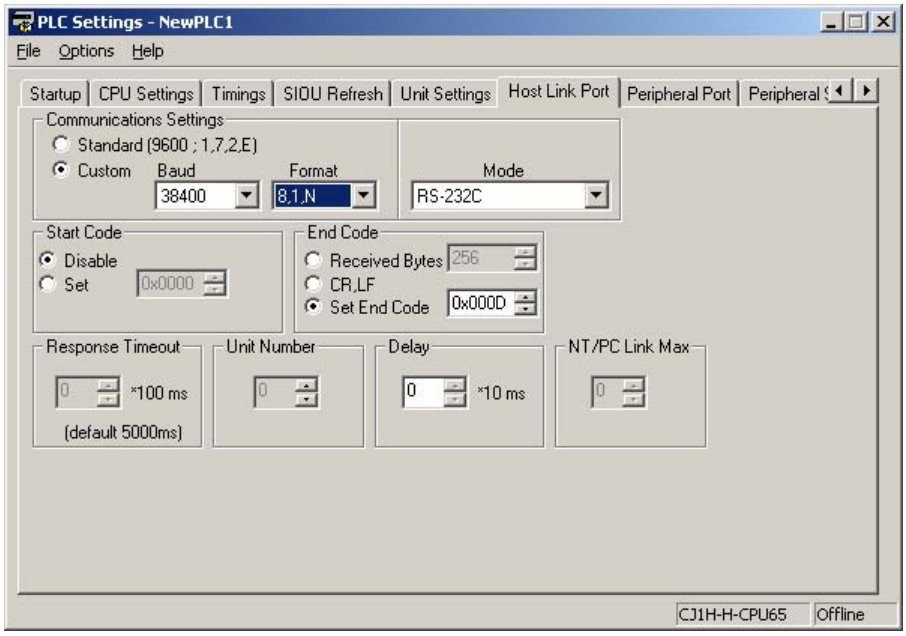
When CIO 00001 turns ON, the status of bit 15 of D00300 is output as the RS signal and the status of bit 14 is output as the ER signal.





Related PLC Setup Settings

CX-Programmer Settings for the CPU Unit's Built-in RS-232C Port



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

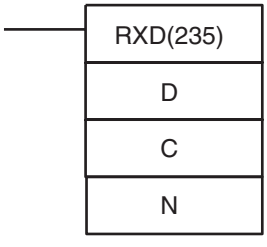
Setup Area word		Bit	Name	Settings
Port 1	Port 2			
D32002	D32012	15	No-protocol Mode Send Delay Specifier	0: Default (0 ms) 1: Use delay in bits 1 to 14.
		0 to 14	No-protocol Mode Send Delay Time	0000 to 7530 hex 0 to 300,000 ms decimal (in 10-ms units)
D32004	D32014	8 to 15	No-protocol Mode Start 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

3-24-4 RECEIVE: RXD(235)

Purpose

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).

Ladder Symbol



D: First destination word

C: Control word

N: Number of bytes to store
0000 to 0100 hex (0 to 256 decimal)

Variations

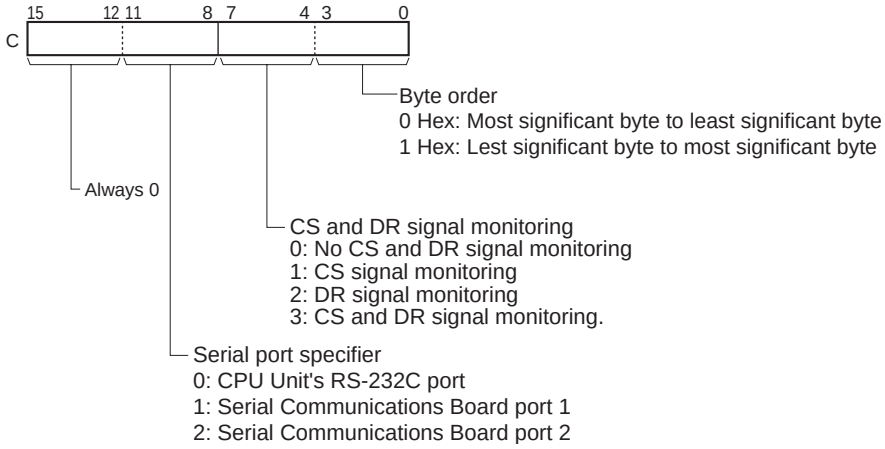
Variations	**Executed Each Cycle for ON Condition**	RXD(235)
	Executed Once for Upward Differentiation	@RXD(235)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

The contents of the control word, C, is as shown below.



Operand Specifications

Area	D	C	N
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A448 to A959	A000 to A447 A448 to A959	
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	#0000 to #0100 (binary) or &0 to &256 (decimal)
Data Registers	---	DR0 to DR15	

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

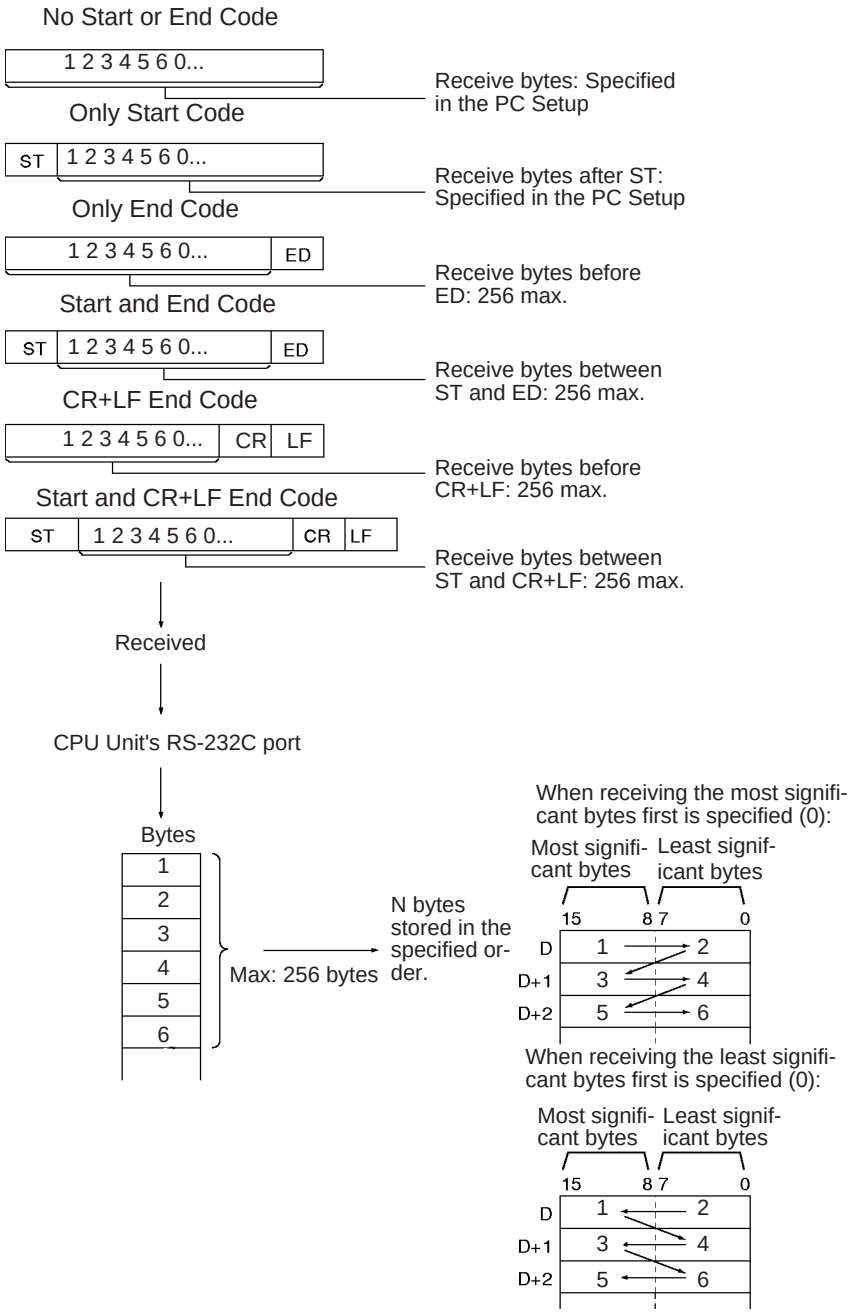
Description

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.

Data can be received only when the port's Receive Ready Flag is ON. The Receive Ready Flag is A39206 for the CPU Unit's RS-232C port, A35606 for Serial Communications Board port 1, or A35614 for Serial Communications Board port 2. Execute RXD(235) only when the corresponding Receive Ready Flag is ON.

Up to 259 bytes can be received, including the receive data (N = 256 bytes max.), the start code, and the end code.

The following diagram shows the order in which data is received and the contents of the receive frame for various settings.


No Start or End Code
1 2 3 4 5 6 0...
Receive bytes: Specified in the PC Setup
Only Start Code
ST
1 2 3 4 5 6 0...
Receive bytes after ST: Specified in the PC Setup
Only End Code
1 2 3 4 5 6 0...
ED
Receive bytes before ED: 256 max.
Start and End Code
ST
1 2 3 4 5 6 0...
ED
Receive bytes between ST and ED: 256 max.
CR+LF End Code
1 2 3 4 5 6 0...
CR
LF
Receive bytes before CR+LF: 256 max.
Start and CR+LF End Code
ST
1 2 3 4 5 6 0...
CR
LF
Receive bytes between ST and CR+LF: 256 max.
Received
CPU Unit's RS-232C port
Bytes
1
2
3
4
5
6
Max: 256 bytes
N bytes stored in the specified order.
When receiving the most significant bytes first is specified (0):
Most significant bytes
Least significant bytes
15
8
7
0
D
D+1
D+2
1
2
3
4
5
6
When receiving the least significant bytes first is specified (0):
Most significant bytes
Least significant bytes
15
8
7
0
D
D+1
D+2
1
2
3
4
5
6

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 (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. (See note.) ON if a RXD(235) was executed for a serial port on a Serial Communications Board that was being restarted. **Note** The Error (ER) Flag will turn ON immediately after another TXD(236) or RXD(235) instruction in the interrupt task. OFF in all other cases.

Precautions

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.

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).

- Start code: None or 00 to FF hex
- End code: None, CR+LF, or 00 to FF hex. If no end code is specified, the number of bytes to received is set from 00 to FF hex (1 to 256 decimal; 00 specifies 256 bytes).

The Reception Completed Flag (note 1) 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 2) 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 3) will turn ON.

If an end code is specified in the PLC Setup or the allocated DM Setup Area, the Reception Completed Flag (note 1) 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 5) and Transmission Error Flag (note 6) 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 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 1) will turn OFF (even if the Reception Overflow Flag (note 3) is ON).

With the CPU Unit's built-in RS-232C port, if the RS-232C Port Restart Bit (note 4) is turned ON, the Reception Completed Flag (note 1) will be turned OFF (even if the Reception Overflow Flag is ON), and the Reception Counter (note 2) will be cleared to 0.

Data will be stored in memory in the order specified in C.

If 0 is specified for N, the Reception Completed Flag (note 1) will be turned OFF, the Reception Counter (note 2) will be cleared to 0, and nothing will be stored in memory.

If CS signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D.

If DR signal monitoring is specified in C, the status of the DR signal will be stored in bit 15 of D.

If CS and DR signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D and the status of the DR signal will be stored in bit 14 of D.

Receive data will not be stored if CS or DR signal monitoring is specified.

If 1, 2, or 3 hex is specified for RS and ER signal control in C, RXD(235) will be executed regardless of the status of the Receive Completed Flag (note 1).

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 (A42404, non-fatal error) and the Error Flag will turn ON.

Note

1. Reception Completed Flags

Built-in RS232C port	A39206
Serial Communications Board port 1:	A35606
Serial Communications Board port 2:	A35614

2. Reception Counters

Built-in RS232C port	A393
Serial Communications Board port 1:	A357
Serial Communications Board port 2:	A358

3. Reception Overflow Flags

Built-in RS232C port	A39207
Serial Communications Board port 1:	A35607
Serial Communications Board port 2:	A35615

4. RS-232C Port Restart Bit

Built-in RS232C port	A52600

5. Overrun Error Flags

Serial Communications Board port 1:	CIO 190804
Serial Communications Board port 2:	CIO 191804

6. Transmission Error Flags

Serial Communications Board port 1:	CIO 190815
Serial Communications Board port 2:	CIO 191815

7. Inner Board Service Disabled Flag

   Serial Communications Board ports 1 and 2: A42404

An error will occur and the Error Flag will turn ON in the following cases.

- The CPU Unit's RS-232C port is specified, but no-protocol mode is not set for the port in the PLC Setup.
- One of the Serial Communications Board's serial ports is specified, but no-protocol mode is not set for the port in the allocated DM Setup Area.

- 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).
- The value of C is not within range.
- The value for N is not between 0000 and 0100 hex.
- TXD(236) or RXD(235) was being executed for the Serial Communications Board in the cyclic task, the cyclic task was interrupted, and another TXD(236) or RXD(235) instruction was executed for the Serial Communications Board in the interrupt task.
- When RXD(235) is used to read data that was received at the CPU Unit's RS-232C port, the remaining data in the port's reception buffer is not cleared, so RXD(235) can be executed repeatedly to read a block of data in parts.
  In contrast, when RXD(235) is used to read data that was received at one of the Serial Communications Board's ports (Serial Communications Board version 1.2 or later), the port's reception buffer is cleared after RXD(235) is executed. Consequently, RXD(235) can not be executed repeatedly to read a block of data in parts.
- 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.
- RXD(235) was executed for a serial port on a Serial Communications Board that was being restarted.

Related Flags and Words

The following PLC Setup settings and Auxiliary Area flag can be used as required when executing RXD(235).

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: None 1: Use end code. 2: Use CR+LF.
	0 to 7	No-protocol Mode Number of bytes of Data	00: 256 bytes 01 to FF: 1 to 255 bytes

DM Setup Area Settings for Serial Communication Board's Ports

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
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

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
D32005	D32015	12 to 15	No-protocol Mode Start Code Specifier	0: None 1: Use start code.
		8 to 11	No-protocol Mode End Code Specifier	0: None 1: Use end code. 2: Use CR+LF.

Auxiliary Area Flags for CPU Unit's RS-232C Port

Name	Address	Contents
RS-232C Port Reception Completed Flag	A39206	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	A39207	ON when more that 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.

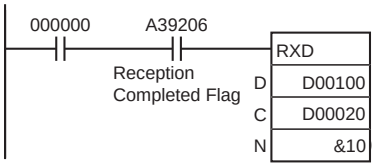
Auxiliary Area Flags for Serial Communication Board's Ports

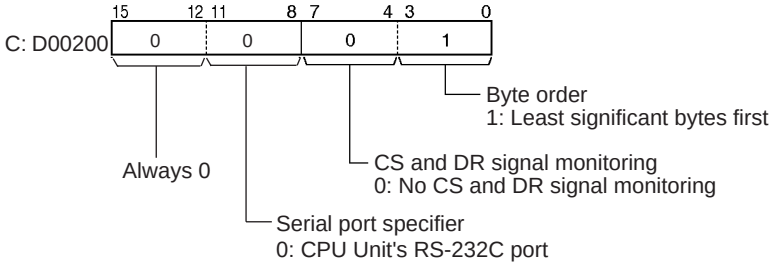
Port	Name	Address	Contents
Port 1	Reception Completed Flag	A35606	ON when no-protocol reception is completed. Number of Receive Bytes Specified: The flag will turn ON when the specified number of bytes has been received. End Code Specified: The flag will turn ON when the end code is received or when 256 bytes have been received.
	Reception Overflow Flag	A35607	ON when more that the expected number of receive bytes has been received in no-protocol mode. Number of Receive Bytes Specified: The flag will turn ON when more data is received after reception was completed but before the received data was not read from the buffer with RXD(235). End Code Specified: The flag will turn ON when 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	A35614	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	A35615	ON when more that the expected number of receive bytes has been received in no-protocol mode. Number of Receive Bytes Specified: The flag will turn ON when more data is received after reception was completed but before the received data was not read from the buffer with RXD(235). End Code Specified: The flag will turn ON when 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	A42404	ON when RXD(235) is executed for a Serial Communications Board that does not support no-protocol mode (a Board without a version number).

Examples

■ Example 1: Basic Operation

When CIO 000000 is ON in the following example, data is received from the RS-232C port and 10 bytes of data are stored starting in D00100.





This example assumes that both a start and end code have been specified in the PC Setup.

ST	31	32	33	34	41	42	43	44	45	46	ED

Stored

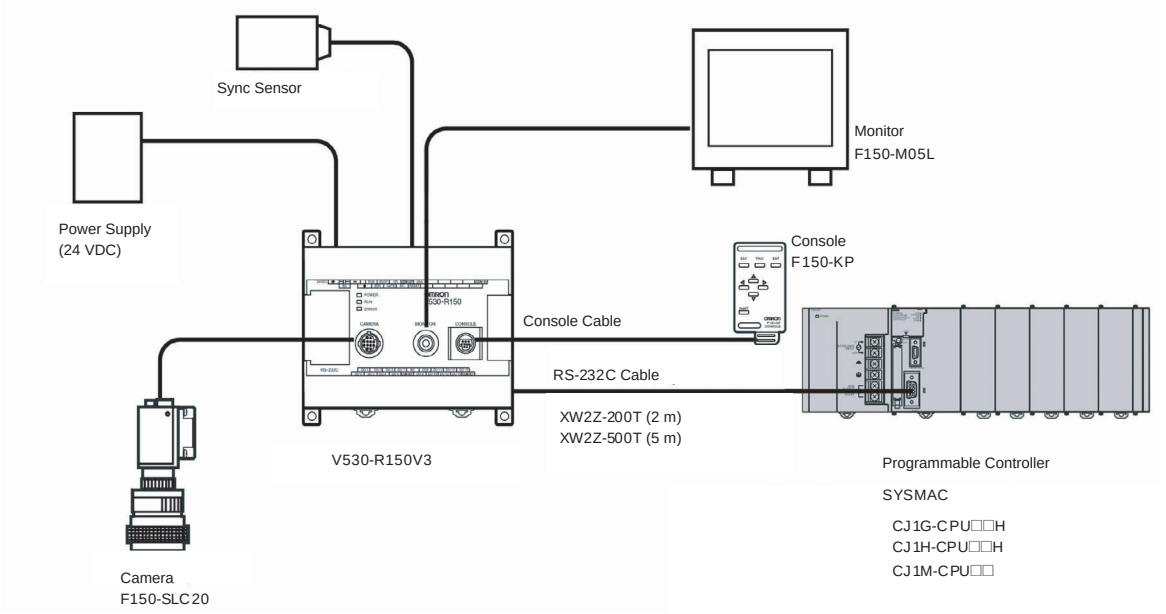
ST: Start code (e.g., 02 hex)
ED: End code (e.g., 03 hex)

	Most significant bytes		Least significant bytes	
	15 to 8		7 to 0	
D: D00100	3	2	3	1
D00101	3	4	3	3
D00102	4	2	4	1
D00103	4	4	4	3
D00104	4	6	4	5

■ Example 2: Sending Data to a Code Reader

This example shows how to received 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

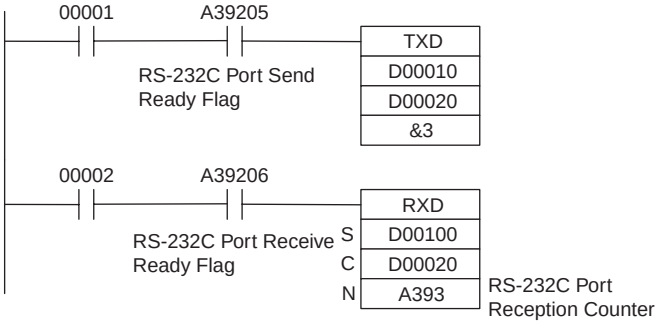
The communications settings of the Code Reader as given in the following table. These are the default settings.

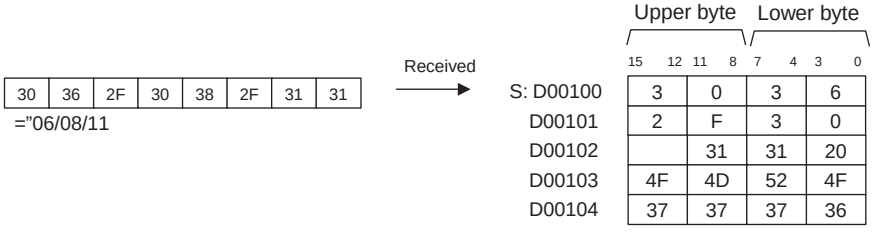
Item	Setting
Communications mode	No-protocol
Baud rate	38,400 bps
Data bit length	8 bits
Parity	None
Stop bits	1
Start code	None
End code	#000D (CR)

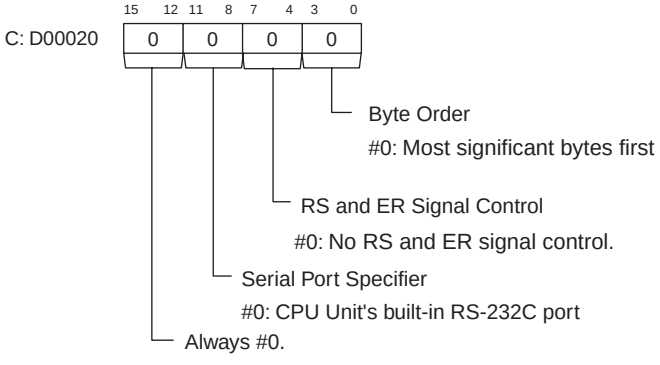
Set the PLC communications settings to the same values in the PLC Setup. Only the end code needs to be set.

Programming Example

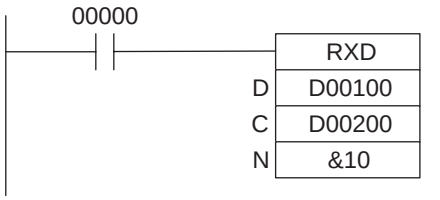
If CIO 000002 turns ON while the RS-232C Port Send Ready Flag (A39205) is ON, the number of bytes of reading results specified in the RS-232C Port Reception Counter (A393) are read from the Code Reader connected to the CPU Unit's built-in RS-232C port and stored starting from the upper byte of D00100.



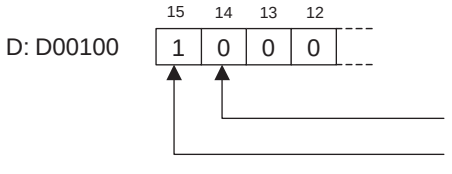


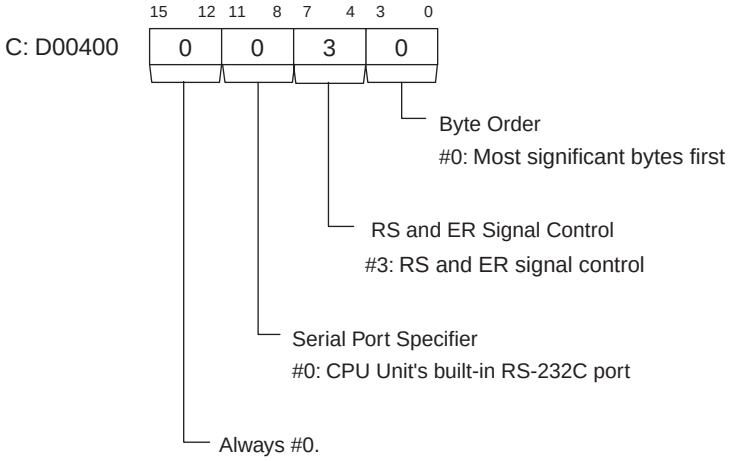


Controlling Signals



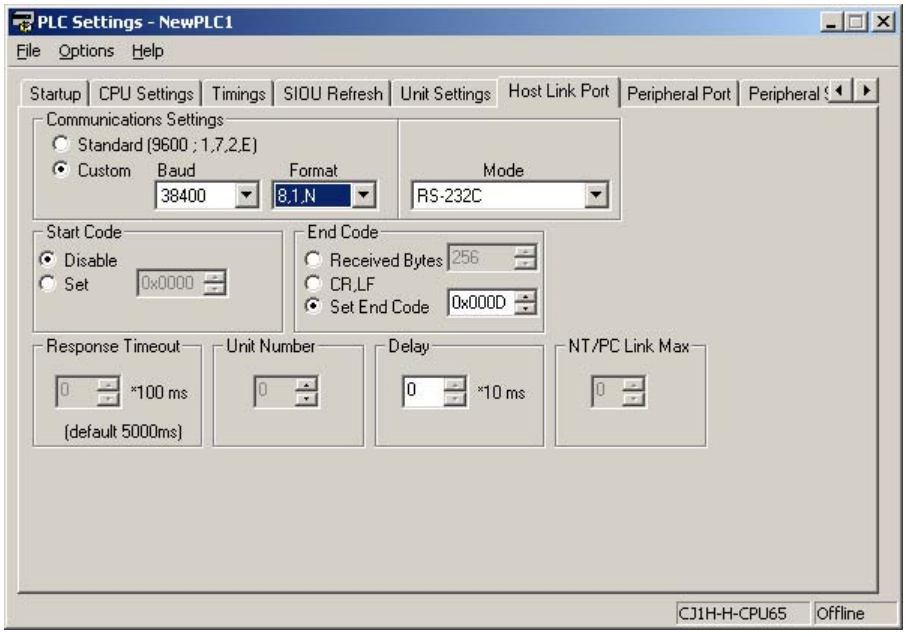
When CIO 00001 turns ON, the status of bit 15 of D00300 is output as the RS signal and the status of bit 14 is output as the ER signal.





Related PLC Setup Settings

CX-Programmer Settings for the CPU Unit's Built-in RS-232C Port



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

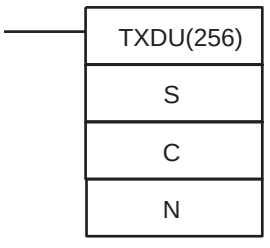
Setup Area word		Bit	Name	Settings
Port 1	Port 2			
D32002	D32012	15	No-protocol Mode Send Delay Specifier	0: Default (0 ms) 1: Use delay in bits 1 to 14.
		0 to 14	No-protocol Mode Send Delay Time	0000 to 7530 hex 0 to 300,000 ms decimal (in 10-ms units)
D32004	D32014	8 to 15	No-protocol Mode Start 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

3-24-5 TRANSMIT VIA SERIAL COMMUNICATIONS UNIT: TXDU(256)

Purpose

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).

Ladder Symbol



S: First source word

C: First control word

N: Number of bytes
0000 to 0100 hex (0 to 256)

Variations

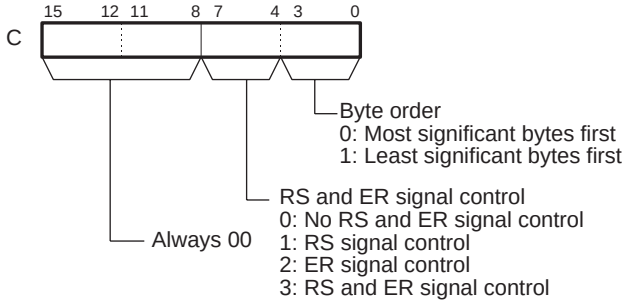
Variations	**Executed Each Cycle for ON Condition**	TXDU(256)
	Executed Once for Upward Differentiation	@TXDU(256)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

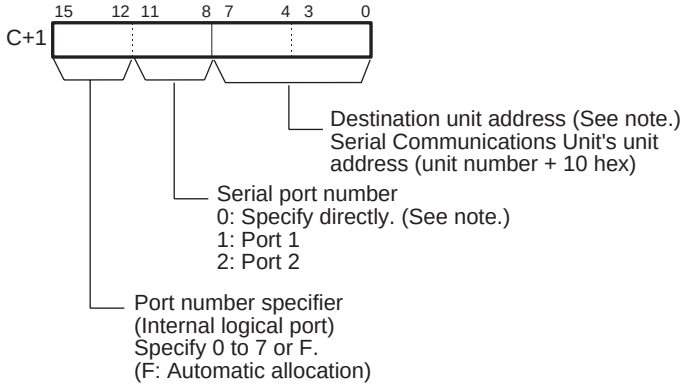
Applicable Program Areas

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

Operands

The contents of the control words, C and C+1, are as shown below.





Note The serial port's unit address can be specified directly by setting the serial port number to 0 and setting the destination unit address to the serial port's unit address. (Set the destination unit address to 80 hex + 4 × unit number for port 1 or 81 hex + 4 × unit number for port 2.)

Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W510	W000 to W511
Holding Bit Area	H000 to H511	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A958	A000 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32766	E00000 to E32767

Area	S	C	D
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	#0000 to #0100 (binary) or &0 to &256 (decimal)
Data Registers	---	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

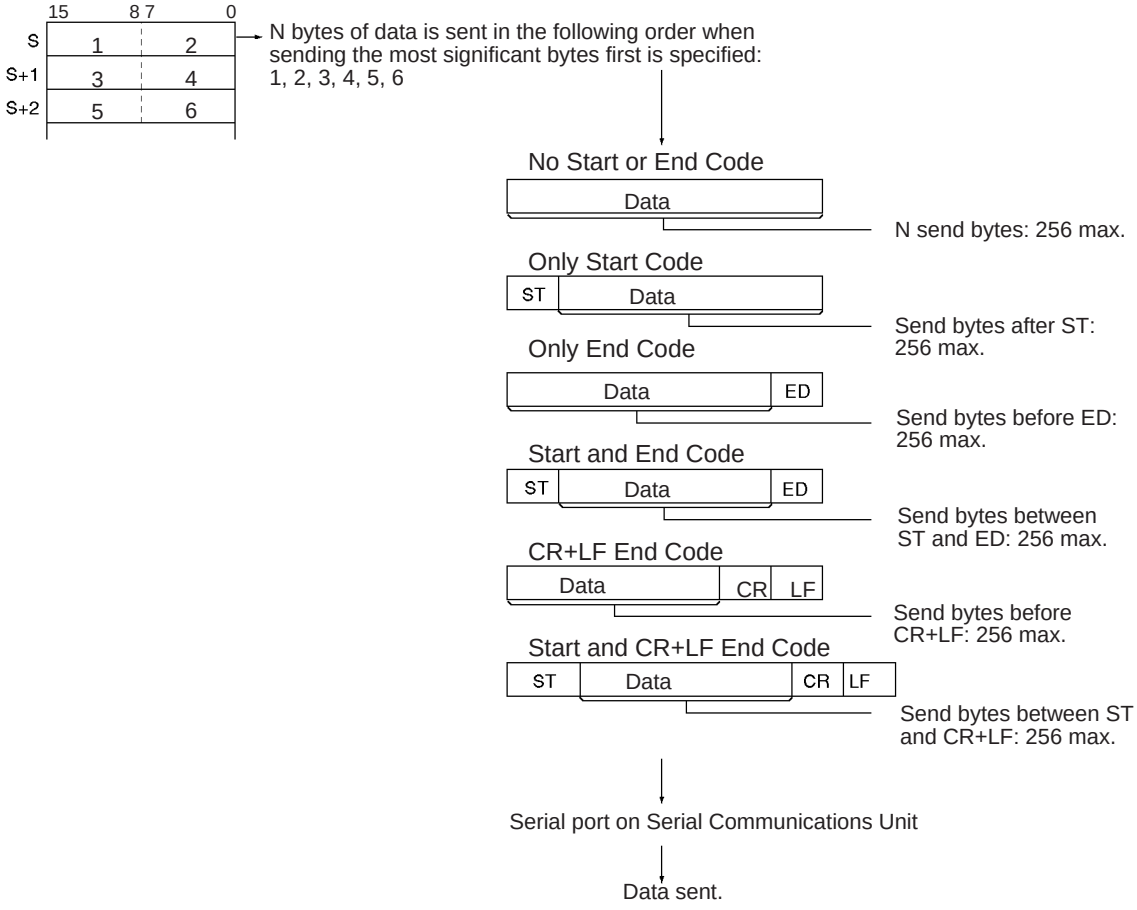
TXDU(256) reads N bytes of data from words S to S+(N÷2)–1 and outputs the raw data in no-protocol mode from the Serial Communications Unit with the unit address specified in bits 0 to 7 of C+1, through the port specified with bits 8 to 11 of C+1. The logical port number can be set to any value between 0 and 7 and is specified with bits 12 to 15 of C+1.

The start and end codes specified for no-protocol mode in the allocated DM Setup Area are added to the data before the data is output. Up to 259 bytes can be sent, including the send data (N = 256 bytes max.), the start code, and the end code.

Data can be sent only when the Communications Port Enabled Flag for the specified logical port (A20200 to A20207 for ports 0 to 7) is ON and the TXDU Instruction Executing Flag (in the allocated DM Setup Area) is OFF.

Note The logical port number can be allocated automatically by setting bits 12 to 15 of C+1 to F. For details, refer to *Automatic Allocation of Communications Ports* on page 1032.

The following diagram shows the order in which data is sent and the contents of the send frame for various start and end code settings.



Flags

Name	Label	Operation
Error Flag	ER	ON if all of the logical ports are being used or the Communications Port Enabled Flag for the specified logical port is OFF when the instruction is executed. ON if the value of C is not within range. ON if the value for N is not between 0000 and 0100 hex. OFF in all other cases.

Precautions

TXDU(256) can be used only for a Serial Communications Unit's serial port that has been set to no-protocol mode.

The following send-message frame formats can be set in the allocated DM Setup Area.

- Start code: None or 00 to FF hex.
- End code: None, CR+LF, or 00 to FF hex.

The data will be sent with any combination of start and/or end codes specified in the allocated DM Setup Area. If start and end codes are specified, the codes will be added to the send data (N). In this case, the maximum number of bytes that can be specified for N is 256 bytes.

Data is sent in the order specified in C.

Nothing will be sent if 0 is specified for N.

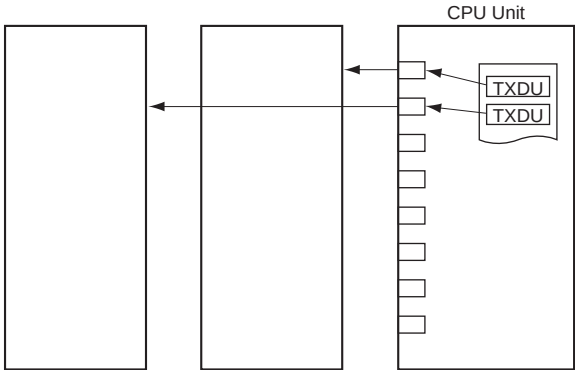
If RS signal control is specified in C, bit 15 of S will be used as the RS signal.

If ER signal control is specified in C, bit 15 of S will be used as the ER signal.

If RS and ER signal control is specified in C, bit 15 of S will be used as the RS signal and bit 14 of S will be used as the ER signal.

TXDU(256) uses a logical port (because it sends an internal FINS command) to output a send sequence command to the Serial Communications Unit (version number 1.2 or later). Since SEND(090), RECV(098), CMND(490), PMCR(260), and RXDU(255) also use logical ports 0 to 7, TXDU(256) cannot be executed for a logical port if that logical port is already being used by one of those instructions or another TXDU(256) instruction.

To ensure that TXDU(256) is not executed while the logical port is busy, program the port's Communications Port Enabled Flag (A20200 to A20207) as a normally open condition.



TXDU(256) can not be executed while the TXDU Instruction Executing Flag (bit 5 of n+9 or n+19, where n = CIO 1500 + 25 × unit number) is ON. To ensure that another TXDU(256) is not executed for the port before the first TXDU(256) is completed, program the port's TXDU Instruction Executing Flag as a normally closed condition.

An error will occur and the Error Flag will turn ON in the following cases.

- The Communications Port Enabled Flag for the specified logical port is OFF when TXDU(256) is executed.
- The value of C is not within range.
- The value for N is not between 0000 and 0100 hex.

Note Depending on the external device, it might be necessary to set a send delay when sending data with TXDU(256). It a send delay is required, set or adjust the delay time in the allocated DM Setup Area.

Related Flags and Words

The following PLC Setup settings and Auxiliary Area flag can be used as required when executing TXD(236).

DM Setup Area Settings

(m = D30000 + 100 × unit number)

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
m+2	m+12	15	No-protocol Mode Send Delay Specifier	0: Default (0 ms) 1: Use delay in bits 1 to 14.
		0 to 14	No-protocol Mode Send Delay Time	0000 to 7530 hex 0 to 300,000 ms decimal (in 10-ms units)

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
m+4	m+14	8 to 15	No-protocol Mode Start Code	00 to FF hex
		0 to 7	No-protocol Mode End Code	00 to FF hex
m+5	m+15	12 to 15	No-protocol Mode Start Code Specifier	0: None 1: Use start code.
		8 to 11	No-protocol Mode End Code Specifier	0: None 1: Use end code. 2: Use CR+LF.

Auxiliary Area

Name	Address	Description
Communications Port Enabled Flags	A20200 to A20207	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	ON when an error occurred during execution of a communications instruction. When a flag is ON, check the completion code in A203 to A210 to troubleshoot the error. OFF when execution has been finished normally. Bits 00 to 07 correspond to communications ports 0 to 7. The flag status is retained until the next communications instruction is executed. Even if an error has occurred, a flag will be reset to 0 the next time that a communications instruction is executed for that port.

Completion Codes

Code	Meaning
0205 hex	Response timeout (This error can occur when the communications mode is set to host link mode.)
0401 hex	Undefined command (This error can occur when the communications mode is set to protocol macro, NT Link, echoback test, or serial gateway mode.)
1001 hex	The command is too long.
1002 hex	The command is too short.
1003 hex	The specified number of data elements does not match the actual amount of send data.
1004 hex	The command format is incorrect.
110C hex	Other parameter error
2201 hex	Operation could not be performed during operation. (Operation disabled because Unit is busy sending.)
2202 hex	Operation could not be performed when stopped. (Operation disabled because Unit is switching protocols.)

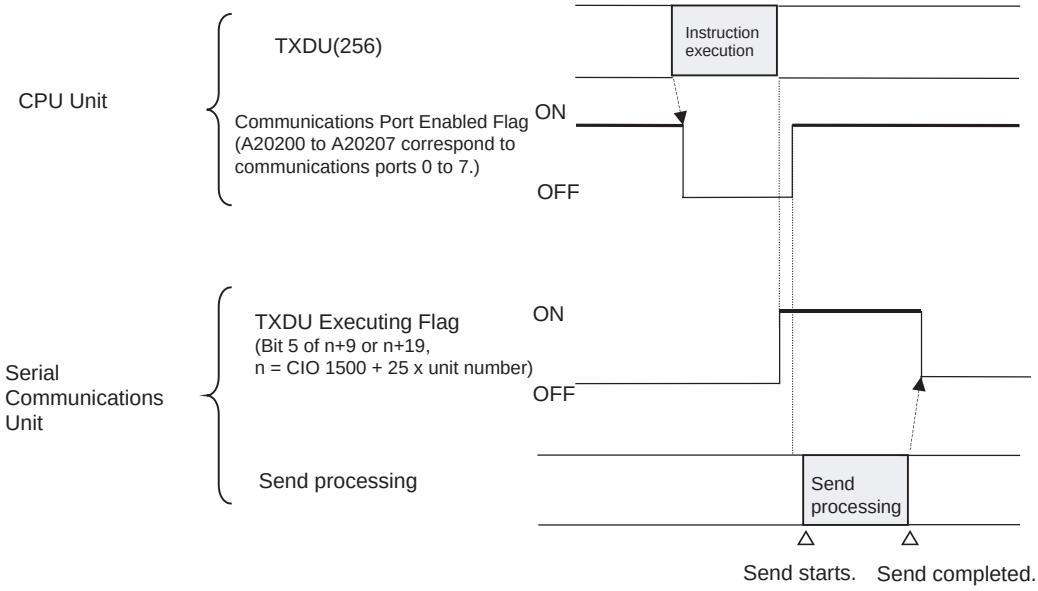
Related Flags in the CPU Bus Unit Area

(n = CIO 1500 + 25 × unit number)

Word		Bit	Name	Status
Port 1	Port 2			
n+9	n+19	05	TXDU Instruction Executing Flag	0: TXDU(256) is not being executed. 1: TXDU(256) is being executed.

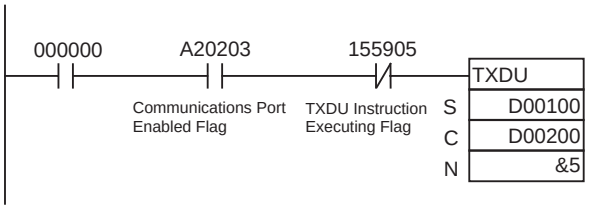
Example: Flag Operation

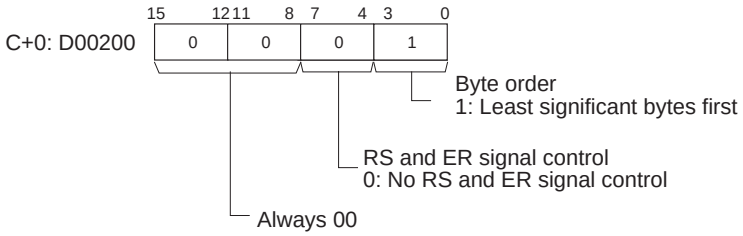
The following diagram shows the operation of the Communications Port Enabled Flag and TXDU Instruction Executing Flag.

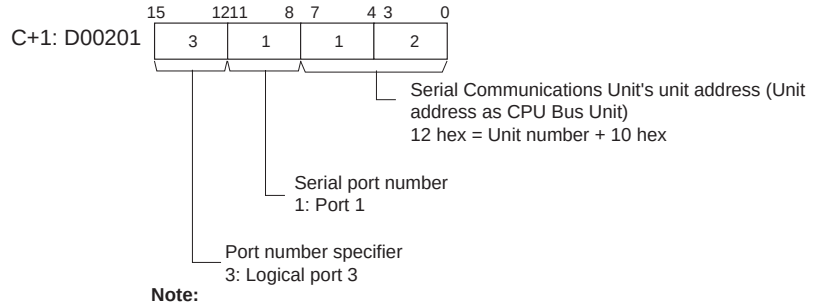


Example: Sending Data

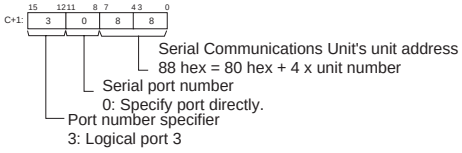
When CIO 000000 is ON, A20203 (the Communications Port Enabled Flag) is ON, and CIO 155905 (the TXDU Instruction Executing Flag for port 1) is OFF in the following example, TXDU(256) outputs data through serial port 1 of the Serial Communications Unit with unit number 2. The 5 bytes of output data are read from the DM Area beginning at the rightmost byte of D00100 and output through logical port 3 to a general-purpose device such as a printer.

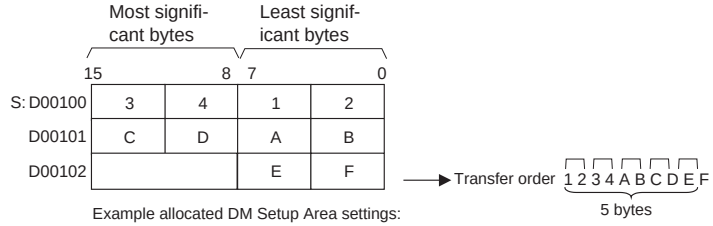






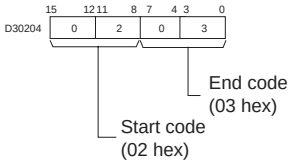
Note:
The serial port's unit address can be specified directly by setting the serial port number to 0 and setting the Serial Communications Unit's unit address to the serial port's unit address.
(Set the unit address to 80 hex + 4 x unit number for port 1 or 81 hex + 4 x unit number for port 2.)



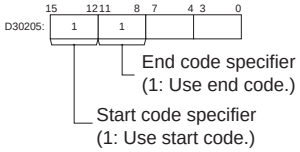


Example allocated DM Setup Area settings:

Start code and end code values



Start code and end code specifiers



In this example, a start and end code have been specified in the allocated DM Setup Area.

ST	12	34	AB	CD	EF	ED

ST: Start code (e.g., 02 hex)
ED: End code (e.g., 03 hex)

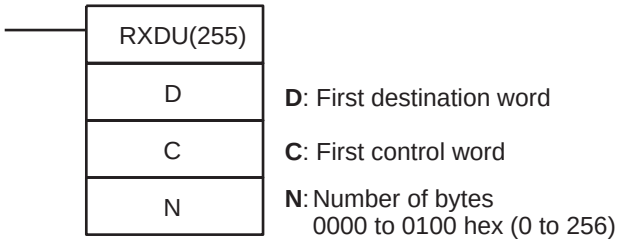
Data sent.

3-24-6 RECEIVE VIA SERIAL COMMUNICATIONS UNIT: RXDU(255)

Purpose

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).

Ladder Symbol



Variations

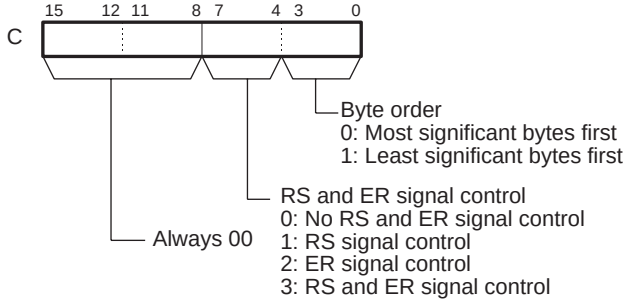
Variations		
	Executed Each Cycle for ON Condition	RXDU(255)
	Executed Once for Upward Differentiation	@RXDU(255)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

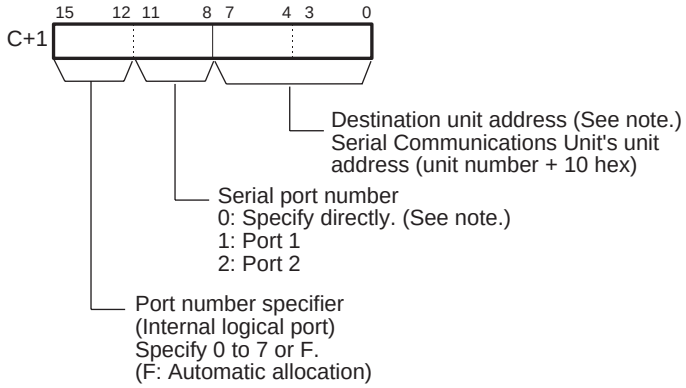
Applicable Program Areas

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

Operands

The contents of the control words, C and C+1, are as shown below.





Note The serial port's unit address can be specified directly by setting the serial port number to 0 and setting the destination unit address to the serial port's unit address. (Set the destination unit address to 80 hex + 4 × unit number for port 1 or 81 hex + 4 × unit number for port 2.)

Operand Specifications

Area	D	C	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W511	W000 to W510	W000 to W511
Holding Bit Area	H000 to H511	H000 to H510	H000 to H511
Auxiliary Bit Area	A000 to A959	A000 to A958	A000 to A959
Timer Area	T0000 to T4095	T0000 to T4094	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4094	C0000 to C4095
DM Area	D00000 to D32767	D00000 to D32766	D00000 to D32767
EM Area without bank	E00000 to E32767	E00000 to E32766	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	#0000 to #0100 (binary) or &0 to &256 (decimal)
Data Registers	---	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

RXDU(255) reads data that has been received in no-protocol mode at the Serial Communications Unit with the unit address specified in bits 0 to 7 of C+1, through the port specified with bits 8 to 11 of C+1, and stores that data starting at D. If fewer than N bytes of data have been received at the port, then only the data that has been received will be stored. The logical port number can be set to any value between 0 and 7 and is specified with bits 12 to 15 of C+1.

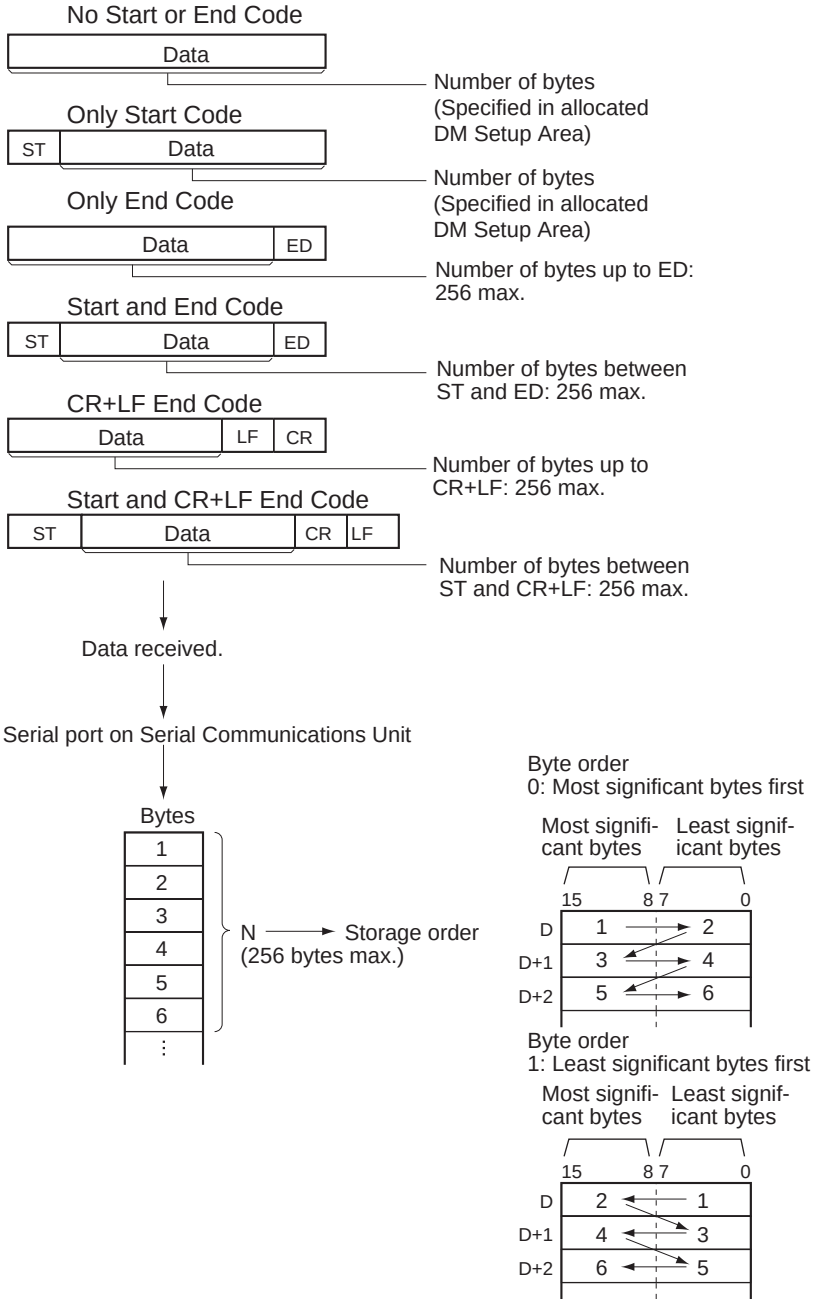
Execute RXDU(255) to read the received data from the buffer when the Reception Completed Flag (in the allocated DM Setup Area) is ON.

Up to 259 bytes can be received, including the receive data (N = 256 bytes max.), the start code, and the end code.

The following diagram shows the order in which data is received and the contents of the receive frame for various settings.

Note The logical port number can be allocated automatically by setting bits 12 to 15 of C+1 to F. For details, refer to *Automatic Allocation of Communications Ports* on page 1032.

The following diagram shows the order in which data is sent and the contents of the send frame for various start and end code settings.



Flags

Name	Label	Operation
Error Flag	ER	ON if all of the logical ports are being used or the Communications Port Enabled Flag for the specified logical port is OFF when the instruction is executed. ON if the value of C is not within range. ON if the value for N is not between 0000 and 0100 hex. OFF in all other cases.

Precautions

RXDU(255) can be used only for a Serial Communications Unit's serial port that has been set to no-protocol mode.

The following receive-message frame formats can be set in the allocated DM Setup Area.

- Start code: None or 00 to FF hex.
- End code: None, CR+LF, or 00 to FF hex. If no end code is specified, the number of bytes to be received is set from 00 to FF hex (1 to 256 decimal; 00 specifies 256 bytes).

The Reception Completed Flag (note 1) will turn ON when the number of bytes specified the allocated DM Setup Area has been received. When the Reception Completed Flag turns ON, the number of bytes in the Reception Counter (note 2) will have the same value as the number of receive bytes specified in the allocated DM Setup Area. If more bytes are received than specified, the Reception Overflow Flag (note 3) will turn ON.

If an end code is specified in the allocated DM Setup Area, the Reception Completed Flag (note 1) will turn ON when the end code is received or when 256 bytes of data have been received. If more data is received after the Reception Completed Flag (note 1) turns ON and before RXDU(255) is executed again, the Reception Overflow Flag (note 3) 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 4) and Transmission Error Flag (note 5) 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 1) will turn OFF (even if the Reception Overflow Flag (note 3) is ON), and the Reception Counter (note 2) will be cleared to 0.

Data will be stored in memory in the order specified in C.

If 0 is specified for N, the Reception Completed Flag (note 1) and Reception Overflow Flag (note 3) will be turned OFF, the Reception Counter (note 2) will be cleared to 0, and nothing will be stored in memory.

If CS signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D.

If DR signal monitoring is specified in C, the status of the DR signal will be stored in bit 15 of D.

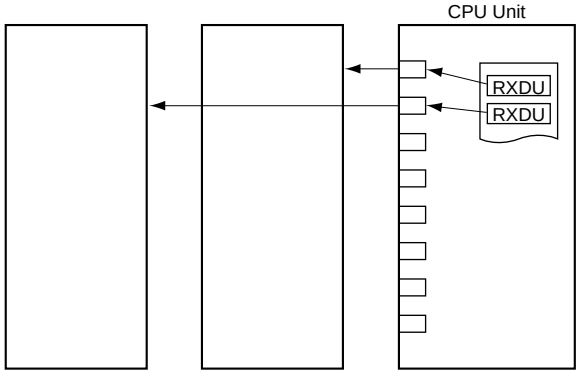
If CS and DR signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D and the status of the DR signal will be stored in bit 14 of D.

Receive data will not be stored if CS or DR signal monitoring is specified.

If 1, 2, or 3 hex is specified for RS and DR signal control in C, RXDU(255) will be executed regardless of the status of the Receive Completed Flag (note 1).

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 (A20200 to A20207) as a normally open condition.



RXDU(255) can not be executed while the Reception Completed Flag (bit 6 of n+9 or n+19, where n = CIO 1500 + 25 × unit number) is ON. Program the Reception Completed Flag as a normally open condition of RXDU(255).

An error will occur and the Error Flag will turn ON in the following cases.

- The Communications Port Enabled Flag for the specified logical port is OFF when RXDU(255) is executed.
- The value of C is not within range.
- The value for N is not between 0000 and 0100 hex.

Note

1. Reception Completed Flags (n = CIO 1500 + 25 × unit number)
   Port 1: Bit 6 of n+9
   Port 2: Bit 6 of n+19
2. Reception Counters (n = CIO 1500 + 25 × unit number)
   Port 1: n+10
   Port 2: n+20
3. Reception Overflow Flags (n = CIO 1500 + 25 × unit number)
   Port 1: Bit 7 of n+9
   Port 2: Bit 7 of n+19
4. Overrun Error Flags (n = CIO 1500 + 25 × unit number)
   Port 1: Bit 4 of n+8
   Port 2: Bit 4 of n+18
5. Transmission Error Flags (n = CIO 1500 + 25 × unit number)
   Port 1: Bit 15 of n+8
   Port 2: Bit 15 of n+18
6. 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.
7. 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.

Related Flags and Words

The following words are related to RXDU(255) operation.

DM Setup Area Settings

(m = D30000 + 100 × unit number)

Setup Area word		Bit	Name	Settings
Port 1	Port 2			
m+4	m+14	8 to 15	No-protocol Mode Start Code	00 to FF hex
		0 to 7	No-protocol Mode End Code	00 to FF hex
m+5	m+15	12 to 15	No-protocol Mode Start Code Specifier	0: None 1: Use start code.
		8 to 11	No-protocol Mode End Code Specifier	0: None 1: Use end code. 2: Use CR+LF.

Auxiliary Area

Name	Address	Description
Communications Port Enabled Flags	A20200 to A20207	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	ON when an error occurred during execution of a communications instruction. When a flag is ON, check the completion code in A203 to A210 to troubleshoot the error. OFF when execution has been finished normally. Bits 00 to 07 correspond to communications ports 0 to 7. The flag status is retained until the next communications instruction is executed. Even if an error has occurred, a flag will be reset to 0 the next time that a communications instruction is executed for that port.

Completion Codes

Code	Meaning
0205 hex	Response timeout (This error can occur when the communications mode is set to host link mode.)
0401 hex	Undefined command (This error can occur when the communications mode is set to protocol macro, NT Link, echoback test, or serial gateway mode.)
1001 hex	The command is too long.
1002 hex	The command is too short.
1004 hex	The command format is incorrect.
110C hex	Other parameter error

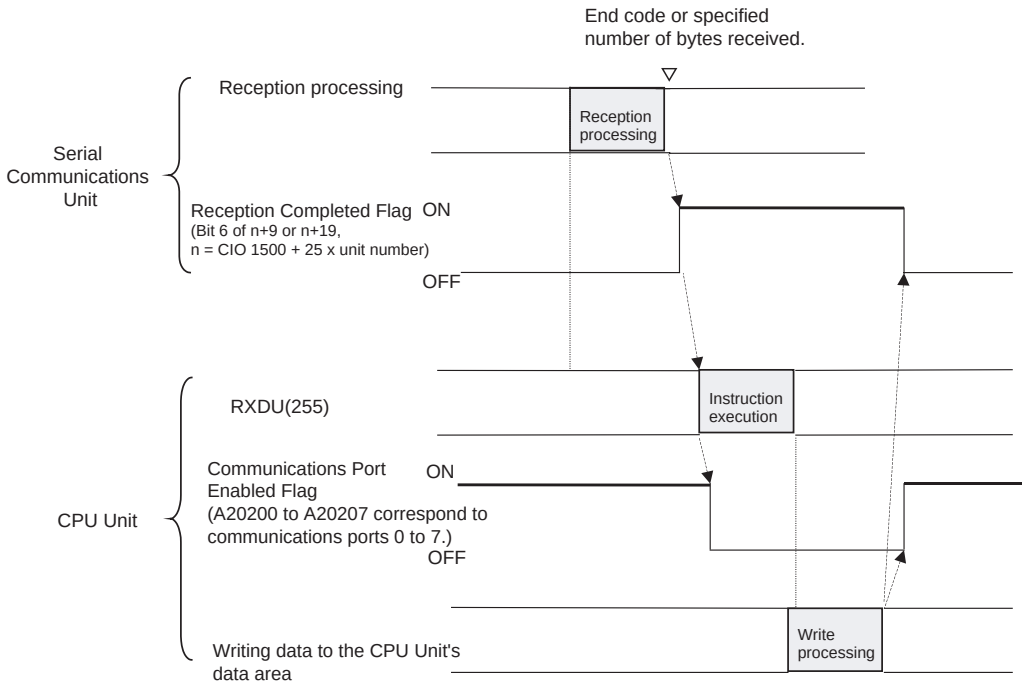
Code	Meaning
2201 hex	Operation could not be performed during operation. (Operation disabled because Unit is busy sending.)
2202 hex	Operation could not be performed when stopped. (Operation disabled because Unit is switching protocols.)

Related Flags in the CPU Bus Unit Area

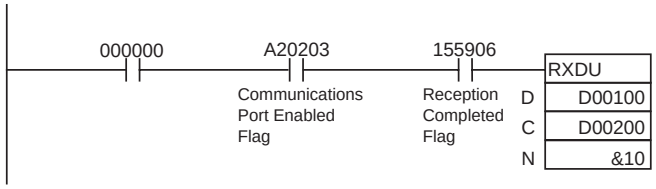
(n = CIO 1500 + 25 × unit number)

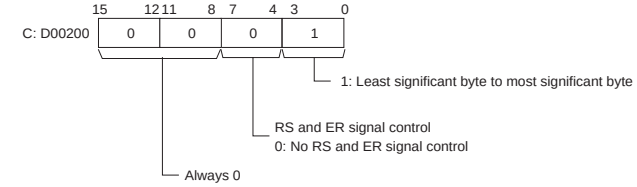
Word		Bit	Function
Port 1	Port 2		
n+8	n+18	04	Overrun Error Flag 1: The reception buffer contained more than 259 bytes of data before RXDU(255) was executed. Note: Once this error flag goes ON, it can be turned OFF only by turning the power OFF and then ON again or restarting the Board.
n+9	n+19	06	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	05	Reception Counter Indicates the number of bytes received in hexadecimal, between 0000 and 0100 hex (0 to 256 decimal).

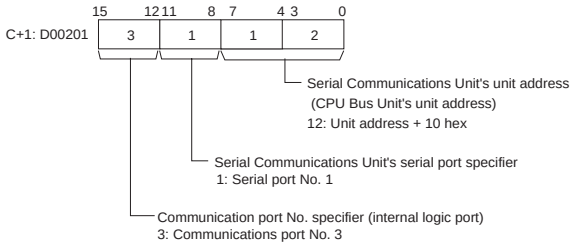
Example: Flag Operation The following diagram shows the operation of RXDU(255) and related flags.



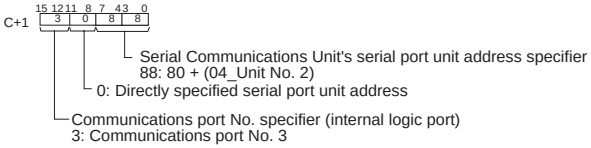
Example: Receiving Data When CIO 000000 is ON, A20203 (the Communications Port Enabled Flag) is ON, and CIO 155906 (the Reception Completed Flag for port 1) is OFF in the following example, RXDU(255) reads the data received through serial port 1 of the Serial Communications Unit with unit number 2. (Logical communications port number 3 is used to receive the data from a general-purpose device such as a bar-code reader.) The 10 bytes of received data are written to the DM Area beginning at the rightmost byte of D00100.

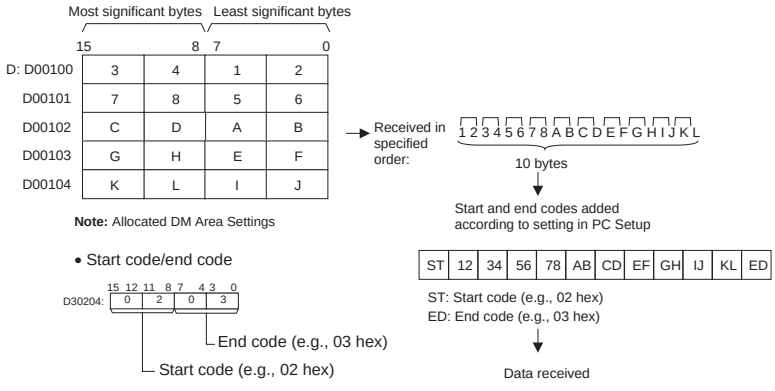




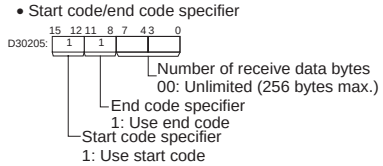


Note: The Serial Communications Unit's serial port unit address can also be directly specified in C+1.





Note: Allocated DM Area Settings

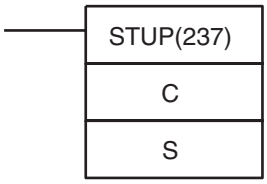


3-24-7 CHANGE SERIAL PORT SETUP: STUP(237)

Purpose

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(237) thus enables the protocol mode to be changed during PLC operation.

Ladder Symbol



C: Control word (port)

S: First source word

Variations

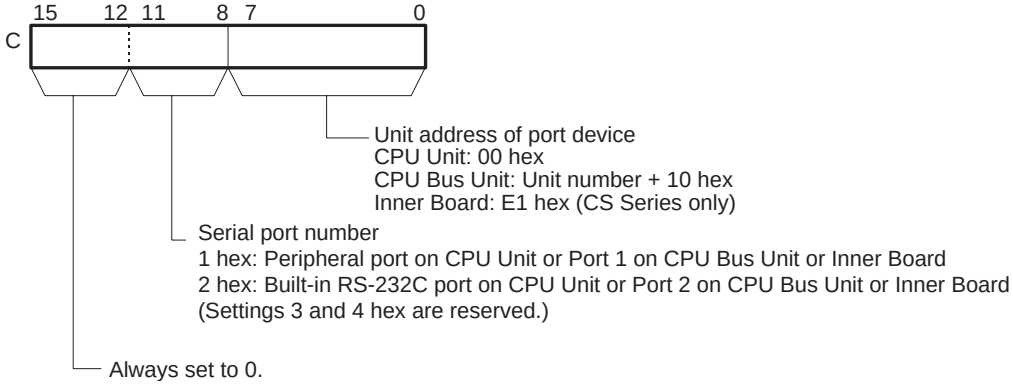
Variations		
	Executed Each Cycle for ON Condition	STUP(237)
	Executed Once for Upward Differentiation	@STUP(237)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

The contents of the control word, C, are as shown below.



Operand Specifications

Area	C	S
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6134
Work Area	W000 to W511	W000 to W502
Holding Bit Area	H000 to H511	H000 to H502
Auxiliary Bit Area	A000 to A438 A448 to A959	A000 to A438 A448 to A950
Timer Area	T0000 to T4095	T0000 to T4086
Counter Area	C0000 to C4095	C0000 to C4086
DM Area	D00000 to D32767	D00000 to D32758
EM Area without bank	E00000 to E32767	E00000 to E32758
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32758 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	

Area	C	S
Constants	Specified values only	#0000
Data Registers	DR0 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

STUP(237) writes 10 words of data from S to S+9 to the communications setup area of the Unit with the specified unit address, as shown in the following table. When the constant #0000 is designated to S, the communications settings of the corresponding port will be set to default.

Unit address	Unit	Port No.	Serial port	Serial port communications setup area
00 hex	CPU Unit	1 hex	Port 1	Communications parameters for the peripheral port in the PLC Setup
		2 hex	Port 2	Communications parameters for the RS-232C port in the PLC Setup
Unit No. + 10 hex	Serial Communications Unit (CPU Bus Unit)	1 hex	Port 1	10 words starting from D30000 + 100 x Unit No.
		2 hex	Port 2	10 words starting from D30000 + 100 x Unit No. + 10
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

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	PLC Setup settings in Programming Console addresses +160 to +169
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

When STUP(237) is executed, the corresponding Port Parameters Changing Flag (A61901, A61902, or A619 to A636) will turn ON. The flag will remain ON until changing the parameters has been completed.

Use STUP(237) to change communications parameter for a port during operation based on specified conditions. For example, STUP(237) can be used to change to Host Link communications for monitoring and programming from a host computer when specified conditions are meet while execution a communications sequence for a modem connection.

Differences between CPU Units

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
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	If the PLC is turned OFF and then ON again, the communications parameters set with STUP(237) are retained.

Flags

Name	Label	Operation
Error Flag	ER	ON if the values in C are not within range. ON if STUP(237) is executed for a port whose Communications Parameter Changing Flag is already ON. ON if STUP(237) is executed in an interrupt task. OFF in all other cases.

Precautions

Communications parameters consist of the protocol mode, baud rate, data format (protocol macro transmission method and protocol macro maximum communications data length), and other parameters. Refer to *CS/CJ-series Programmable Controllers Operation Manual* (W339) or *CS/CJ-series Serial Communications Boards and Serial Communications Unit Operation Manual* (W336) for the serial port that is to be set for details.

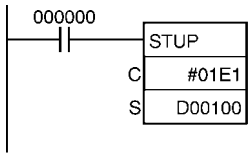
Related Flags and Words

The following flags can be used as required when executing STUP(237). These flags are in the Auxiliary Area.

Name	Address	Contents
Peripheral Port Parameters Changing Flag	A61901	ON when the communications parameters are being changed for the peripheral port.
RS-232C Port Parameters Changing Flag	A61902	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)	A63601 to A63604	ON when the communications parameters are being changed for a port on the Serial Communications Board.

Examples

When CIO 000000 turns ON in the following example, the communications parameters for serial port 1 of the Serial Communications Board (Inner Board) are changes to the settings contained in the 10 words from D00100 to D00109. In this example, the setting are changed the protocol mode to the protocol macro mode.



S: D00100	0	6	0	0	Port setting: Default, Protocol mode: 6 hex (protocol macro).
S+1: D00101	0	0	0	0	Baud rate: Default (9,600 bps)
S+2: D00102					
to		to			
S+9: D00109					

Transferred

DM words allocated to the communications setup of the Serial Communications Board.

D32000	0	6	0	0
D32001	0	0	0	0
D32002				
to		to		
D32009				

3-25 Network Instructions

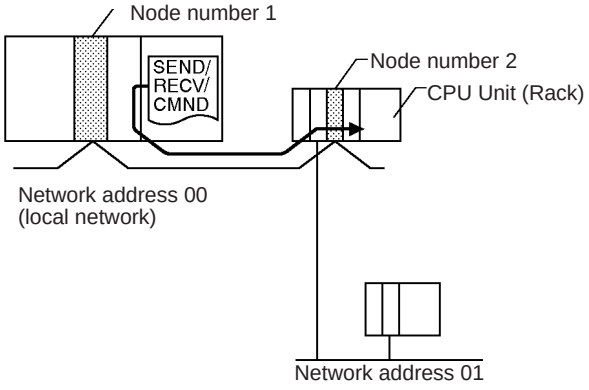
3-25-1 About SYSMAC NET Link/SYSMAC LINK Operations

The network instructions can be divided into two types, SEND(090)/RECV(098) and CMND(490). 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.

Instruction	Message content	Operation
SEND(090)/ RECV(098)	Commands to transmit/ receive data (FINS command)	CPU Unit: SEND(090) or RECV(098). Data transmission →. Data reception ←. Other device: CPU Unit, CS1 CPU Bus Unit or computer
CMND(490)	Arbitrary commands (FINS command)	CPU Unit: CMND(490). Command sent →. Response returned ←. Other device: CPU Unit, CS1 CPU Bus Unit, or computer

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* 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.



1,2,3... 1. Network address:
Address of the network (local network = 00)

2. Node number
Logical address in the network

3. Unit number
Unit number of the destination Unit

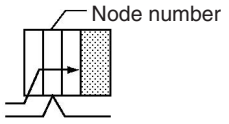
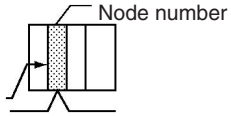
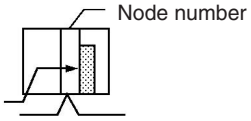
   a) CPU Unit: 00

   b) CPU Bus Unit: Unit number +10 hexadecimal

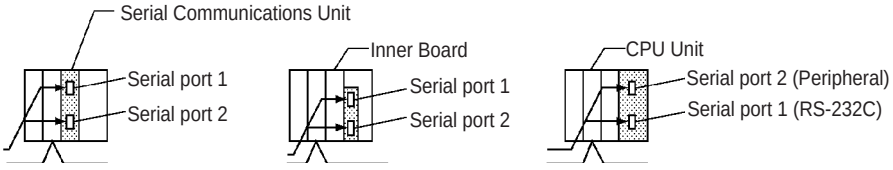
   c) Special I/O Unit (except for C200H-series Special I/O Units):
   Unit number +20 hexadecimal

d) Inner Board (CS Series only):
E1 hexadecimal

e) Computer: 01

Unit number (hexadecimal)	Destination device
00	
Unit number +10	
E1	
01	

Note It is also possible to directly specify a serial port (unit address) within the destination device.



Serial Port Unit Addresses:

- Serial Communications Unit ports

  Port 1: 80 hex + 4 × unit number

Unit number	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hexadecimal	80	84	88	8C	90	94	98	9C	A0	A4	A8	AC	B0	B4	B8	BC
Decimal	128	132	136	140	144	148	152	156	160	164	168	172	176	180	184	188

  Port 2: 81 hex + 4 × unit number

Unit number	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hexadecimal	81	85	89	8D	91	95	99	9D	A1	A5	A9	AD	B1	B5	B9	BD
Decimal	129	133	137	141	145	149	153	157	161	165	169	173	177	181	185	189

- 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)

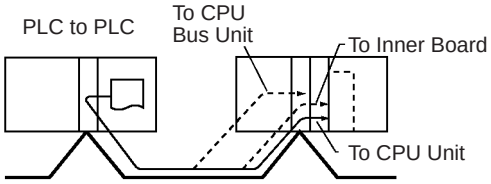
Network Communications Patterns

The following examples show three types of network communications: communications from a PLC to other devices in a network, communications from a

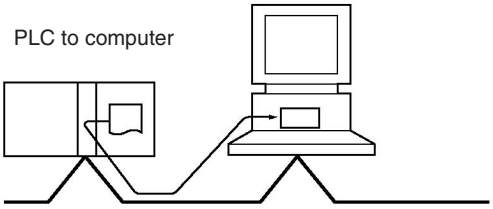
PLC to serial ports on other devices in a network, and communications to a host computer connected by a Host Link.

Communications to Another Device in the Network

The following example shows communications from a PLC to devices in another PLC (the CPU Unit, CPU Bus Unit, or Inner Board). For more details, refer to the Operation Manual for the network (Controller Link or Ethernet) being used.

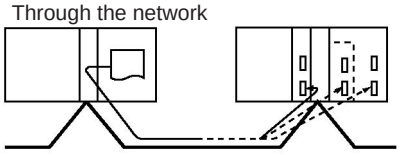


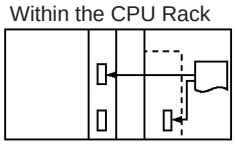
This example shows communications from a PLC to a personal computer.



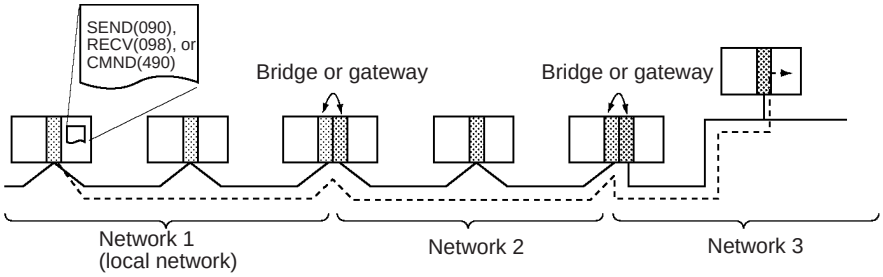
Communications to a Serial Port in the Network

These examples show communications from a PLC to serial ports in devices in the network. The first shows communications to serial ports in devices in another PLC (the CPU Unit, CPU Bus Unit, or Inner Board) and the second shows communications to a serial port within the CPU Rack itself.





Note Communications can span up to 8 network levels, including the local network. (The local network is the network where the communications originate.)



In order to communicate through the network, it is necessary to register a routing table in each PLC's CPU Unit which indicates the route by which data

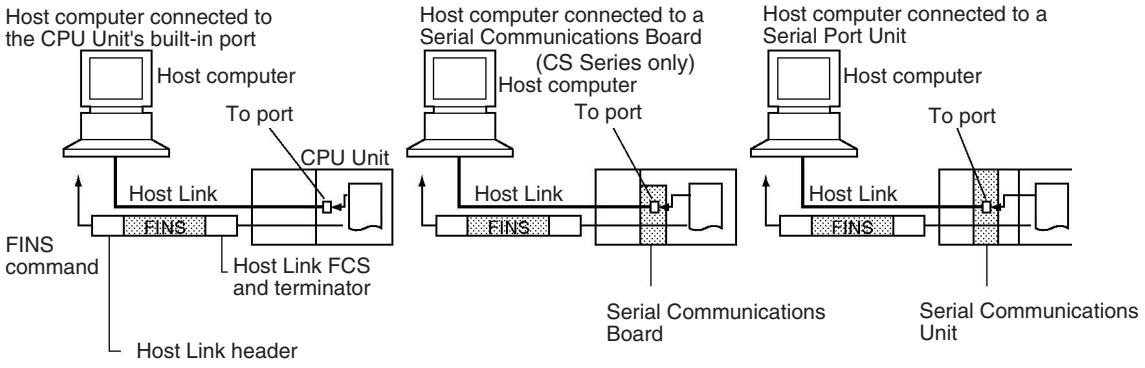
will be transferred to the desired node. Each routing table is made up of a local network table and a relay network table.

1,2,3... 1. Local network table
This table shows the unit numbers and network addresses of the nodes connected to the local PLC.

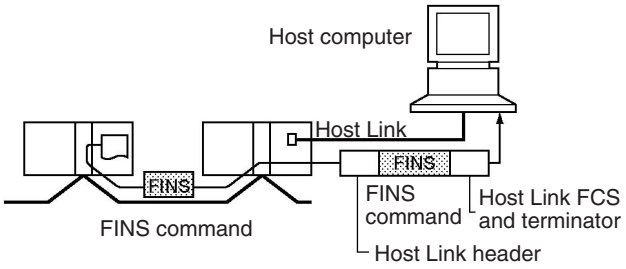
2. Relay network table
This table shows the node numbers and network addresses of the first relay nodes to destination networks that are not connected to the local PLC.

Communications to a Host Computer (Host Link)

By issuing a SEND(090), RECV(098), or CMND(490) instruction to a serial port set to Host Link mode, the necessary Host Link header and terminator will be attached to the FINS command and the command will be sent to the host computer.



Note Host Link communications can be sent through the network. In this case, the FINS command travels through the network normally. When the command reaches the Host Link system, the necessary Host Link header and terminator are attached to the FINS command and the command is sent to the host computer.

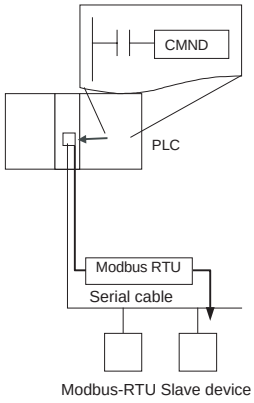


Serial Gateway Communications to a Component or Host Link Slave

It is possible to send FINS commands (or send/receive data) to a component or Host Link Slave connected to the PLC through its serial port with the serial gateway function.

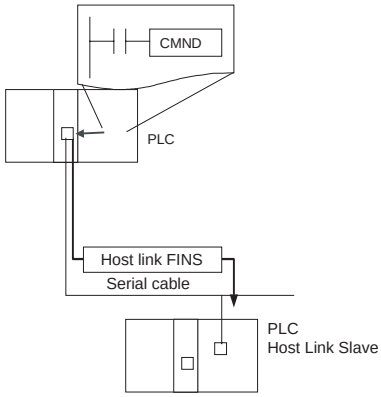
- Sending to a Component

  When a CMND(490) instruction is executed to a serial port that supports the serial gateway function, the serial gateway function converts the command to a CompoWay/F, Modbus-RTU, or Modbus-ASCII command.



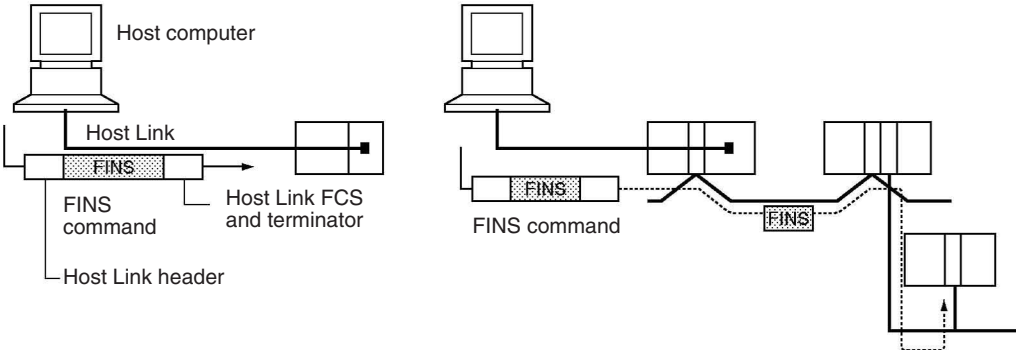
- Sending to a PLC operating as a Host Link Slave

  When a CMND(490), SEND(090), or RECV(098) instruction is executed to a serial port that supports the serial gateway function, the serial gateway function can send any FINS command or send/receive data.



Communications from a Host Computer (Host Link)

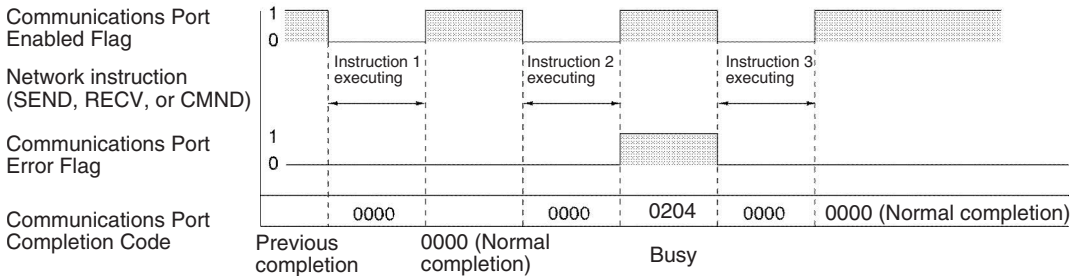
It is possible to send FINS commands from a host computer to the PLC to which it is connected as well as other devices in the network (CPU Units, Special I/O Units, computers, etc.). In this case, the necessary Host Link header and terminator must be attached to the FINS command when it is sent.



Communications Flags

The operation of the communications flags is outlined below.

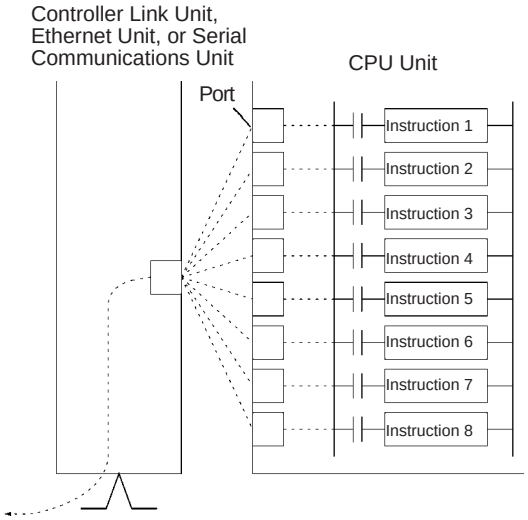
- The Communications Port Enabled Flag is reset to 0 when communications are in progress and set to 1 when communications are completed (normally or not).
- The status of the Communications Port Error Flag is maintained until the next time that data is transmitted or received.
- The Communications Port Error Flag will be reset to 0 the next time that data is transmitted or received, even if there was an error in the previous operation.



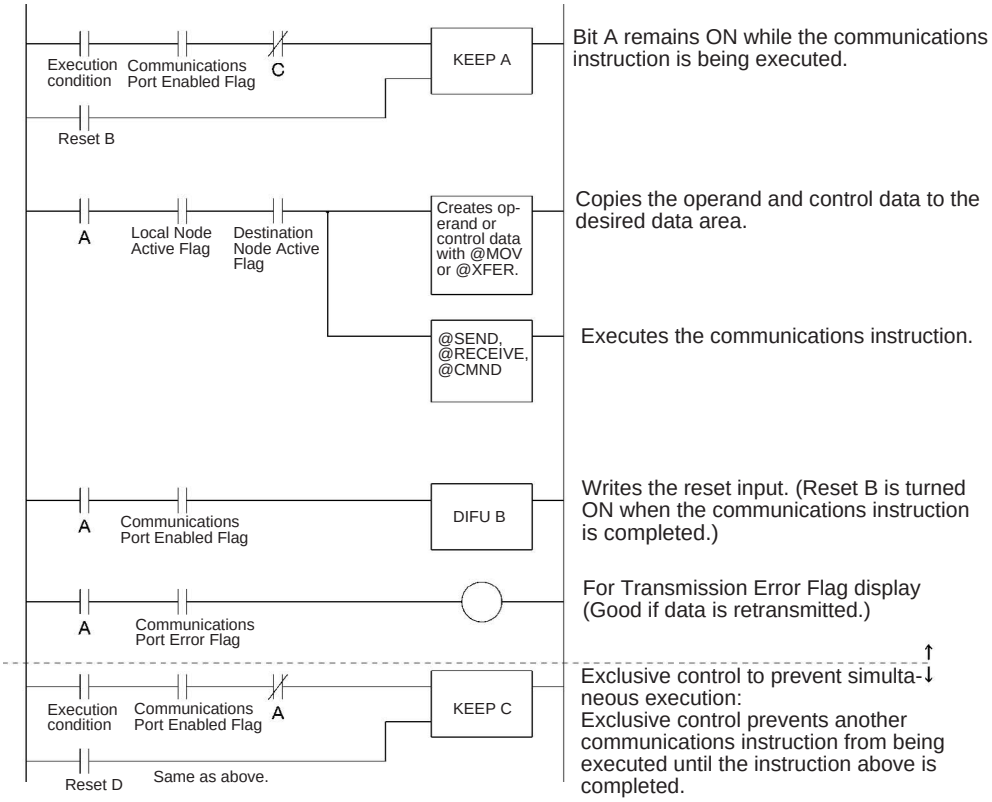
About Communications Port Numbers

There are 8 logical communications ports provided, so 8 communications instructions can be executed simultaneously. Only one instruction can be executed at a time for each communications port. Exclusive control must be used when more than 8 instructions are executed.

These 8 communications port numbers are shared by the network instructions (SEND(090), RECV(098), and CMND(490)), the serial communications instructions (TXDU(256) and RXDU(255)), and the PROTOCOL MACRO instruction (PMCR(260)). Be sure not to specify the same port number on two instructions at the same time.



The following diagram shows an example of exclusive control.



Automatic Allocation of Communications Ports

■ Overview

The following instructions all use one communications port (logical port) between ports 0 to 7.

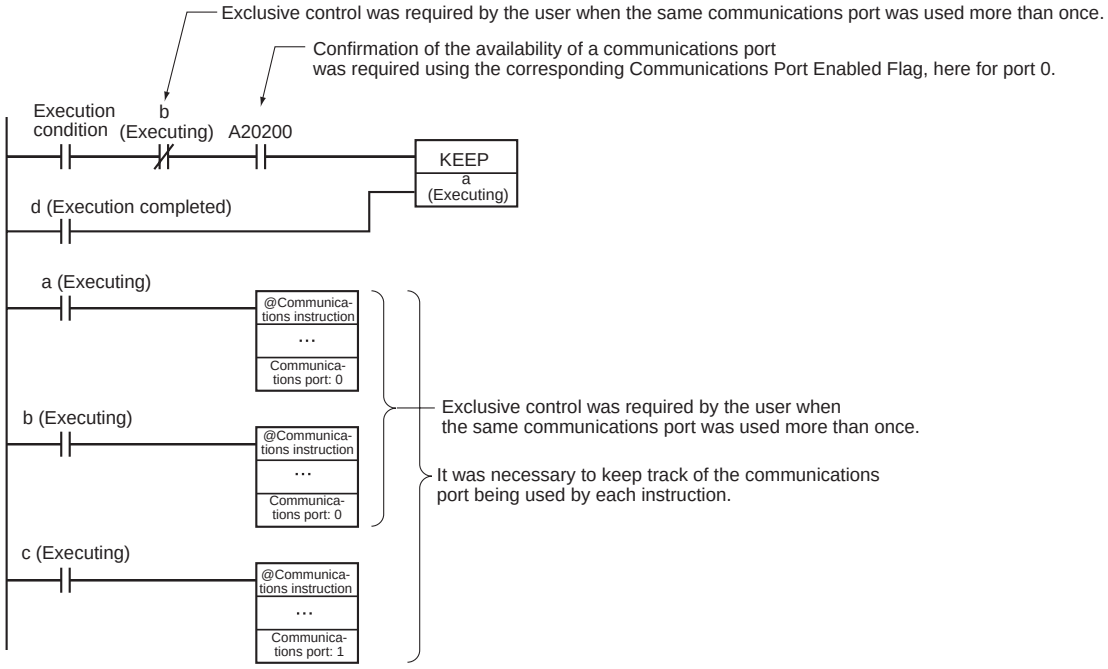
- Network Communications Instructions: SEND(090), RECV(098), and CMND(490)
- Serial Communications Instructions: PMCR(260), TXDU(256), and RXDU(255)

In this section, all of the above instructions are referred to as Communications Instructions.

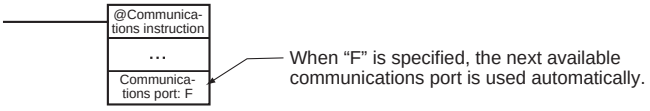
Each communications port can be used by only one instruction at a time. The following steps were previously necessary to use the communications ports.

- When programming, it was necessary to keep track of the communications ports that were being used to designate them in operands.
- In the ladder program, it was necessary to confirm the availability of communications ports before using them.

Example of Previous Programming Requirements



Now, for CS1-H, CJ1-H, CJ1M, and CS1D 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.



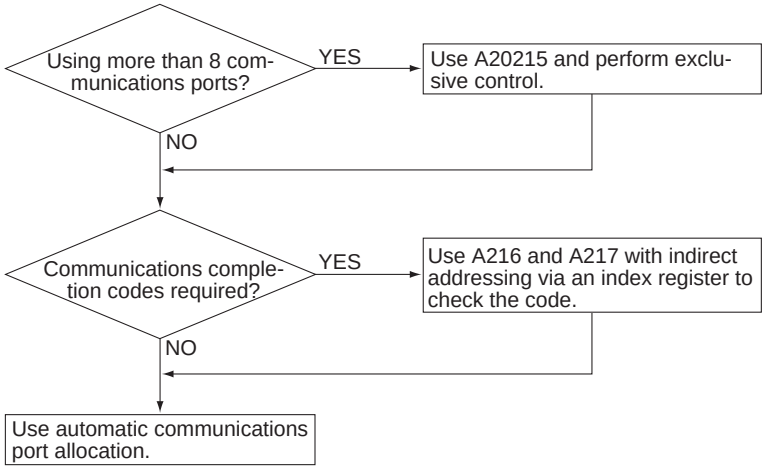
This saves the programmer from having to keep track of communications ports while programming. The differences between assigning specific port numbers and automatically allocating port numbers are given in the following table.

Item	Specific number assignments	Automatic allocation
Specification of the communications port number in the control data	0 to 7	F
Exclusive control	Required.	Not required unless more than 8 communications ports are required at the same time.
Flag applications	LD or LD NOT used with flag corresponding to the specified communications port.	TST(350) or TSTN(351) used with A218 (Used Communications Port Number).
Network communications completion codes	Completion code for communications port specified by user is accessed.	Completion codes are accessed by using the I/O memory address stored in A216 and A217 (Network Communications Completion Code Storage Address) and index register indirect addressing.

■ Auxiliary Area Bits and Words Used when Automatically Allocating Communications Ports

Address	Bits	Name	Description
A202	15	Network Communications Port Allocation Enabled Flag	ON when there is a communications port available for automatic allocation. This flag can be used to confirm if all eight communications ports have already been allocated before executing communications instructions.
A214	00 to 07	First Cycle Flags after Network Communications Finished	Each flag will turn ON for just one cycle after communications have been completed. Bits 00 to 07 correspond to ports 0 to 7. Use the Used Communications Port Number stored in A218 to determine which flag to access. **Note:** These flags are not effective until the next cycle after the communications instruction is executed. Delay accessing them for at least one cycle.
	08 to 15	Do not use.	
A215	00 to 07	First Cycle Flags after Network Communications Error	Each flag will turn ON for just one cycle after a communications error occurs. Bits 00 to 07 correspond to ports 0 to 7. Use the Used Communications Port Number stored in A218 to determine which flag to access. **Note:** These flags are not effective until the next cycle after the communications instruction is executed. Delay accessing them for at least one cycle.
	08 to 15	Do not use.	
A216 and A217	---	Network Communications Completion Code Storage Address	The completion code for a communications instruction is automatically stored at the address with the I/O memory address given in these words. Place this address into an index register and use indirect addressing through the index register to reach the communications completion code.
A218	---	Used Communications Port Number	When a communications instruction is executed, the number of the communications port that was used is stored in this word. Values 0000 to 0007 hex correspond to communications ports 0 to 7.

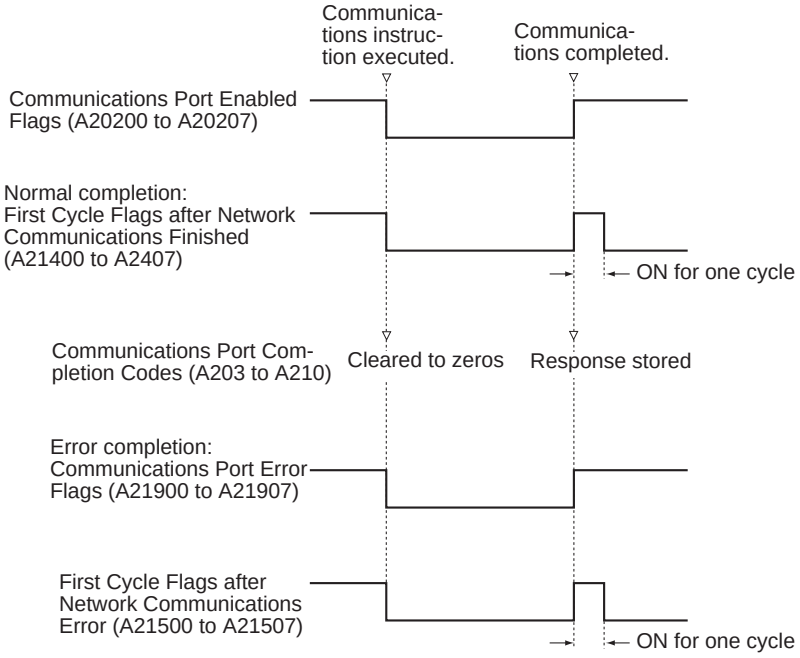
Note 1. Use the following flowchart to determine whether to use the Network Communications Port Allocation Enabled Flag (A20215) and the Network Communications Completion Code Storage Address (A216 and A217).



2. The Auxiliary Area bits and words used for user-specified communications ports are listed in the following table.

Address	Bits	Name	Description
A202	00 to 07	Communications Port Enabled Flags	ON when a communications instruction can be executed with the corresponding port number. Bits 00 to 07 correspond to communications ports 0 to 7. The completion of communications can be confirmed by monitoring when a flag turns ON. The flag will turn OFF when execution of a communications instruction is started.
A203 to A210	---	Communications Port Completion Codes	These words contain the completion codes for the corresponding port numbers when communications instructions have been executed. Words A203 to A210 correspond to communications ports 0 to 7.
A219	00 to 07	Communications Port Error Flags	ON when an error occurred during execution of a communications instruction. When a flag is ON, check the completion code in A203 to A210 to troubleshoot the error. Turn OFF then execution has been finished normally. Bits 00 to 07 correspond to communications ports 0 to 7.

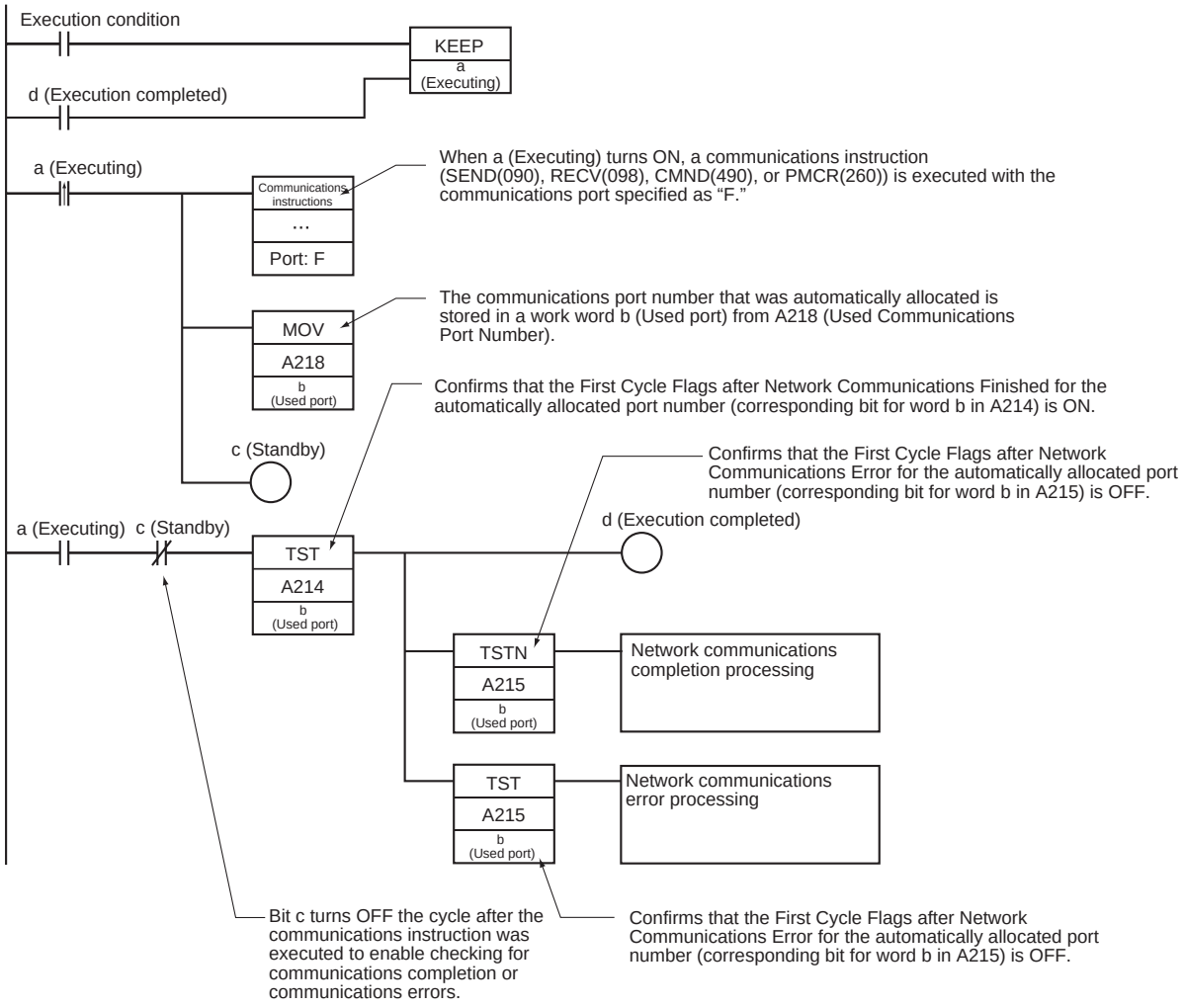
Flag/Word Operation



■ Applications Methods

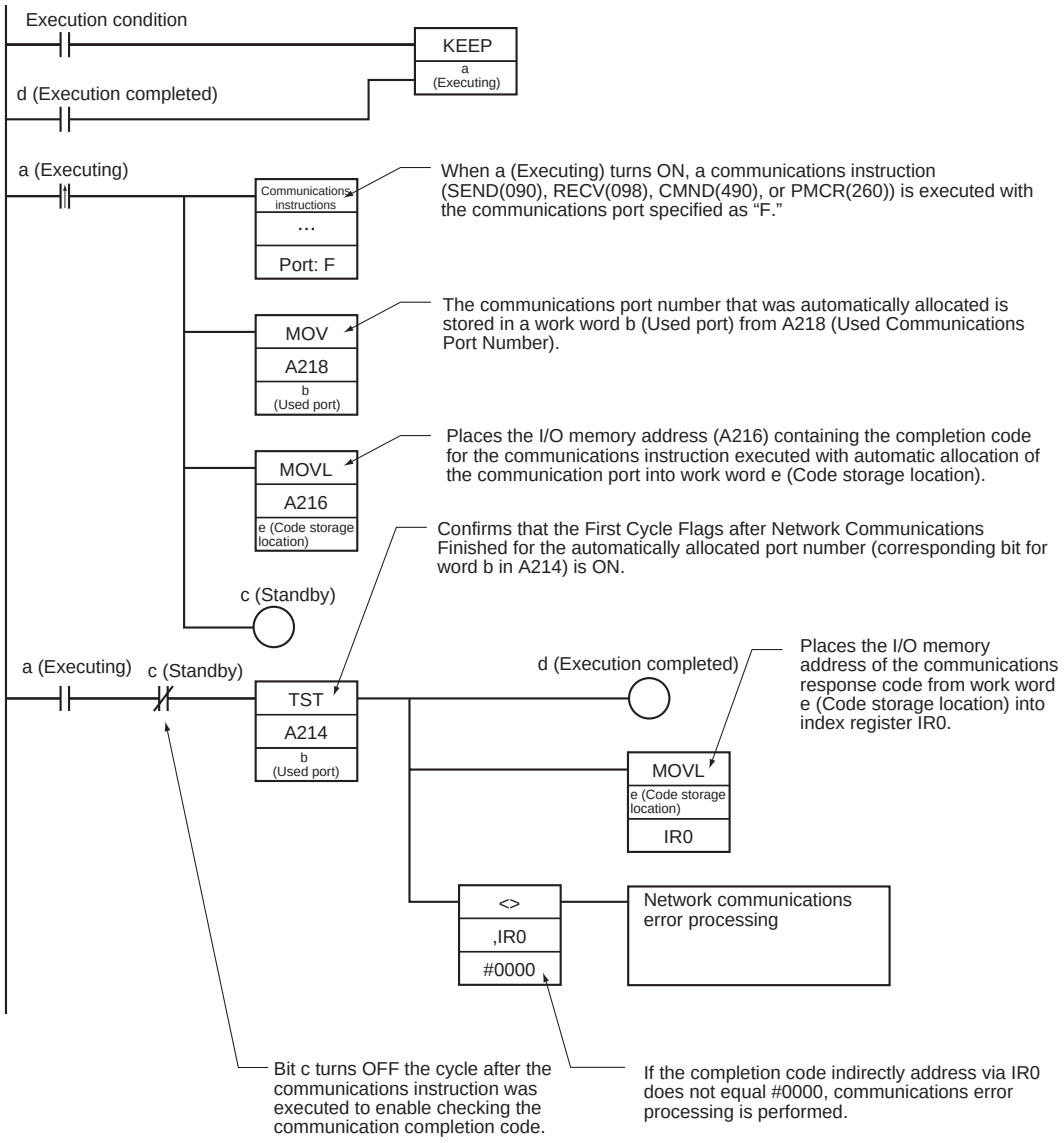
To use automatic communications port allocation, set the communications port number of “F” and then program as shown below.

Completing and Processing Error after Executing Communications Instructions



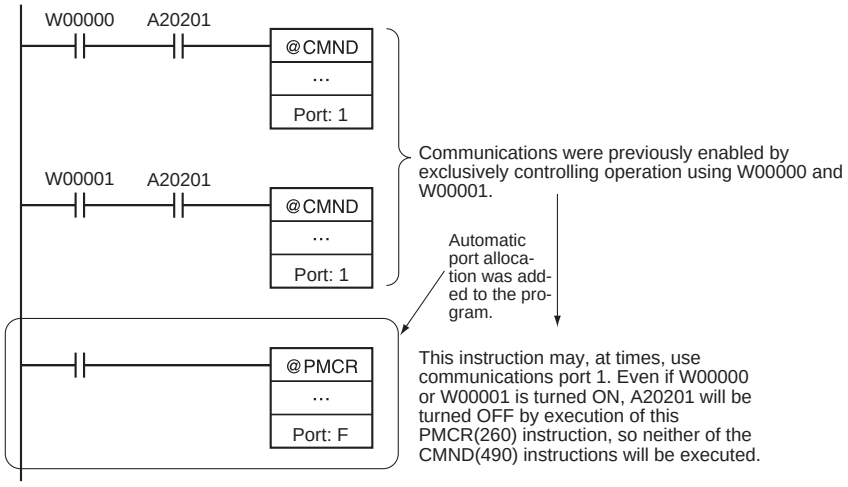
Accessing the Completion Code after Executing Communications Instructions

The completion codes are generally used to troubleshoot errors when they occur. A completion code of 0000 hex can, however, also be used to confirm that communications have completed normally.



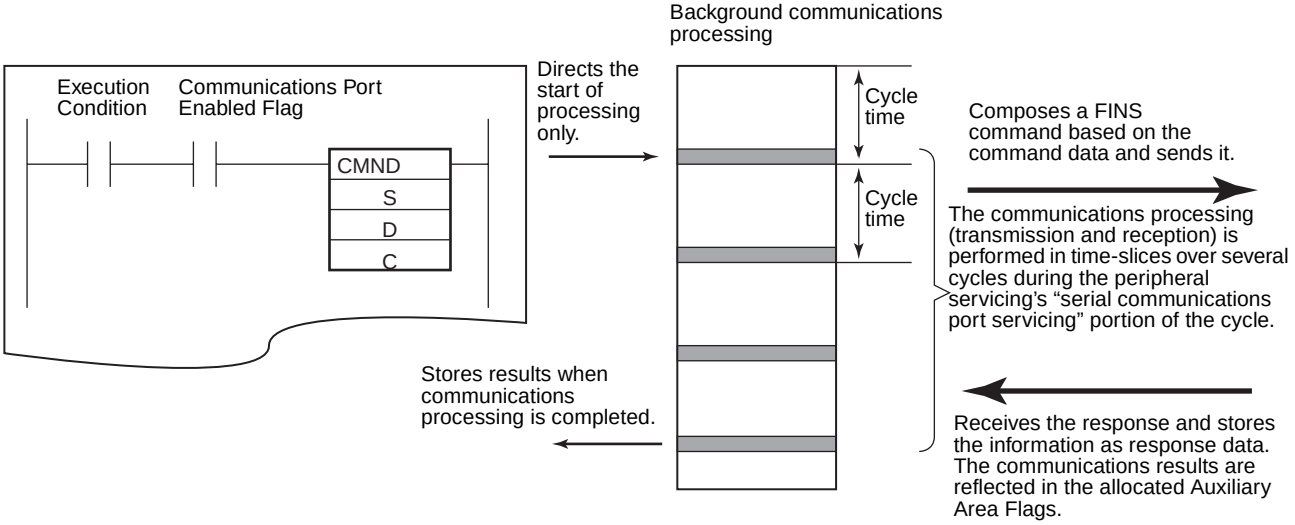
Note Both user-specified communications port numbers and automatically specified communications port numbers can be used in the same program. It is possible, however, that the communications port numbers specified by the user will be used for automatic allocation. It is thus important to check the program carefully when adding communications instructions that use automatic communications port allocation to an existing program, as shown in the following example.

Programming Example



Timing the Execution of Network Instructions

A Network Instruction just starts the communications processing when its execution condition is established. The actual communications processing is executed in the background in the "serial communications port servicing" portion of peripheral servicing.

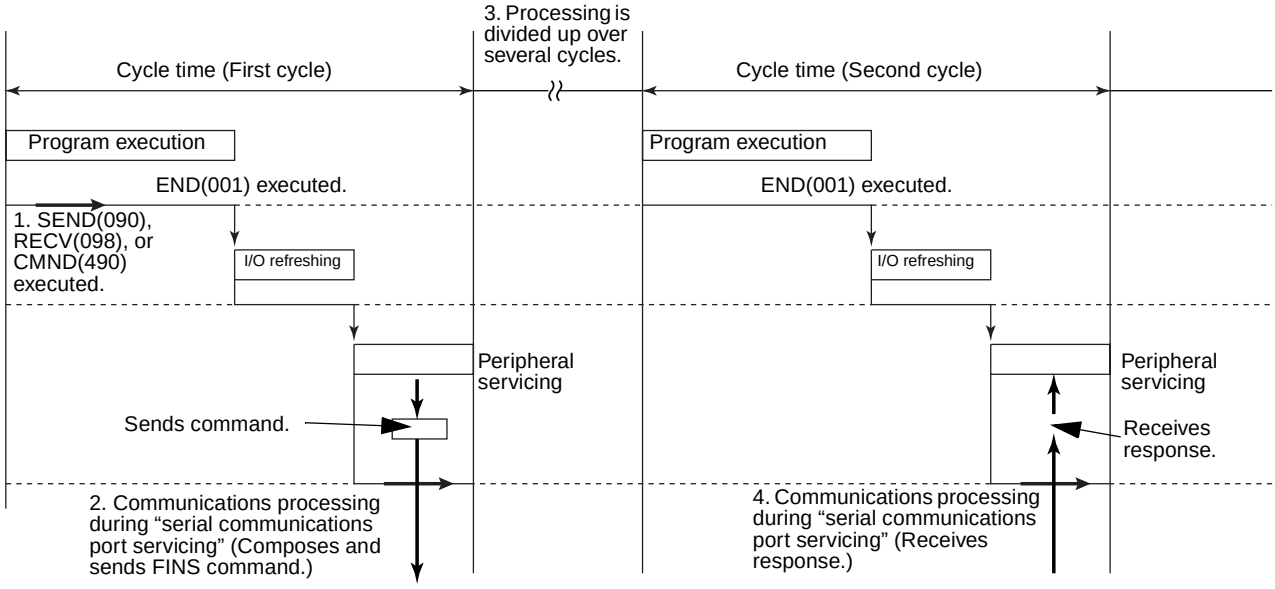


The communications processing is performed as follows:

1. If the corresponding Communications Port Enabled Flag (A20200 to A20207) is ON when the execution condition is established, the system performs the following processes:
   - Turns OFF the port's Communications Port Enabled Flag and Communications Port Error Flag (A21900 to A21907).
   - Sets the port's Communications Port Completion Code (A203 to A210) to 0000.
   - Reads the control words (beginning at C) and starts communications processing (sending a FINS command or receiving a response.)
2. In the peripheral servicing's "serial communications port servicing" portion of the cycle, the system composes a FINS command based on the operands (see note) and sends the FINS command to the Communications Unit or other destination node.

Note When SEND(090) is being executed, the contents of S and D are read and a FINS command for data transmission is composed.
When RECV(098) is being executed, the content of S is read and a FINS command for data reception is composed.
When CMND(490) is being executed, the content of S is read and the corresponding FINS command is composed.

3. If the send processing cannot be completed in a the time available in "serial communications port servicing" period, the processing will be continued in the next cycle's serial communications port servicing.
4. When a response is returned, the system performs the following processes:
   - Refreshes the destination words specified in the Network instruction with the response data.
   - Turns ON the port's Communications Port Enabled Flag.
   - Refreshes the port's Communications Port Error Flag (A21900 to A21907) and Communications Port Completion Code (A203 to A210).



3-25-2 About Explicit Message Instructions

Methods for Using Explicit Message Communications

There are two methods that can be used to send explicit messages from a PLC.

- Use the CMND(490) to send a FINS command code of 2801 hex (EXPLICIT MESSAGE SEND).
- Use the following Explicit Message Instructions. (See note.)

Note These instructions are supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Explicit Message Instructions

The following instructions, which are used specially for explicit messages, are called Explicit Message Instructions.

Instruction	Name	Outline
EXPLT(720)	EXPLICIT MESSAGE SEND	Sends an explicit message with any service code. Note: Functionally, this instruction is the same as sending CMND(490) with a FINS command code of 2801 hex.
EGATR(721)	EXPLICIT GET ATTRIBUTE	Sends an explicit message with a service code of 0E hex (GET ATTRIBUTE SINGLE).
ESATR(721)	EXPLICIT SET ATTRIBUTE	Sends an explicit message with a service code of 10 hex (SET ATTRIBUTE SINGLE).
EGATR(721)	EXPLICIT WORD READ	Uses an explicit message to read data from a CPU Unit.
EGATR(721)	EXPLICIT WORD WRITE	Uses an explicit message to write data to a CPU Unit.

Features of Explicit Message Instructions

- Explicit Message Instructions do not require giving a 2801 hex FINS command and are much simpler to program than CMND(490).
- With the EXPLICIT GET/SET ATTRIBUTE instructions, entering the service code is not required and only information from the class ID onward needs to be entered.
- With the EXPLICIT WORD READ/WRITE instructions, the I/O memory address in the local and remote CPU Units can be specified directly.

  Code specifications for area types and hexadecimal word addresses are not required. (These are required for CMND(490) instructions with service code 1E (word data read) or 1F hex (word data write).)
  This enables easy reading and writing of data between CPU Units using explicit message communications (like SEND/RECV instructions for FINS commands).

Operation

The Explicit Communications Error Flag is used to determine if communications ended normally or in error.

For error completions (i.e., when the flag is ON), the Communications Port Error Flag for FINS commands is used to determine if the explicit message was never sent (i.e., when the flag is ON) or if there was an error in the explicit message that was sent (i.e., when the flag is OFF).

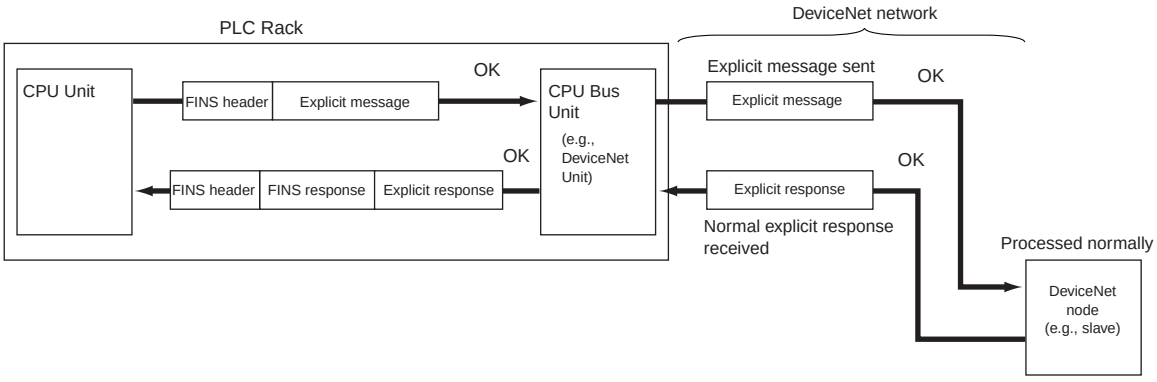
The Communications Port Completion Code will contain 0000 hex after a normal end, an explicit message error code after an explicit communications error end, and a FINS message completion code after a FINS error end.

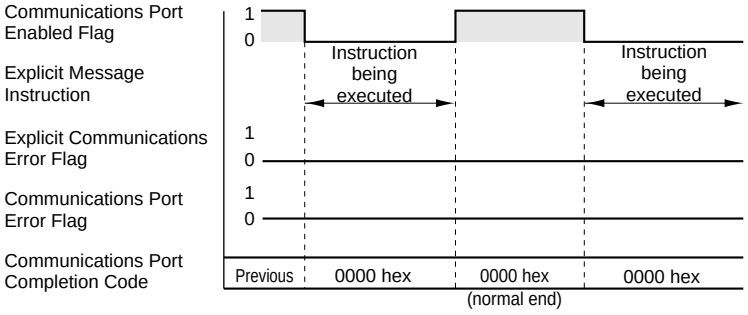
Condition		Explicit Communications Error Flag (A21300 to A21307: Communications port No. 0 to 7)	Communications Port Error Flag (A21900 to A21907: Communications port No. 0 to 7)	Communications Port Completion Code (A203 to A210: Communications port No. 0 to 7)
1) Normal end		OFF	OFF	0000 hex
2) Error end	a) When the explicit message could not be sent	ON	ON	FINS messages completion code
	b) When the explicit message was sent but an explicit error response was returned		OFF	Explicit message error code

1) Normal End

An explicit message is sent and a normal response is returned.

The corresponding Explicit Communications Error Flag (A21300 to 07: Communications port No. 0 to 7) will be OFF and the Network Communications Response Code (A203 to A210: Communications port No. 0 to 7) will contain the explicit message normal response code of 0000 hex.





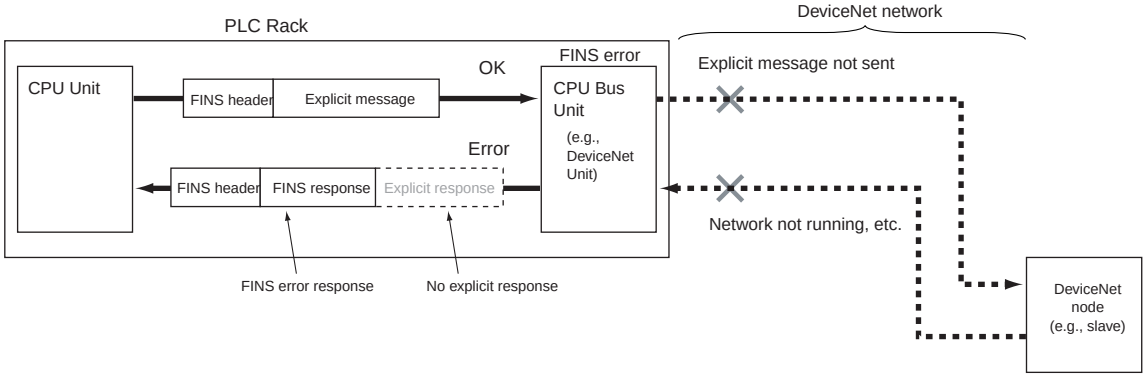
2) Error End

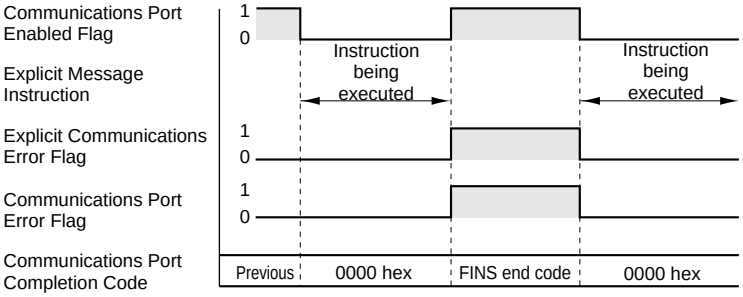
The are two possibilities for error ends, as described in the next two subsections.

a) When the Explicit Message Could Not Be Sent

In this case, the explicit message was never sent on the network, e.g., because the network was not running. Here, both the Explicit Communications Error Flag (A21300 to A21307: Communications port No. 0 to 7) and the Communications Port Error Flag (A21900 to A21907: Communications port No. 0 to 7) will turn ON.

After completion, the Communications Port Completion Code (A203 to A210: Communications port No. 0 to 7) will contain the FINS message error code.

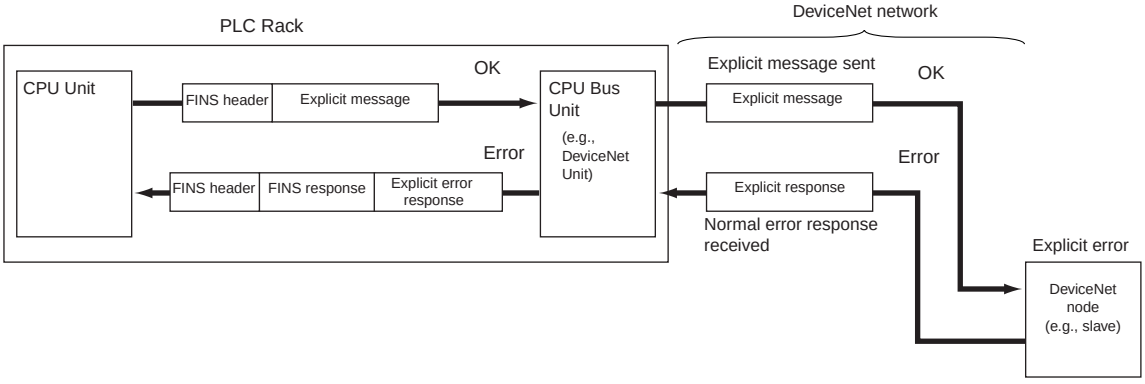


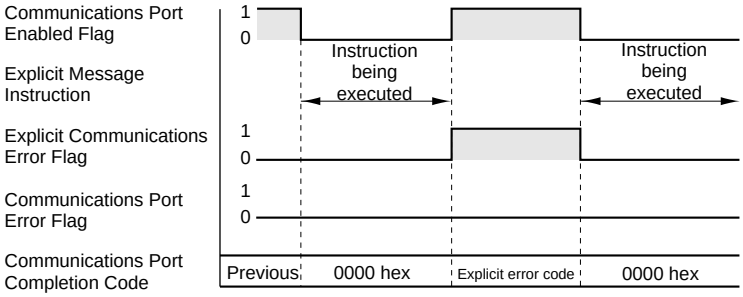


b) When the Explicit Message Was Sent But an Explicit Error Response Was Returned

In this case, the explicit message was sent but an error existed in the explicit message command frame (code not supported, illegal size, etc.). Here, the Explicit Communications Error Flag (A21300 to 07: Communications port No. 0 to 7) will turn ON and the Network Communications Error Flag (A21900 to 07: Communications port No. 0 to 7) will remain OFF.

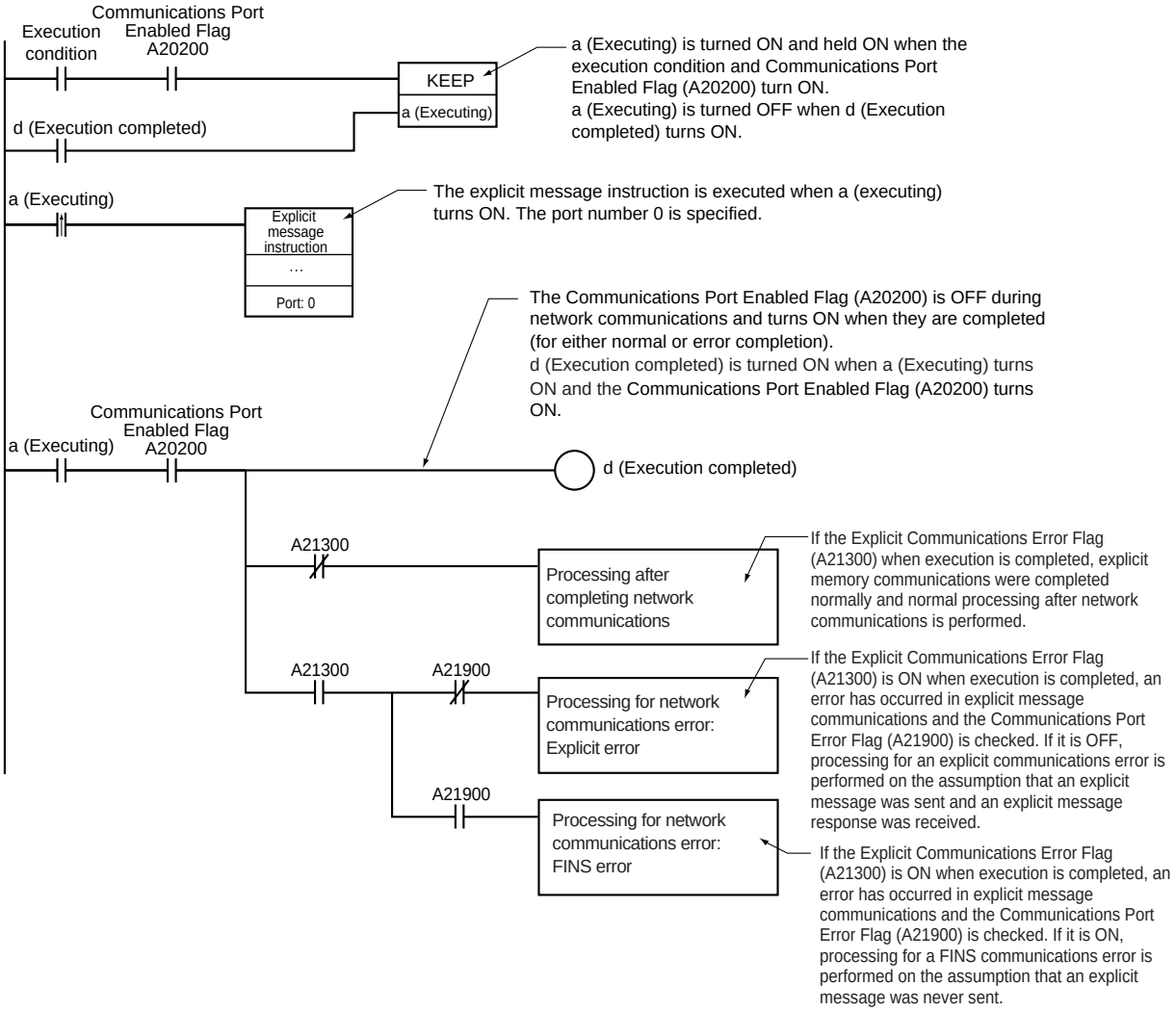
After completion, the Network Communications Response Code (A203 to A210: Communications port No. 0 to 7) will contain the explicit message error code.



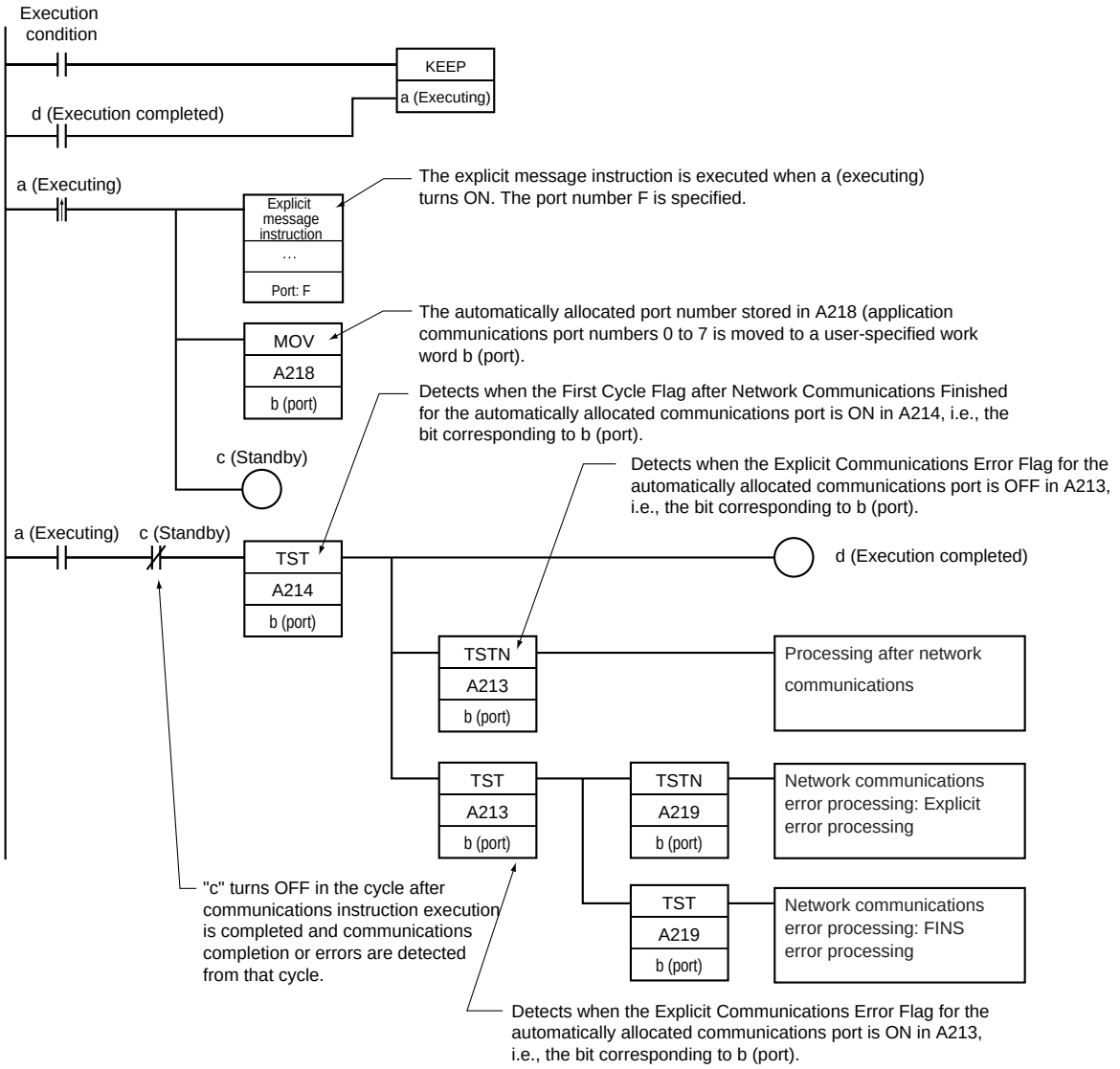


Ladder Programming Examples

Example 1: User Specification of Communications Port Number



Example 2: Automatic Allocation of Communications Port Number

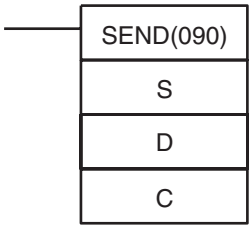


3-25-3 NETWORK SEND: SEND(090)

Purpose

Sends data to a node in the network.

Ladder Symbol



S: First source word (local node)

D: First destination word (remote node)

C: First control word

Variations

Variations	**Executed Each Cycle for ON Condition**	SEND(090)
	Executed Once for Upward Differentiation	@SEND(090)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: First control word

The five control words C to C+4 specify the number of words being transmitted, the destination, and other settings shown in the following table.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words: 0001 to maximum allowed[1] (4-digit hexadecimal)	
C+1	Destination network address: 00 to 7F (0 to 127)[2, 4]	Bits 08 to 11:Serial port number[3] (physical port) 1 hex: Port 1 2 hex: Port 2 (Do not set 0, 3, or 4.) Bits 12 to 15: Always 0.
C+2	Destination unit address: 00 to FE[5]	Destination node address: 00 to maximum allowed[6]
C+3	No. of retries: 00 to 0F (0 to 15)	Bits 08 to 11: Communications port number (internal logic port): 0 to 7, Automatic allocation: F[7] Bits 12 to 15: Response setting 0: Response requested. 8: No response requested.[8]
C+4	Response monitoring time: 0001 to FFFF (0.1 to 6553.5 seconds) (The default setting of 0000 sets a monitoring time of 2 seconds.)	

Note

1. The maximum number of words allowed depends on the network being used. For a Controller Link, the allowed range is 0001 to 03DE (1 to 990 words).
2. Set the destination network address to 00 to transmit within the local network. When two or more CPU Bus Units are mounted, the network address will be the unit number of the Unit with the lowest unit number.
3. The following two methods can be used to send data to the host computer through a serial port with the host link while initiating communications from the PLC.
   a) Set the destination unit address (bits 00 to 07 of C+2) to the unit address of the CPU Unit or 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

b) Set the destination unit address directly into bits 00 to 07 of C+2. In this case, set the serial port number in bits 08 to 11 of C+1 to 0 for direct specification.

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)

4. When specifying the serial port without a routing table for the serial gateway function (conversion to host link FINS), set the serial port's unit address in the destination network address byte.
5. The unit address indicates the Unit, as shown in the following table.

Unit	Unit address setting
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)

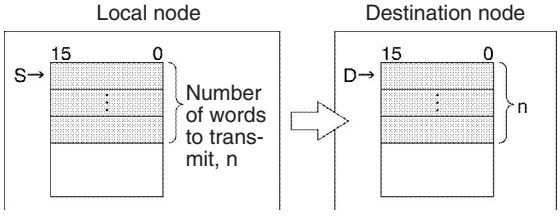
6. The maximum node number depends on the network being used. For a Controller Link, the allowed range is 00 to 20 hexadecimal (0 to 32). Set the destination node number to FF to broadcast to all nodes; set it to 00 to transmit within the local node.
7. Refer to *Automatic Allocation of Communications Ports* on page 1032 for details on using automatic allocation of the communications port number (logical port).
8. When the destination node number is set to FF (broadcast transmission), there will be no response even if bits 12 to 15 are set to 0.

Operand Specifications

Area	S	D	C
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6139
Work Area	W000 to W511		W000 to W507
Holding Bit Area	H000 to H511		H000 to H507
Auxiliary Bit Area	A000 toA959		A000 to A955
Timer Area	T0000 to T4095		T0000 to T4091
Counter Area	C0000 to C4095		C0000 to C4091
DM Area	D00000 to D32767		D00000 to D32763
EM Area without bank	E00000 to E32767		E00000 to E32763
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32763 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 o C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

SEND(090) transfers the data beginning at word S to addresses beginning at D in the designated device through the PLC's CPU Bus or over a network. The number of words to be transmitted is specified in C.



If the destination node number is set to FF, the data will be broadcast to all of the nodes in the designated network. This is known as a broadcast transmission.

If a response is requested (bits 12 to 15 of C+3 set to 0) but a response has not been received within the response monitoring time, the data will be retransmitted up to 15 times (retries set in bits 0 to 3 of C+3). There will be no response or retries for broadcast transmissions.

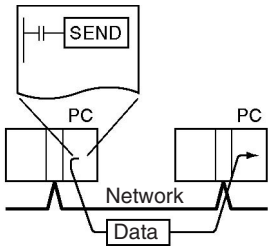
SEND(090) can be used to transmit data to a particular serial port in the destination device as well as the device itself.

Data can be transmitted to a host computer connected to the PLC's serial port (when set to host link mode) as well as a PLC or computer connected through a Controller Link or Ethernet network.

If the Communications Port Enabled Flag is ON for the communications port specified in C+3 when SEND(090) is executed, the corresponding Communications Port Enabled Flag (ports 00 to 07: A20200 to A20207) and Communications Port Error Flag (ports 00 to 07: A21900 to A21907) will be turned OFF and 0000 will be written to the word that contains the completion code (ports 00 to 07: A203 to A210). Data will be transmitted to the destination node once the flags have be set.

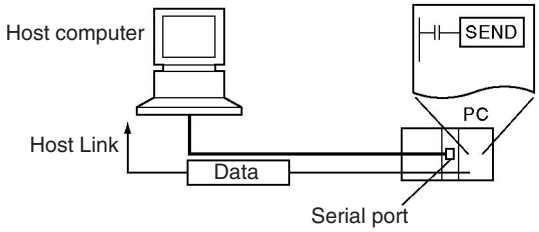
Transmission through the Network

SEND(090) can be used to transmit data from the PLC to the specified data area in a PLC or computer connected by a Controller Link network or Ethernet link.



Transmission through Host Link

When 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.



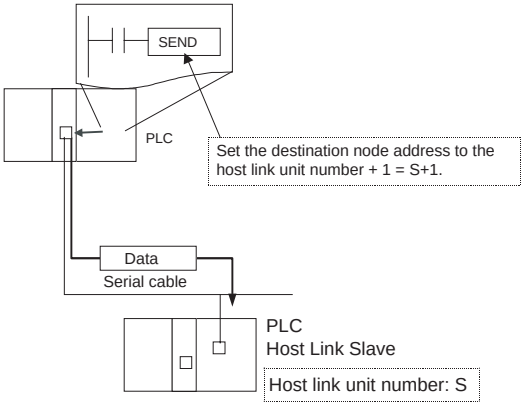
If SEND(090) is sent to the serial port of the CPU Unit, a Serial Communications Board (CS Series only), or Serial Communications Unit, a command is sent from the serial port to the host computer. The command is a FINS message enclosed between a host link header and terminator. The FINS command is a 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).

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.



Flags

Name	Label	Operation
Error Flag	ER	ON if the serial port number specified in C+1 is not within the range of 00 to 04. ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C+3. OFF in all other cases.

The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A20200 to A20207	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	A21900 to A21907	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of a network instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when an instruction is executed.

Precautions

If the Communications Port Enabled Flag is OFF for the port number specified in C+3, the instruction will be treated as NOP(000) and will not be executed. The Error Flag will be turned ON in this case.

When an address in the current bank of the EM Area is specified for D, the transmitted data will be written to the current EM bank of the destination node.

When data will be transmitted outside of the local network, the user must register routing tables in the PLCs (CPU Units) in each network. (Routing tables indicate the routes to other networks in which destination nodes are connected.)

Refer to the FINS command response codes in the *CS/CJ Series Communications Commands Reference Manual* (W342) for details on the completion codes for network communications.

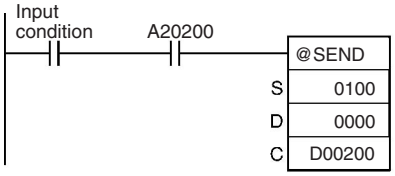
Only one network instruction may be executed for a communications port at one time. To ensure that SEND(090) is not executed while a port is busy, program the port's Communications Port Enabled Flag (A20200 to A20207) as a normally open condition.

Communications port numbers 00 to 07 are shared by the network instructions and PMCR(260), so SEND(090) cannot be executed simultaneously with PMCR(260) if the instructions are using the same port number.

Noise and other factors can cause the transmission or response to be corrupted or lost, so we recommend setting the number of retries to a non-zero value which will cause SEND(090) to be executed again if the response is not received within the response monitoring time.

Example 1

When the input condition and A20200 (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.



C	D00200	0 0 : 0 A	Number of words to send: 10 words
C+1	D00201	0 1 : 0 0	Transmit to network 0 and port 1 of Serial Communications Board
C+2	D00202	0 0 : 1 0	Node number 0, unit address 10
C+3	D00203	0 0 : 0 0	Response requested, port number 0, no retries
C+4	D00204	0 0 : 0 0	Response monitoring time: 2 seconds (0000: default value)

It is necessary create a program at the host computer to receive the data and send a response.

Example 2

When CIO 000000 and A20207 (the Communications Port Enabled Flag for port 07) are ON in the following example, the ten words from D00100 to D00109 are transmitted to node number 3 in the local network where they are written to the ten words from D00200 to D00209. The data will be retransmitted up to 3 times if a response is not received within ten seconds.

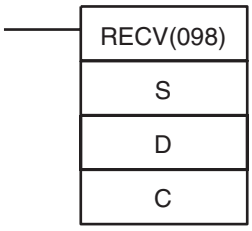
C: D00300	0 0 : 0 A	Number of words to send: 10 words
C+1: D00301	0 0 : 0 0	Transmit to the local network and the device itself
C+2: D00302	0 3 : 0 0	Node number 3, unit address 00 (CPU Unit)
C+3: D00303	0 7 : 0 3	Response requested, port number 7, 3 retries
C+4: D00304	0 0 : 6 4	Response monitoring time: 0064 hexadecimal (10 seconds)

3-25-4 NETWORK RECEIVE: RECV(098)

Purpose

Requests data to be transmitted from a node in the network and receives the data.

Ladder Symbol



S: First source word (remote node)

D: First destination word (local node)

C: First control word

Variations

Variations		
	Executed Each Cycle for ON Condition	RECV(098)
	Executed Once for Upward Differentiation	@RECV(098)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: First control word

The five control words C to C+4 specify the number of words to be received, the source of the transmission, and other settings shown in the following table.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words: 0001 to maximum allowed[1] (4-digit hexadecimal)	
C+1	Source network address: 00 to 7F (0 to 127)[2, 4]	Bits 08 to 11:Serial port number (physical port) 1 hex: Port 1 2 hex: Port 2 (Do not set 0, 3, or 4.) Bits 12 to 15: Always 0.
C+2	Source unit address[5]	Source node address: 00 to maximum allowed[6]
C+3	No. of retries: 00 to 0F (0 to 15)	Port number: 00 to 07 (F: Automatic allocation)[7] Response is fixed to “required.”
C+4	Response monitoring time: 0001 to FFFF (0.1 to 6553.5 seconds) (The default setting of 0000 sets a monitoring time of 2 seconds.)	

Note

1. The maximum number of words allowed depends on the network being used. For a Controller Link, the allowed range is 0001 to 03DE (1 to 990 words).
2. Set the source network address to 00 to specify a source within the local network. When two or more CPU Bus Units are mounted, the network address will be the unit number of the Unit with the lowest unit number.
3. The following two methods can be used to receive data from a host computer through a serial port with the host link while initiating communications from the PLC.

a) Set the source unit address (bits 00 to 07 of C+2) to the unit address of the 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

b) Set the source unit address directly into bits 00 to 07 of C+2. In this case, set the serial port number in bits 08 to 11 of C+1 to 0 for direct specification.

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)

4. When specifying the serial port without a routing table for the serial gateway function (conversion to host link FINS), set the serial port's unit address in the source network address byte.
5. The unit address indicates the Unit, as shown in the following table.

Unit	Unit address setting
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	Unit address setting
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)

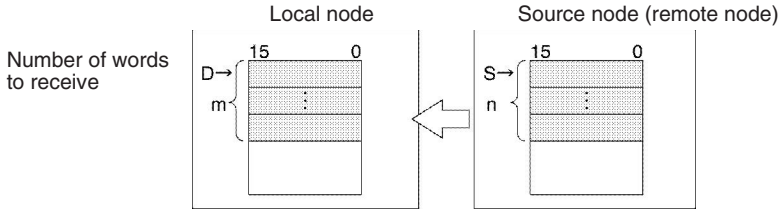
6. The maximum node number depends on the network being used. For a Controller Link, the allowed range is 00 to 20 hexadecimal (0 to 32). Set the source node number to 00 to transmit within the local node.
7. Refer to *Automatic Allocation of Communications Ports* on page 1032 for details on using automatic allocation of the communications port number (logical port).

Operand Specifications

Area	S	D	C
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6139
Work Area	W000 to W511		W000 to W507
Holding Bit Area	H000 to H511		H000 to H507
Auxiliary Bit Area	A000 to A447 A448 to A959	A448 to A959	A000 to A443 A448 to A955
Timer Area	T0000 to T4095		T0000 to T4091
Counter Area	C0000 to C4095		C0000 to C4091
DM Area	D00000 to D32767		D00000 to D32763
EM Area without bank	E00000 to E32767		E00000 to E32763
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32763 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

RECV(098) requests the number of words specified in C beginning at word S to be transferred from the designated device to the local PLC. The data is received through the PLC's CPU Bus or over the network and written to the PLC's data area beginning at D.



A response is required with RECV(098) because the response contains the data being received. If the response has not been received within the response monitoring time set in C+4, the request for data transfer will be retransmitted up to 15 times (retries set in bits 0 to 3 of C+3).

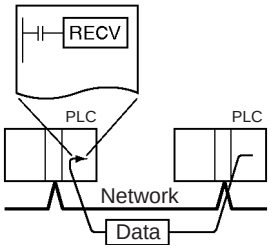
RECV(098) can be used to request a data transmission from a particular serial port in the source device as well as the device itself.

Data can be received from a host computer connected to the PLC's serial port (when set to host link mode) as well as a PLC or computer connected through a Controller Link or Ethernet network.

If the Communications Port Enabled Flag is ON for the communications port specified in C+3 when SEND(090) is executed, the corresponding Communications Port Enabled Flag (ports 00 to 07: A20200 to A20207) and Communications Port Error Flag (ports 00 to 07: A21900 to A21907) will be turned OFF and 0000 will be written to the word that contains the completion code (ports 00 to 07: A203 to A210). Data will be received from the destination node once the flags have be set.

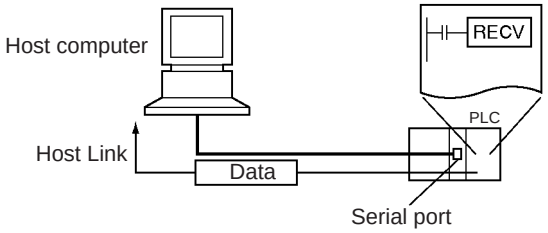
Transmission through the Network

RECV(098) can be used to receive data transmitted the specified data area in a PLC or computer connected by a Controller Link network or Ethernet link and write that data to the specified data area in the local PLC.



Transmission through Host Link

When the 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.



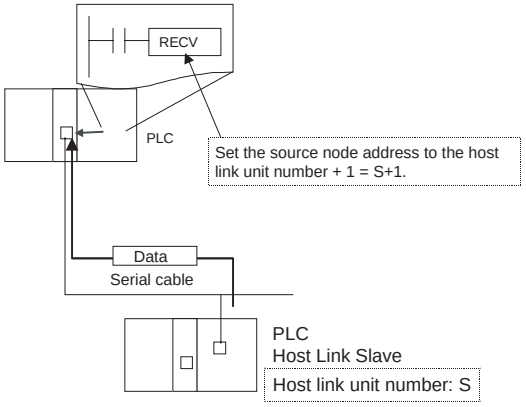
If RECV(098) is executed for the serial port of the CPU Unit, a Serial Communications Board (CS Series only), or Serial Communications Unit, a command is sent from the serial port to the host computer. The command is a FINS message enclosed between a host link header and terminator. The FINS command 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).

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.



Flags

Name	Label	Operation
Error Flag	ER	ON if the serial port number specified in C+1 is not within the range of 00 to 04. ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C+3. OFF in all other cases.

The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A20200 to A20207	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.

Name	Address	Operation
Communications Port Error Flag	A21900 to A21907	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of a network instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

Precautions

If the Communications Port Enabled Flag is OFF for the port number specified in C+3, the instruction will be treated as NOP(000) and will not be executed. The Error Flag will be turned ON in this case.

When an address in the current bank of the EM Area is specified for D, the transmitted data will be written to the current EM bank of the destination node.

When data will be transmitted outside of the local network, the user must register routing tables in the PLCs (CPU Units) in each network. (Routing tables indicate the routes to other networks in which destination nodes are connected.)

Refer to the FINS command response codes in the *CS/CJ Series Communications Commands Reference Manual* (W342) for details on the completion codes for network communications.

Only one network instruction may be executed for a communications port at one time. To ensure that RECV(098) is not executed while a port is busy, program the port's Communications Port Enabled Flag (A20200 to A20207) as a normally open condition.

Communications port numbers 00 to 07 are shared by the network instructions and PMCR(260), so RECV(098) cannot be executed simultaneously with PMCR(260) if the instructions are using the same port number.

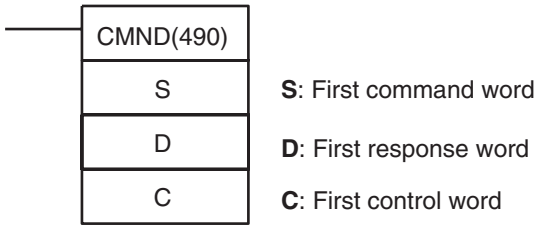
Noise and other factors can cause the transmission or response to be corrupted or lost, so we recommend setting the number of retries to a non-zero value which will cause RECV(098) to be executed again if the response is not received within the response monitoring time.

3-25-5 DELIVER COMMAND: CMND(490)

Purpose

Sends an FINS command and receives the response. Refer to the *CS/CJ Series Communications Commands Reference Manual* for details on FINS commands.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	CMND(490)
	Executed Once for Upward Differentiation	@CMND(490)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Operands

C: First control word

The six control words C to C+5 specify the number of bytes of command data and response data, the destination, and other settings shown in the following table.

Word	Bits 00 to 07	Bits 08 to 15
C	Bytes of command data: 0002 to maximum allowed[1] (4-digit hexadecimal)	
C+1	Bytes of response data: 0000 to maximum allowed[1 to 3] (4-digit hexadecimal)	
C+2	Destination network address: 00 to 7F (0 to 127)[4, 6]	Bits 08 to 11:Serial port number (physical port) 1 hex: Port 1 2 hex: Port 2 (Do not set 0, 3, or 4.) Bits 12 to 15: Always 0.
C+3	Destination unit address: 00 to FE[5, 7, 9]	Destination node number: 00 to maximum allowed[8]
C+4	No. of retries: 00 to 0F (0 to 15)	Bits 08 to 11: Port number (internal logic port): 0 to 7 (F: Automatic allocation)[10] Bits 12 to 15: Response setting 0: Response requested. 8: No response requested.[11]
C+5	Response monitoring time: 0001 to FFFF (0.1 to 6553.5 seconds) (The default setting of 0000 sets a monitoring time of 2 seconds.)	

Note

1. The number of bytes of command data in C is 0002 to the maximum data length in hexadecimal. For example, the number of bytes would be 0002 to 07C6 hex (2 to 1,990 bytes) for Controller Link systems. The number of bytes for the local CPU Unit is 07C6 hex (1,990 bytes). The number of bytes of command data depends on the network.
2. The number of bytes of response data in C+1 is 0000 to the maximum data length in hexadecimal. For example, the number of bytes would be 0000 to 07C6 hex (0 to 1,990 bytes) for Controller Link systems. The number of bytes for the local CPU Unit is 07C6 hex (1,990 bytes). The number of bytes of response data depends on the network.
3. Refer to the operation manual for the specific network for the maximum data lengths for the command data and response data. For any FINS command passing through multiple networks, the maximum data lengths for the command data and response data are determined by the network with the smallest maximum data lengths.
4. Set the destination network address to 00 to transmit within the local network. When two or more CPU Bus Units are mounted, the network address will be the unit number of the Unit with the lowest unit number.

5. The following two methods can be used to send a FINS command to a host computer through a serial port with the host link host link while initiating communications from the PLC, or the serial gateway function (converted to CompoWay/F, Modbus-RTU, or Modbus-ASCII).

   a) Set the destination unit address (bits 00 to 07 of C+3) to the unit address of the 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+3, bits 00 to 07)	Unit	Serial port number (C+2, 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

   b) Set the destination unit address directly into bits 00 to 07 of C+3. In this case, set the serial port number in bits 08 to 11 of C+2 to 0 for direct specification.

   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)

6. When specifying the serial port without a routing table for the serial gateway function (conversion to host link FINS), set the serial port's unit address in the destination network address byte.

7. The unit address indicates the Unit, as shown in the following table.

Unit	Unit address setting
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

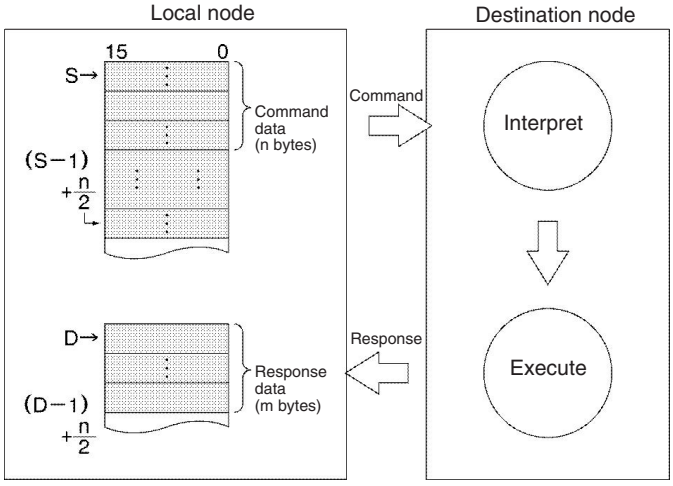
Unit	Unit address setting
Unit connected to network (not necessary to specify Unit)	FE hex
Direct specification of the serial port's unit address	Serial Communications Unit ports Port 1: 80 hex + 4 × unit number Port 2: 81 hex + 4 × unit number Serial Communications Board ports Port 1: E4 hex (228 decimal) Port 2: E5 hex (229 decimal) CPU Unit ports Peripheral port: FD hex (253 decimal) RS-232C port: FC hex (252 decimal)

8. The maximum node number depends on the network being used. For a Controller Link, the allowed range is 00 to 20 hexadecimal (0 to 32). Set the destination node number to FF to broadcast to all nodes; set it to 00 to transmit within the local node.
9. When specifying the serial port in the serial gateway function (conversion to host link FINS), set the destination unit address to the host link unit number of the destination PLC + 1 (setting range: 1 to 32).
10. Refer to *Automatic Allocation of Communications Ports* on page 1032 for details on using automatic allocation of the communications port number (logical port).
11. When the destination node number is set to FF (broadcast transmission), there will be no response even if bits 12 to 15 are set to 0.

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6138
Work Area	W000 to W511		W000 to W506
Holding Bit Area	H000 to H511		H000 to H506
Auxiliary Bit Area	A000 to A447 A448 to A959	A448 to A959	A000 to A442 A448 to A954
Timer Area	T0000 to T4095		T0000 to T4090
Counter Area	C0000 to C4095		C0000 to C4090
DM Area	D00000 to D32767		D00000 to D32762
EM Area without bank	E00000 to E32767		E00000 to E32762
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32763 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

CMND(490) transfers the specified number of bytes of FINS command data beginning at word S to the designated device through the PLC's CPU Bus or over a network. The response is stored in memory beginning at word D.



CMND(490) can be used to transmit command data to a particular serial port in the destination device as well as the device itself. CMND(490) operates just like SEND(090) if the FINS command code is 0102 (MEMORY AREA WRITE) and just like RECV(098) if the code is 0101 (MEMORY AREA READ).

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."

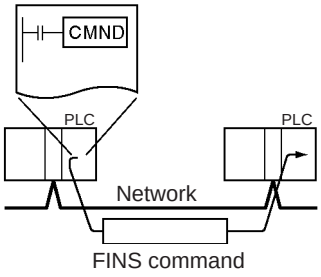
An error will occur if the amount of response data exceeds the number of bytes of response data set in C+1.

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.

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: A20200 to A20207) and Communications Port Error Flag (ports 00 to 07: A21900 to A21907) will be turned OFF and 0000 will be written to the word that contains the completion code (ports 00 to 07: A203 to A210). The command data will be transmitted to the destination node(s) once the flags have be set.

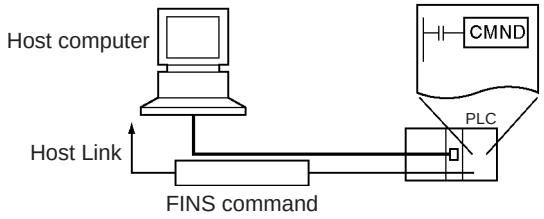
Transmission through the Network

CMND(490) can be used to transmit any FINS command to a personal computer or a PLC (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.



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 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).

Serial Gateway Communications to a Component or Host Link Slave

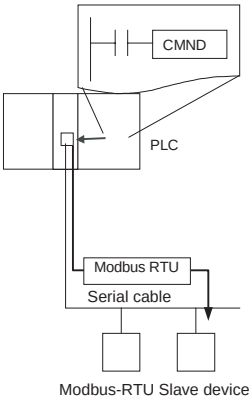
It is possible to send FINS commands (or send/receive data) to a component or Host Link Slave connected to the PLC through its serial port with the serial gateway function.

- Sending to a Component
  (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).

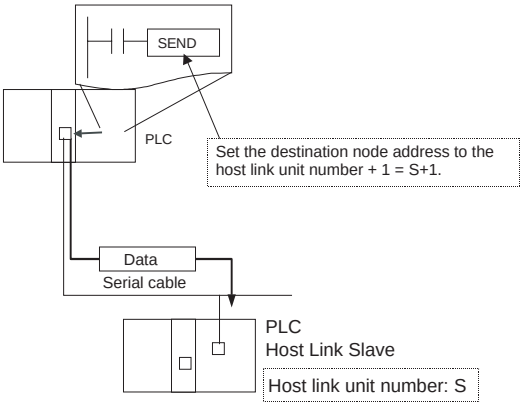
  Convert to CompoWay/F command: 2803 hex
  Convert to Modbus-RTU command: 2804 hex (See note.)
  Convert to Modbus-ASCII command: 2805 hex (See note.)

  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

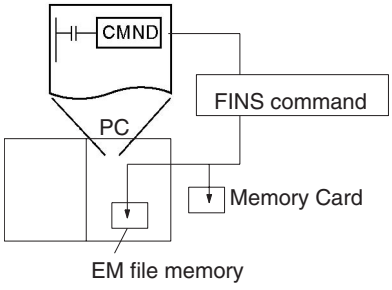
  The serial gateway function can be used to send any FINS command to a PLC that is connected as a host link slave and through a Serial Communications Board or Unit's serial port. In this case, the destination node address must be set to the host link unit number + 1.



Sending a FINS Command to the CPU Unit Executing CMND(490) (Except for CS-series CS1 CPU Units Prior to V1)

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* for details.

The File Memory Operation Flag (A34313) 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 A34313 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.



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. OFF in all other cases.

The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A20200 to A20207	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	A21900 to A21907	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	A34313	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

Precautions

If the Communications Port Enabled Flag is OFF for the port number specified in C+4, the instruction will be treated as NOP(000) and will not be executed. The Error Flag will be turned ON in this case.

When data will be transmitted outside of the local network, the user must register routing tables in the PLCs (CPU Units) in each network. (Routing tables indicate the routes to other networks in which destination nodes are connected.)

Refer to the FINS command response codes in the *CS/CJ Series Communications Commands Reference Manual* (W342) for details on the completion codes for network communications.

Communications port numbers 00 to 07 are shared by the network and serial communications instruction instructions (SEND(090), RECV(098), CMND(490), PMCR(260), TXDU(256), or RXDU(255)), so only one of these instructions may be executed for a communications port at one time. To

ensure that CMND(490) is not executed while a port is busy, program the port's Communications Port Enabled Flag (A20200 to A20207) as a normally open condition.

Always use one of the Communications Port Enabled Flags (A20200 to A20207) in an NO input condition and the File Memory Operation Flag (A34313) 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.

Examples

The following program section shows an example of sending a FINS command to another CPU Unit.

When CIO 000000 and A20207 (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 D00200 to D00211.

The MEMORY AREA READ command reads 10 words from D00010 to D00019. The response contains the 2-byte command code (0101), the 2-byte completion code, and then the 10 words of data, for a total of 12 words or 24 bytes.

The data will be retransmitted up to 3 times if a response is not received within ten seconds.

	15 87 0	
S: D00100	0 1 : 0 1	Command code: 0101 hexadecimal (MEMORY AREA READ)
S+1: D00101	8 2 : 0 0	D00010 (Data area = 82 hexadecimal, address = 000A00)
S+2: D00102	0 A : 0 0	
S+3: D00103	0 0 : 0 A	Number of words to read = 0A hexadecimal (10 decimal)

	15 87 0	
C: D00300	0 0 : 0 8	Bytes of command data: 0008 (8 decimal)
C+1: D00301	0 0 : 1 8	Bytes of response data: 0018 (24)
C+2: D00302	0 0 : 0 0	Transmit to the local network and the device itself
C+3: D00303	0 3 : 0 0	Node number 3, unit address 00 (CPU Unit)
C+4: D00304	0 7 : 0 3	Response requested, port number 7, 3 retries
C+5: D00305	0 0 : 6 4	Response monitoring time: 0064 hexadecimal (10 seconds)

The following program section shows an example of sending a FINS command to the local CPU Unit.

When CIO 000000 and A20207 (the Communications Port Enabled Flag for port 07) are ON and A34313 (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 D00100 to D00101. 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:	D00006	2 2 1 5	Command code: 2215 Hex (CREATE/DELETE DIRECTORY)
S+1:	D00007	8 0 0 0	Disk No.: 8000 Hex (Memory Card)
S+2:	D00008	0 0 0 0	Parameter: 0000 Hex (create directory)
S+3:	D00009	4 3 5 3	Subdirectory name: CS1□□□□□. □□□ (□= space)
S+4:	D00010	3 1 2 0	
S+5:	D00011	2 0 2 0	
S+6:	D00012	2 0 2 0	
S+7:	D00013	2 E 2 0	
S+8:	D00014	2 0 2 0	
S+9:	D00015	0 0 0 6	Directory name length: 0006 (6 characters)
S+10:	D00016	5 C 4 F	Absolute directory path: \OMRON
S+11:	D00017	4 D 5 2	
S+12:	D00018	4 F 4 E	

		15 8 7 0	
S:	D00000	0 0 1 A	Bytes of command data: 001A (26 decimal)
S+1:	D00001	0 0 0 4	Bytes of response data: 0004 (4)
S+2:	D00002	0 0 0 0	Destination network address: 00 Hex (local network)
S+3:	D00003	0 0 0 0	Destination unit address: 00 Hex, Destination node number: 00 Hex (CPU Unit at local node)
S+4:	D00004	0 7 0 0	Response requested, port number 7, 0 retries
S+5:	D00005	0 0 0 0	Response monitoring time: 0000 Hex (2 seconds)

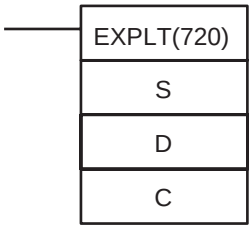
3-25-6 EXPLICIT MESSAGE SEND: EXPLT(720)

Purpose

Sends an explicit message with any service code.

This instruction is supported by only CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



S: First word of send message

D: First word of received message

C: First control word

Variations

Variations		
	Executed Each Cycle for ON Condition	EXPLT(720)
	Executed Once for Upward Differentiation	@EXPLT(720)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

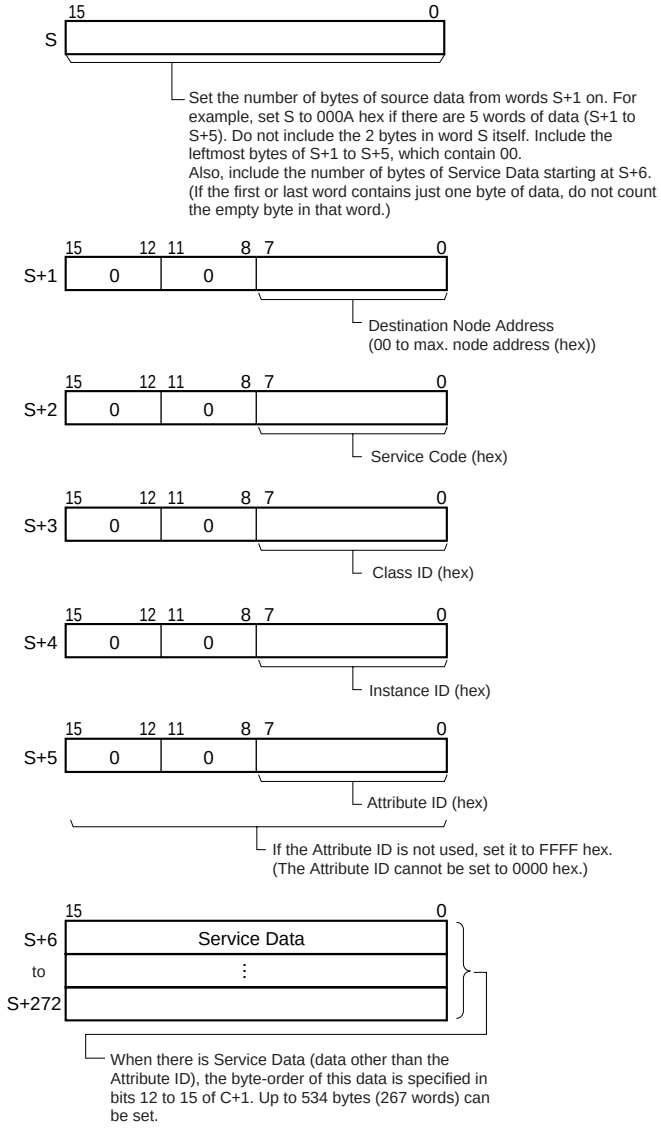
Applicable Program Areas

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

Operands

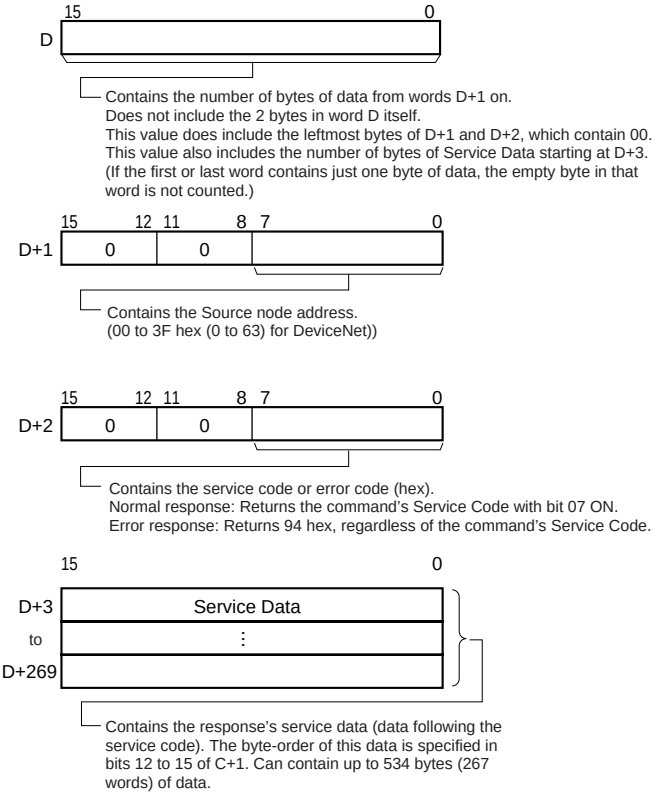
S: First word of send message

Specifies the first word of the send message (S to S+272 max.).



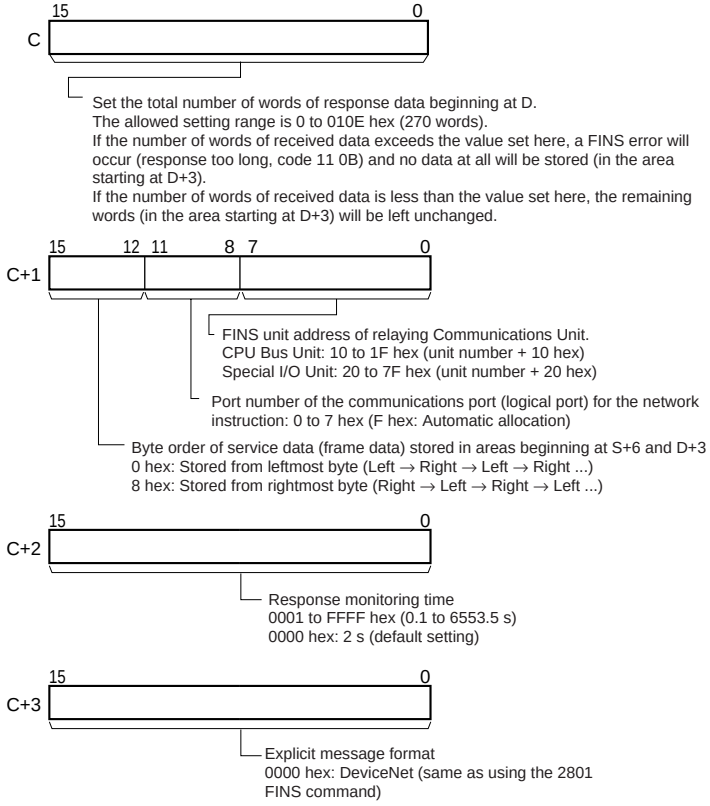
D: First word of received message

Specifies the first word of the received message (D to D+269 max.).



C: First control word

Specifies the first of four control words (C to C+3).



Operand Specifications

Area	S	D	C
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6140
Work Area	W000 to W511		W000 to W508
Holding Bit Area	H000 to H511		H000 to H508
Auxiliary Bit Area	A000 to A959	A448 to A959	A000 to A956
Timer Area	T0000 to T4095		T0000 to T4092
Counter Area	C0000 to C4095		C0000 to C4092
DM Area	D00000 to D32767		D00000 to D32764
EM Area without bank	E00000 to E32767		E00000 to E32764
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32764 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

Sends the explicit message command (stored in the range of words beginning at S+2) to the node address specified in S+1, via the Communications Unit with the FINS unit address specified in bits 00 to 07 of C+1. When the response to the explicit message is received, it is stored in the range of words beginning at D+2.

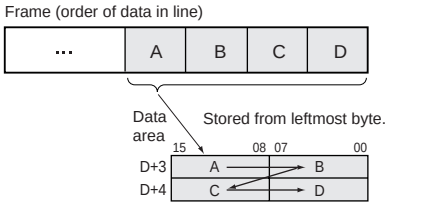
Number of Bytes Settings

The number of bytes of send data in S includes the 10 bytes in S+1 to S+5 as well as the number of bytes of service data beginning at S+6. (For example, if there is 1 byte of service data, there are 11 bytes of data all together, so S must be set to 000B hex.)

The number of bytes of received data in D includes the 4 bytes in D+1 and D+2 as well as the number of bytes of service data beginning at D+3. (For example, if there is 1 byte of service data, there are 5 bytes of data all together and D contains 0005 hex.)

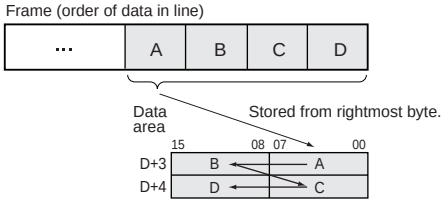
The setting in bits 12 to 15 of C+1 (0 or 8 hex) determines the byte-order of the service data stored at S+6 and D+3.

- Storing Data from the Leftmost Byte
  Set bits 12 to 15 of C+1 to 0 hex.



Note: A, B, C, and D represent bytes of data.

- Storing Data from the Rightmost Byte
  Set bits 12 to 15 of C+1 to 8 hex.



Note: A, B, C, and D represent bytes of data.

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

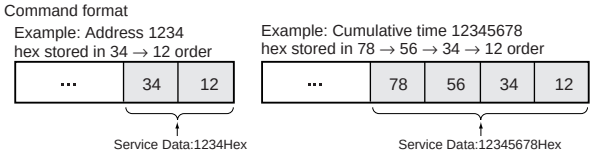
For details on the general operation of the explicit message instructions, refer to *3-25-2 About Explicit Message Instructions*.

The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A20200 to A20207	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	A21300 to A21307	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	A21900 to A21907	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

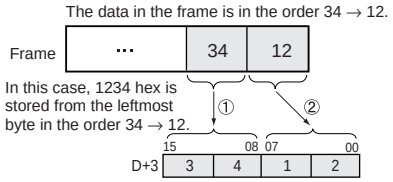
Precautions

Be sure that the order of bytes in the source data matches the order in the explicit message's frame (order of data in the line). For example, when the service data is in 2-byte or 4-byte units, the order of data in the frame is leftmost to rightmost order in 2-digit pairs, as shown in the following diagram.

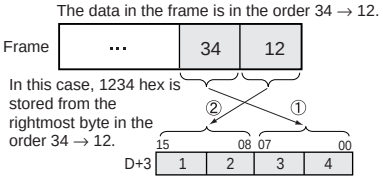


The following diagrams show how data is stored in the data areas when the service data is in 2-byte or 4-byte units.

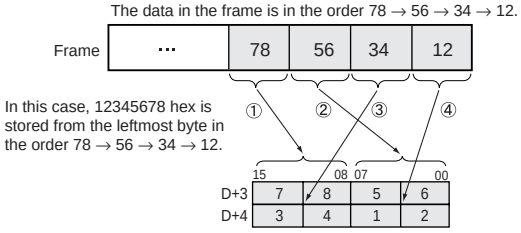
1. Data in 2-byte Units
   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 1234 hex in D+3



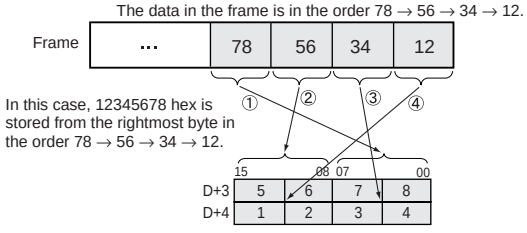
   - Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
     Example: Storing the value 1234 hex in D+3



2. Data in 4-byte Units
   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 12345678 hex in D+3 and D+4



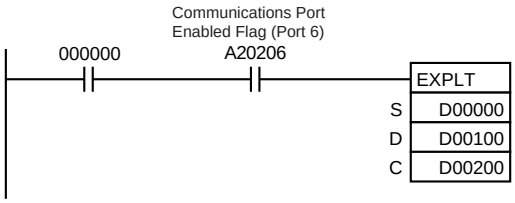
   - Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
     Example: Storing the value 12345678 hex in D+3 and D+4



Note The examples above only show the storage of received data in D+3, but send data is stored in S+6 in the same way.

Example

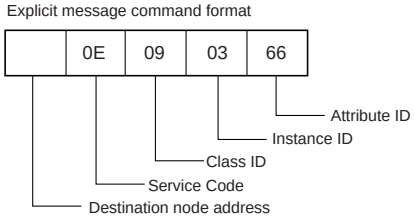
In this example, EXPLT(720) is used to read the total ON time or number of contact operations from a DRT2 Slave (I/O Terminal).

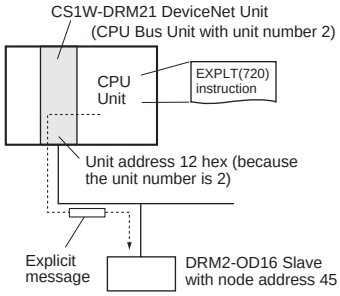


When CIO 000000 and A20206 (the Communications Port Enabled Flag for port 06) are ON, EXPLT(720) reads the Total ON Time (s) or Number of Contact Operations from a DRT2 Slave (I/O Terminal). In this case, the Total ON Time or Number of Contact Operations for input 3 are read.

Service Code = 0E hex, Class ID = 09 hex, Instance ID = 03 hex, and Attribute ID = 66 hex.

For example, a value of 2,752,039 s is returned as the response for the Total ON Time.





S:	D00000	0	0	0	A	Number of bytes of data: S+1 to S+5 = 5 words = 10 bytes = 0A hex
S+1:	D00001	0	0	2	D	Slave's node address = 45 = 2D hex
S+2:	D00002	0	0	0	E	Service Code = 0E hex
S+3:	D00003	0	0	0	9	Class ID = 09 hex
S+4:	D00004	0	0	0	3	Instance ID = 03 hex (Input 3)
S+5:	D00005	0	0	6	6	Attribute ID = 66 hex

D:	D00100	0	0	0	8	Contains 08 hex for 8 bytes of received data in response frame.
D+1:	D00101	0	0	2	D	Returns Slave's node address = 45 = 2D hex.
D+2:	D00102	0	0	8	E	Service Code = 8E hex (normal completion)
D+3:	D00103	2	7	F	E	Service Data = 0029FE27 hex (2,752,039 s decimal)
D+4:	D00104	2	9	0	0	

C:	D00200	0	0	0	4	Set 5 words = 0005 hex since there are 5 words in D to D+5.
C+1:	D00201	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:	D00202	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+3:	D00203	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)

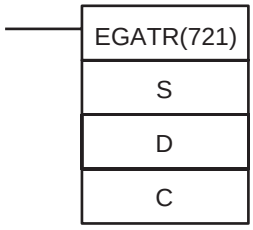
3-25-7 EXPLICIT GET ATTRIBUTE: EGATR(721)

Purpose

Sends an information/status read command in an explicit message (Get Attribute Single, Service Code: 0E hex).

This instruction is supported by only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



S: First word of send message

D: First word of received message

C: First control word

Variations

Variations	**Executed Each Cycle for ON Condition**	EGATR(721)
	Executed Once for Upward Differentiation	@EGATR(721)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

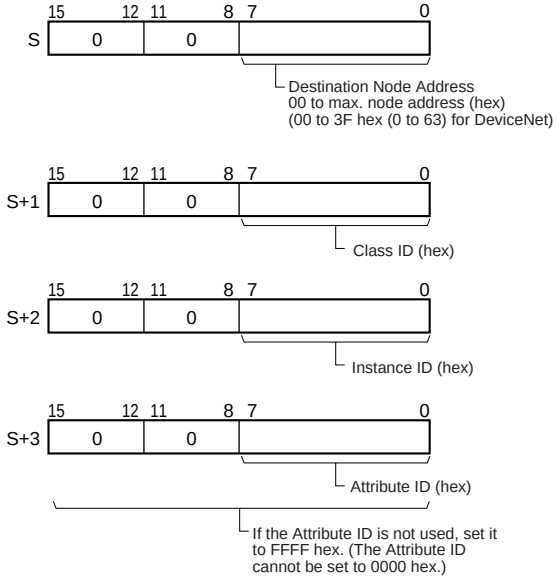
Applicable Program Areas

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

Operands

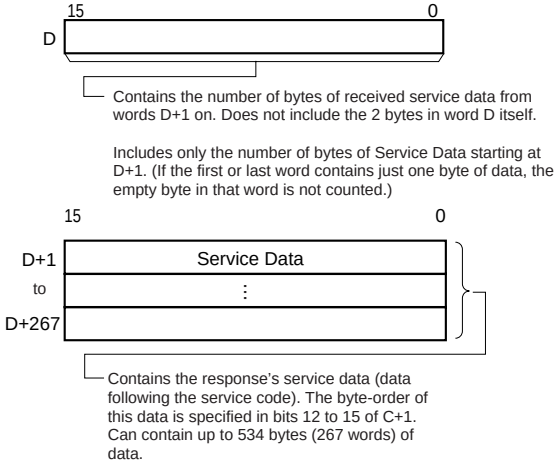
S: First word of send message

Specifies the first word of the send message (S to S+3).



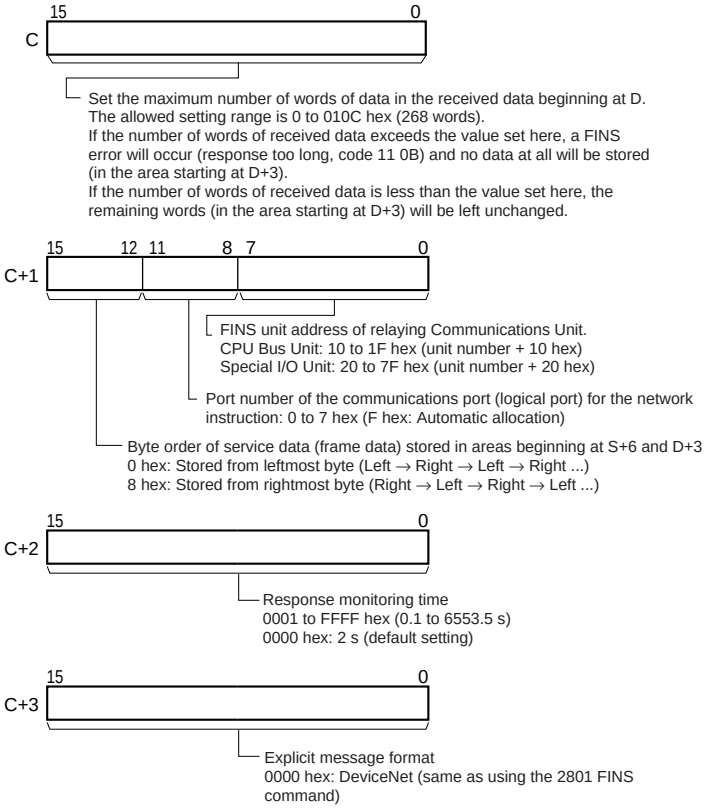
D: First word of received message

Specifies the first word of the received message (D to D+267 max.).



C: First control word

Specifies the first of four control words (C to C+3).



Operand Specifications

Area	S	D	C
CIO Area	CIO 0000 to CIO 6140	CIO 0000 to CIO 6143	CIO 0000 to CIO 6140
Work Area	W000 to W508	W000 to W511	W000 to W508
Holding Bit Area	H000 to H508	H000 to H511	H000 to H508
Auxiliary Bit Area	A000 to A956	A000 to A959	A000 to A956
Timer Area	T0000 to T4092	T0000 to T4095	T0000 to T4092

Area	S	D	C
Counter Area	C0000 to C4092	C0000 to C4095	C0000 to C4092
DM Area	D00000 to D32764	D00000 to D32767	D00000 to D32764
EM Area without bank	E00000 to E32764	E00000 to E32767	E00000 to E32764
EM Area with bank	En_00000 to En_32764 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32764 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

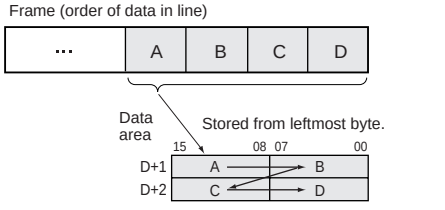
Description

Sends the “read information/status” explicit message command (stored in words S+1 to S+3) to the node address specified in S, via the Communications Unit with the FINS unit address specified in bits 00 to 07 of C+1.

When the response to the explicit message is received, the response’s service data (data following the service code) is stored in the range of words beginning at D+1.

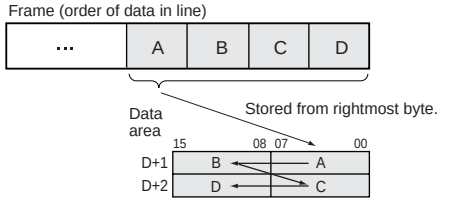
The number of bytes of received data indicated in D is the number of bytes of service data. (For example, if there is 1 byte of service data, D will contains 0001 hex. D will contain 0001 hex regardless of the byte order setting, i.e., whether the byte is stored in the rightmost or leftmost byte of D.)
The setting in bits 12 to 15 of C+1 (0 or 8 hex) determines the byte-order of the service data stored at S+6 and D+3.

- Storing Data from the Leftmost Byte
  Set bits 12 to 15 of C+1 to 0 hex.



Note: A, B, C, and D represent bytes of data.

- Storing Data from the Rightmost Byte
  Set bits 12 to 15 of C+1 to 8 hex.



Note: A, B, C, and D represent bytes of data.

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

For details on the general operation of the explicit message instructions, refer to *3-25-2 About Explicit Message Instructions*.

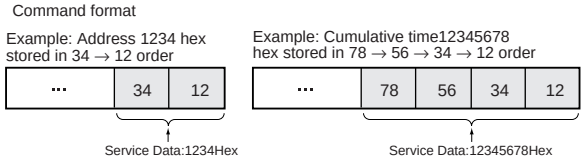
The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A20200 to A20207	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	A21300 to A21307	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of explicit message communications. The flags will be turned ON if the explicit message was not sent or the message was sent but an error response was returned. The flag status is retained until the next explicit message instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.

Name	Address	Operation
Communications Port Error Flag	A21900 to A21907	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

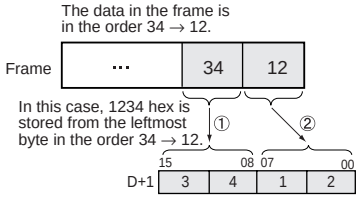
Precautions

Be sure that the order of bytes in the source data matches the order in the explicit message's frame (order of data in the line). For example, when the service data is in 2-byte or 4-byte units, the order of data in the frame is leftmost to rightmost order in 2-digit pairs, as shown in the following diagram.

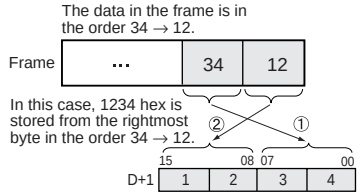


The following diagrams show how data is stored in the data areas when the service data is in 2-byte or 4-byte units.

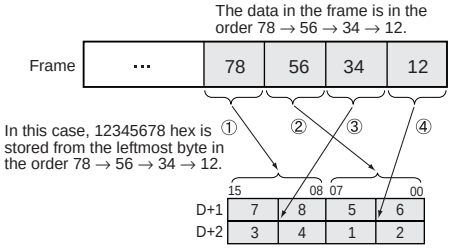
1. Data in 2-byte Units
   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 1234 hex in D+1



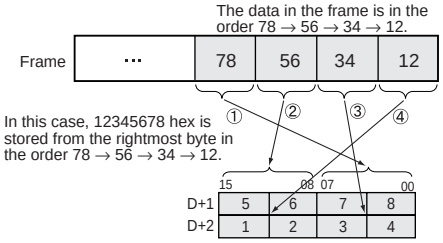
- Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
  Example: Storing the value 1234 hex in D+1



2. Data in 4-byte Units
   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 12345678 hex in D+1 and D+2

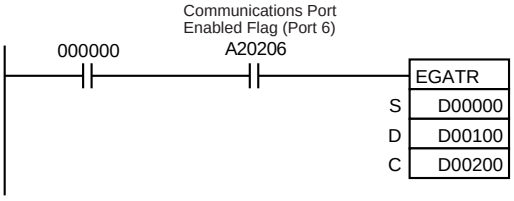


   - Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
     Example: Storing the value 12345678 hex in D+1 and D+2



Example

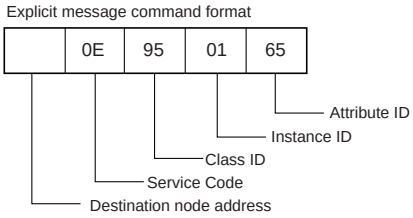
In this example, EGATR(721) is used to read the general status of a DRT2 Slave (I/O Terminal).

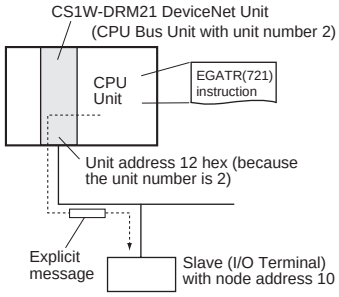


When CIO 000000 and A20206 (the Communications Port Enabled Flag for port 06) are ON, EGATR(721) reads the general status of the DRT2 Slave (I/O Terminal). In this case, the Total ON Time or Number of Contact Operations for input 3 are read.

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.

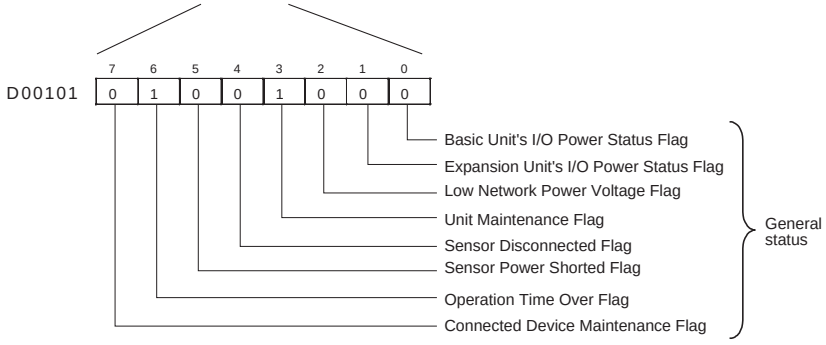




S:	D00000	0	0	0	A	Slave's node address = 10 = 0A hex
S+1:	D00001	0	0	9	5	Class ID = 95 hex
S+2:	D00002	0	0	0	1	Instance ID = 01 hex
S+3:	D00003	0	0	6	5	Attribute ID = 65 hex

C:	D00200	0	0	0	2	Set 2 words = 0002 hex since there are 2 words in D to D+1.
C+1:	D00201	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:	D00202	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+3:	D00203	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)

D:	D00100	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:	D00101	0	0	4	8	



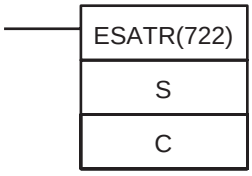
3-25-8 EXPLICIT SET ATTRIBUTE: ESATR(722)

Purpose

Sends an information write command in an explicit message (Set Attribute Single, Service Code: 10 hex).

This instruction is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



S: First word of send message

C: First control word

Variations

Variations	**Executed Each Cycle for ON Condition**	ESATR(722)
	Executed Once for Upward Differentiation	@ESATR(722)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

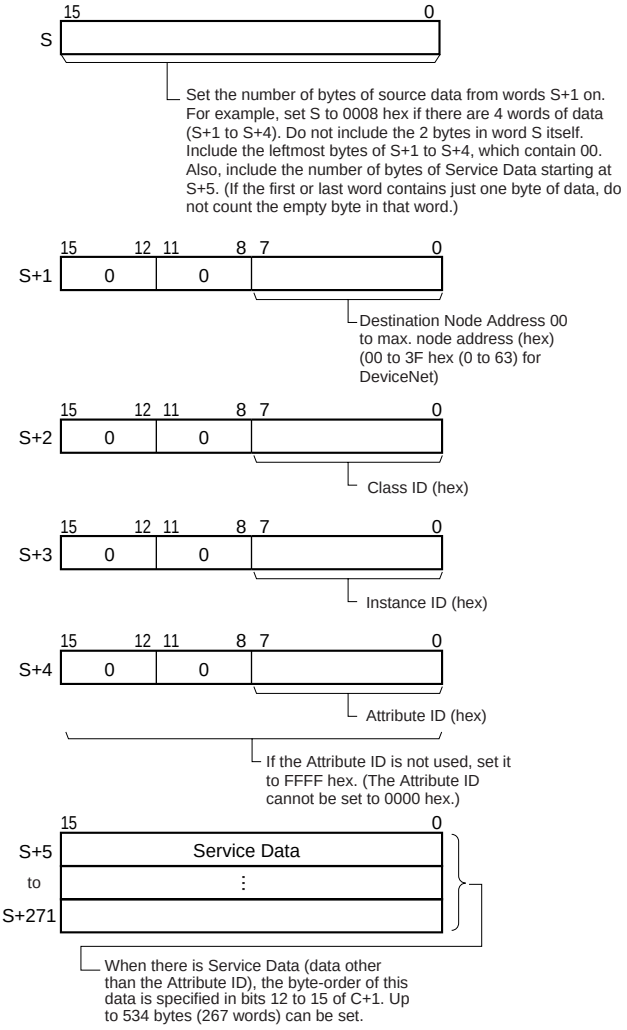
Applicable Program Areas

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

Operands

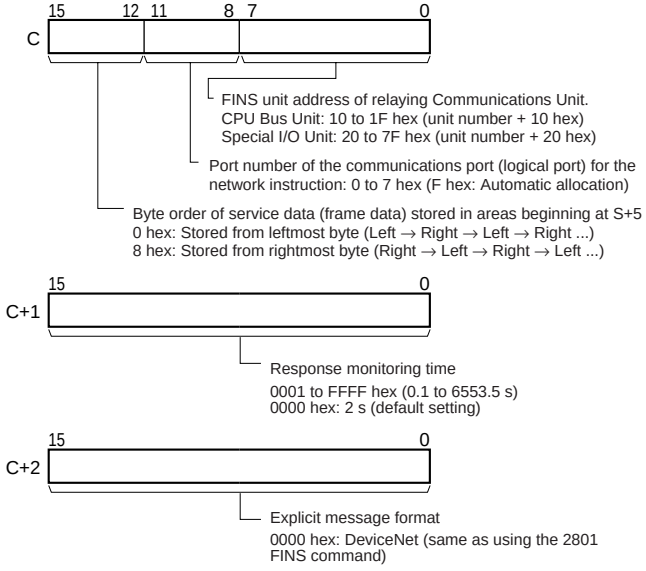
S: First word of send message

Specifies the first word of the send message (S to S+271 max.).



C: First control word

Specifies the first of three control words (C to C+2).



Operand Specifications

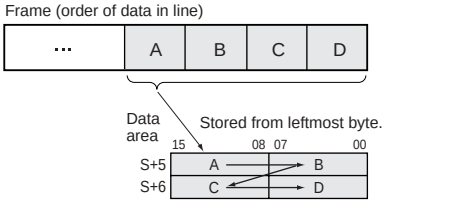
Area	S	C
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6141
Work Area	W000 to W511	W000 to W509
Holding Bit Area	H000 to H511	H000 to H509
Auxiliary Bit Area	A000 to A959	A000 to A957
Timer Area	T0000 to T4095	T0000 to T4093
Counter Area	C0000 to C4095	C0000 to C4093
DM Area	D00000 to D32767	D00000 to D32765
EM Area without bank	E00000 to E32767	E00000 to E32765
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32765 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

Sends the explicit message command with service code 10 hex (stored in the range of words beginning at S+2) to the node address specified in S+1, via the Communications Unit with the FINS unit address specified in bits 00 to 07 of C. When the response to the explicit message is received, it is stored in the range of words beginning at D+2.

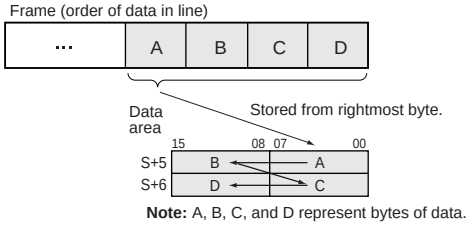
The setting in bits 12 to 15 of C (0 or 8 hex) determines the byte-order of the service data stored at S+5.

- Storing Data from the Leftmost Byte
  Set bits 12 to 15 of C to 0 hex.



Note: A, B, C, and D represent bytes of data.

- Storing Data from the Rightmost Byte
  Set bits 12 to 15 of C to 8 hex.



Note: A, B, C, and D represent bytes of data.

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

For details on the general operation of the explicit message instructions, refer to *3-25-2 About Explicit Message Instructions*.

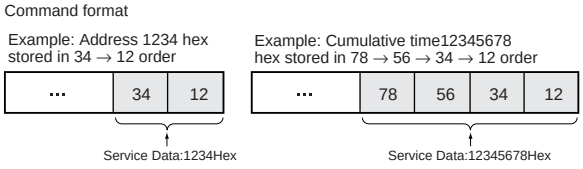
The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A20200 to A20207	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	A21300 to A21307	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of explicit message communications. The flags will be turned ON if the explicit message was not sent or the message was sent but an error response was returned. The flag status is retained until the next explicit message instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.

Name	Address	Operation
Communications Port Error Flag	A21900 to A21907	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

Precautions

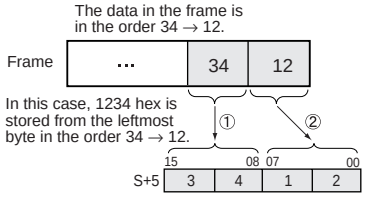
Be sure that the order of bytes in the source data matches the order in the explicit message's frame (order of data in the line). For example, when the service data is in 2-byte or 4-byte units, the order of data in the frame is leftmost to rightmost order in 2-digit pairs, as shown in the following diagram.



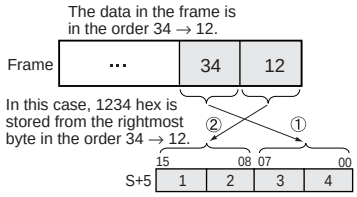
The following diagrams show how data is stored in the data areas when the service data is in 2-byte or 4-byte units.

1. Data in 2-byte Units

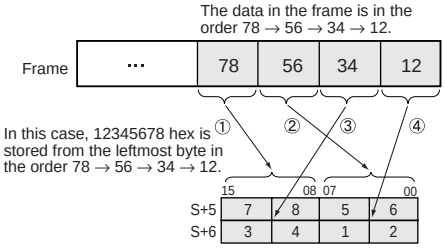
   • Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
   Example: Storing the value 1234 hex in S+5



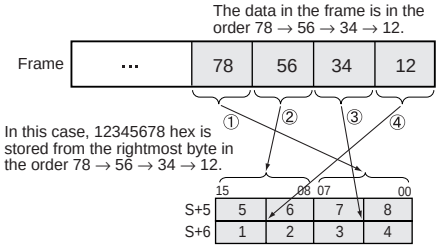
- Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
  Example: Storing the value 1234 hex in S+5



2. Data in 4-byte Units
   - Storing Data from the Leftmost Byte (Bits 12 to 15 of C = 0 hex)
     Example: Storing the value 12345678 hex in S+5 and S+6

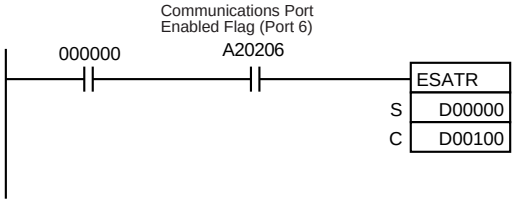


   - Storing Data from the Rightmost Byte (Bits 12 to 15 of C = 8 hex)
     Example: Storing the value 12345678 hex in S+5 and S+6



Example

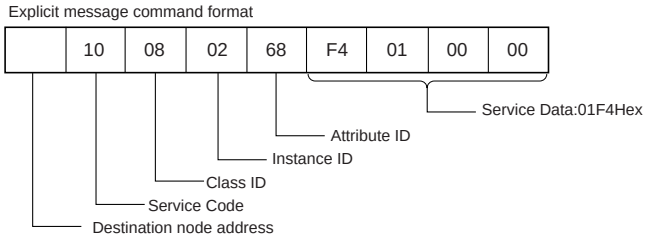
In this example, ESATR(722) is used to overwrite the Number of Contact Operations set value in a DRT2 Slave (I/O Terminal).

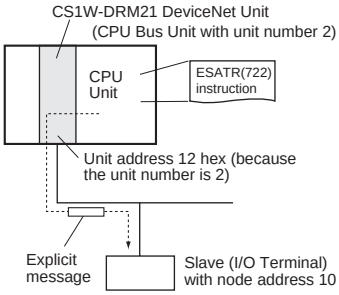


When CIO 000000 and A20206 (the Communications Port Enabled Flag for port 06) are ON, EXPLT(720) writes the Number of Contact Operations set value for input 2 in a DRT2 Slave (I/O Terminal).

(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:	D00000	0	0	0	C	Number of bytes of data: S+1 to S+6 = 6 words = 12 bytes = 0C hex
S:+1	D00001	0	0	0	A	Slave's node address = 10 = 0A hex
S+2:	D00002	0	0	0	8	Class ID = 08 hex
S+3:	D00003	0	0	0	2	Instance ID = 02 hex
S+4:	D00004	0	0	6	8	Attribute ID = 68 hex
S+5:	D00005	0	1	F	4	Service Data = F401 hex
S+6:	D00006	0	0	0	0	

C:	D00201	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:	D00202	0	0	0	0	Response monitoring time = 0000 hex (2 s)
C+2:	D00203	0	0	0	0	Explicit format type = 0000 hex (DeviceNet format)

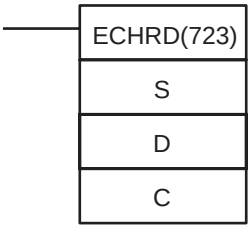
3-25-9 EXPLICIT WORD READ: ECHRD(723)

Purpose

Reads data to the local CPU Unit from another CPU Unit in the network. (The remote CPU Unit must support explicit messages.)

This instruction is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



S: First source word in remote CPU Unit

D: First destination word in local CPU Unit

C: First control word

Variations

Variations	**Executed Each Cycle for ON Condition**	ECHRD(723)
	Executed Once for Upward Differentiation	@ECHRD(723)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

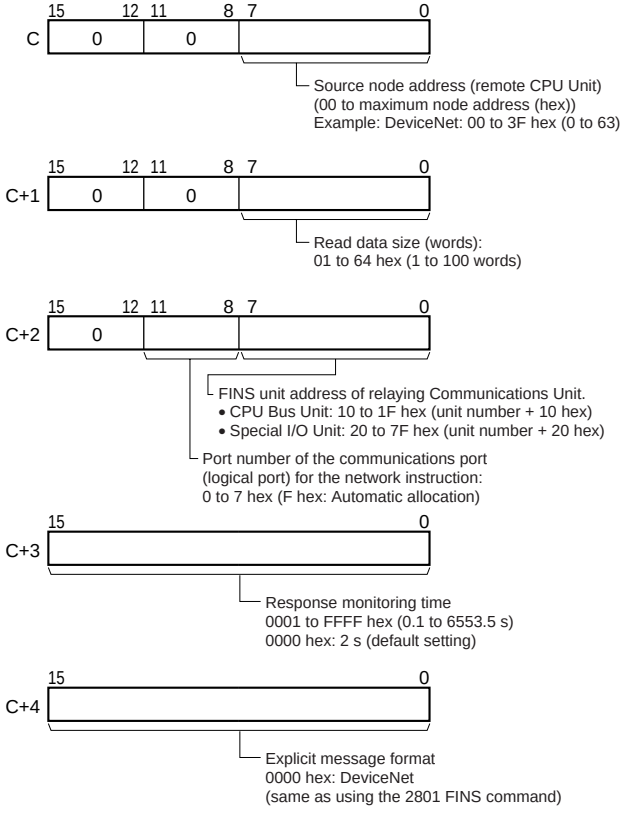
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: First Source Word in Remote CPU Unit
Specifies the leading word address containing the data to be read from the remote CPU Unit.

D: First Destination Word in Local CPU Unit
Specifies the leading word address where the read data will be stored in the local CPU Unit.

C: First Control Word
Specifies the first of five control words (C to C+4).



Operand Specifications

Area	S	D	C
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6139
Work Area	W000 to W511		W000 to W507
Holding Bit Area	H000 to H511		H000 to H507
Auxiliary Bit Area	A000 to A959	A448 to A959	A000 to A955
Timer Area	T0000 to T4095		T0000 to T4091
Counter Area	C0000 to C4095		C0000 to C4091
DM Area	D00000 to D32767		D00000 to D32763
EM Area without bank	E00000 to E32767		E00000 to E32763
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32763 (n = 0 to C)

Area	S	D	C
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

Reads the specified number of words from the first read word (specified in S) in the remote CPU Unit with the node address specified in C, and stores the data in the local CPU Unit memory words beginning at D.

Note ECHRD(723) sends an explicit message with the Service Code 1C hex (Byte Data Read).

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

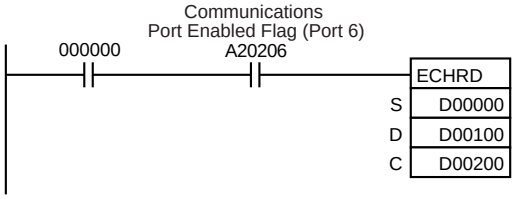
For details on the general operation of the network instructions, refer to *3-25-2 About Explicit Message Instructions*.

The following table shows relevant bits and flags in the Auxiliary Area.

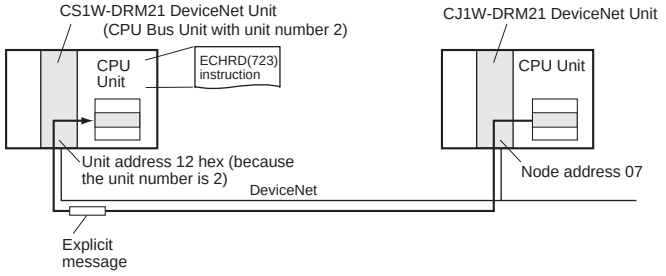
Name	Address	Operation
Communications Port Enabled Flag	A20200 to A20207	These flags are turned ON to indicate that network instructions, including PMCR(260) may be executed for the corresponding ports (00 to 07). A flag is turned OFF when a network instruction is being executed for the corresponding port and turned ON again when the instruction is completed.
Explicit Communications Error Flag	A21300 to A21307	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	A21900 to A21907	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

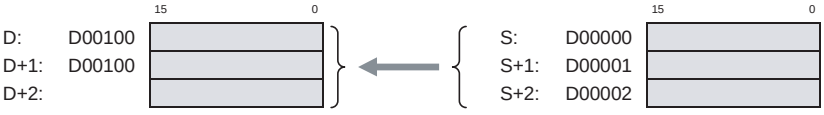
Example

In this example, ECHRD(723) is used to read the I/O memory of 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 000000 and A20206 (the Communications Port Enabled Flag for port 06) are ON, ECHRD(723) reads D00000 to D00002 from the I/O memory of the CJ-series CPU Unit with node address 07 on the DeviceNet Network and stores the data in D00100 to D00102 of the local CPU Unit.





		15–8	7–0	
C:	D00200	0 0	0 7	Node address of remote CPU Unit to be read = 07 hex (node 07)
C+1:	D00201	0 0	0 3	Read data size (number of words) = 3 hex
C+2:	D00202	0 6	1 2	Communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+3:	D00203	0 0	0 0	Response monitoring time = 0000 hex (2 s)
C+4:	D00204	0 0	0 0	Explicit format type = 0000 hex (DeviceNet format)

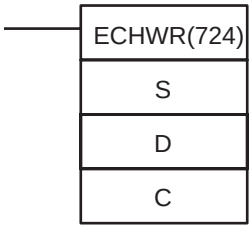
3-25-10 EXPLICIT WORD WRITE: ECHWR(724)

Purpose

Writes data from the local CPU Unit to another CPU Unit in the network. (The remote CPU Unit must support explicit messages.)

This instruction is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.

Ladder Symbol



S: First source word in local CPU Unit

D: First destination word in remote CPU Unit

C: First control word

Variations

Variations		
	Executed Each Cycle for ON Condition	ECHWR(724)
	Executed Once for Upward Differentiation	@ECHWR(724)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

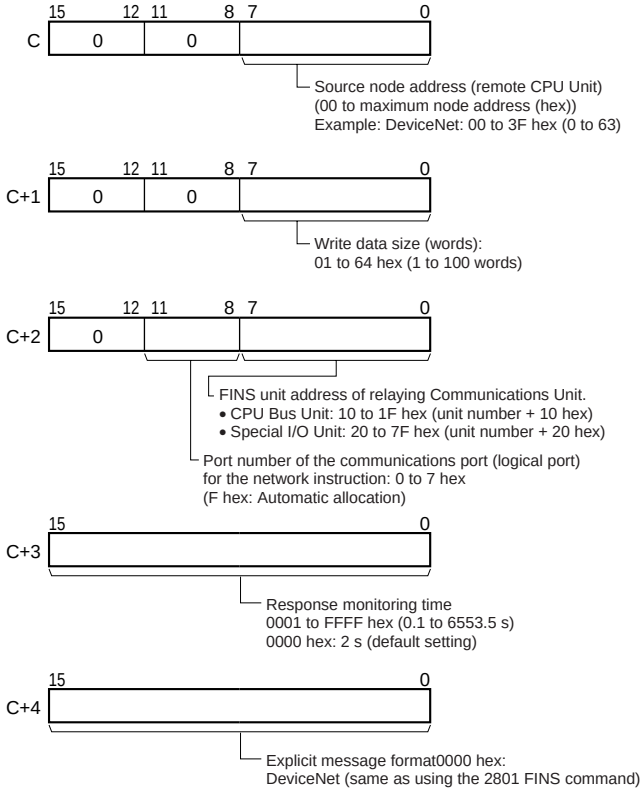
Operands

S: First Source Word in Local CPU Unit

Specifies the leading word address in the local CPU Unit containing the write data.

D: First Destination Word in Remote CPU Unit
Specifies the leading word address of the write destination in the remote CPU Unit.

C: First Control Word
Specifies the first of five control words (C to C+4).



Operand Specifications

Area	S	D	C
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6139
Work Area	W000 to W511		W000 to W507
Holding Bit Area	H000 to H511		H000 to H507
Auxiliary Bit Area	A000 to A959	A448 to A959	A000 to A955
Timer Area	T0000 to T4095		T0000 to T4091
Counter Area	C0000 to C4095		C0000 to C4091
DM Area	D00000 to D32767		D00000 to D32763
EM Area without bank	E00000 to E32767		E00000 to E32763
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		En_00000 to En_32763 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		

Area	S	D	C
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

Writes the specified number of words beginning at S from the local CPU Unit to the write destination beginning at D in the remote CPU Unit with the node address specified in C.

Note ECHWR(724) sends an explicit message with the Service Code 1E hex (Byte Data Write).

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag is OFF for the communications port number specified in C. OFF in all other cases.

The corresponding Explicit Communications Error Flag will be OFF if the instruction ended normally or ON if an error occurred.

If an error occurred (corresponding flag in A213 ON), the corresponding Communications Port Error Flag can be used to determine whether the explicit message itself was not sent (corresponding flag in A219 ON) or that the message was sent but there was an error in the message (corresponding flag in A219 OFF).

The corresponding Communications Port Completion Code (A203 to A210) will be 0000 hex if the instruction ended normally, an explicit message error code if an explicit messaging error occurred, or a FINS error code if a FINS error occurred.

For details on the general operation of the explicit message instructions, refer to *3-25-2 About Explicit Message Instructions*.

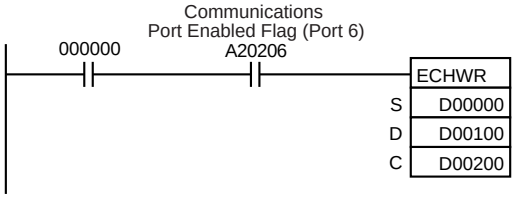
The following table shows relevant bits and flags in the Auxiliary Area.

Name	Address	Operation
Communications Port Enabled Flag	A20200 to A20207	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	A21300 to A21307	These flags are turned ON to indicate that an error has occurred at the corresponding ports (00 to 07) during execution of explicit message communications. The flags will be turned ON if the explicit message was not sent or the message was sent but an error response was returned. The flag status is retained until the next explicit message instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.

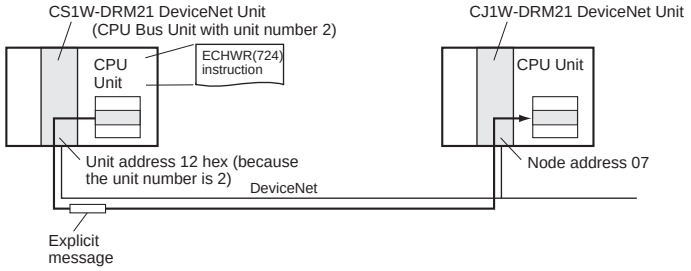
Name	Address	Operation
Communications Port Error Flag	A21900 to A21907	These flags are turned ON to indicate that the explicit message itself was not sent from the corresponding ports (00 to 07) during execution of an explicit message instruction. The flag status is retained until the next network instruction is executed. The flag will be turned OFF when the next instruction is executed even if an error occurred previously.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding ports (00 to 07) following execution of a network instruction. The corresponding word will contain 0000 while the Explicit Communications Error Flag is OFF. The corresponding word will contain a FINS error code when that port's Explicit Communications Error Flag and Communications Port Error Flag are both ON. The corresponding word will contain the appropriate explicit message error code when that port's Explicit Communications Error Flag is ON and the Communications Port Error Flag is OFF. The corresponding word will contain 0000 while the network instruction is being executed and the completion code will be written when the instruction is completed. These words are cleared when program execution begins.

Example

In this example, ECHWR(724) is used to write data from the I/O memory of the local CPU Unit to the I/O memory of a CJ-series CPU Unit on the DeviceNet network.



When CIO 000000 and A20206 (the Communications Port Enabled Flag for port 06) are ON, ECHWR(724) reads D00000 to D00002 from the I/O memory of the local CPU Unit and stores the data in D00100 to D00102 of the CJ-series CPU Unit with node address 07 on the DeviceNet Network



		15 8	7 0	
C:	D00200	0 0	0 7	Node address of remote CPU Unit to be written to = 07 hex (node 07)
C+1:	D00201	0 0	0 3	Write data size (number of words) = 3 hex
C+2:	D00202	0 6	1 2	Communications port = 6 hex (port 6), and the DeviceNet Unit's unit address = 12 hex
C+3:	D00203	0 0	0 0	Response monitoring time = 0000 hex (2 s)
C+4:	D00204	0 0	0 0	Explicit format type = 0000 hex (DeviceNet format)

3-26 File Memory Instructions

This section describes instructions used with file memory (EM Area or Memory Cards).

Note File memory can also be manipulated by executing CMND(490) to send a FINS command to the local CPU Unit. Refer to the *CS/CJ-series PLC Operation Manual* for details.

Instruction	Mnemonic	Function code	Page
READ DATA FILE	FREAD	700	1099
WRITE DATA FILE	FWRIT	701	1106
WRITE TEXT FILE	TWRIT	704	1113

3-26-1 Precautions when Using Memory Cards

Confirm the following items before using a Memory Card.

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.

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 *3-25-5 DELIVER COMMAND: CMND(490)* for a specific example using CMND(490).

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.

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.

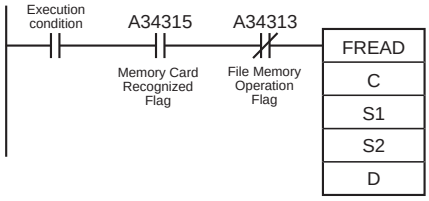
The allocation unit size of the Memory Card can be checked from a DOS prompt using CHKDSK. The specific procedure is omitted here. Refer to general computer references for more information on allocation unit sizes.

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.

1,2,3...

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 (A34315) as an input condition, as shown below.



Note The structure of data files is 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.

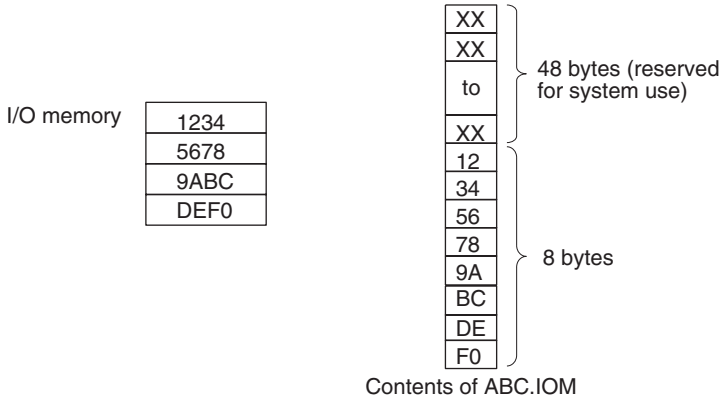
TWRIT(704)

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.

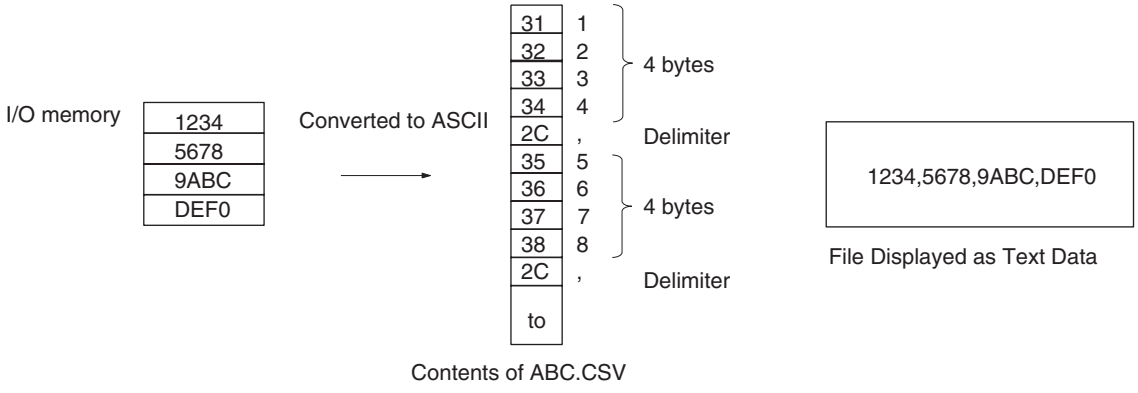
CMND(490)

CMND(490) can be used to format files, delete files, copy files, and change file names by sending FINS commands for Memory Card operations. For details, refer to *Section 5 File Memory Functions* in the *SYSMAC CS/CJ Series Programmable Controllers Programming Manual* (W394).

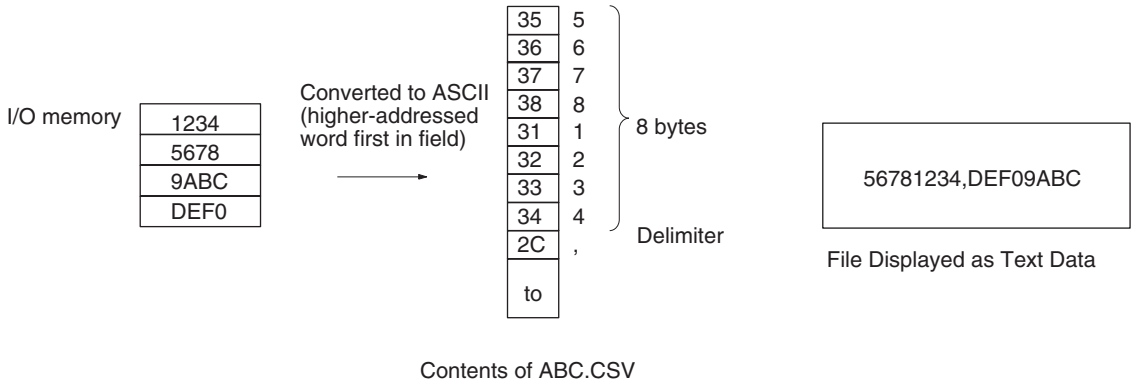
For binary format (.IOM), the data will be as follows when 1234 hex, 5678 hex, 9ABC hex, and DEF0 hex are stored in the file ABC.IOM (although the user does not normally need to be concerned with this structure):



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	A34300 to A34302	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	A34307	ON when the Memory Card is not formatted or a formatting error has occurred.
Memory Card Detected Flag (version 1 (-V1) or higher only)	A34315	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	A34308	ON when an error occurred when writing to the file. ON when the file being written is write-protected.
File Write Impossible Flag	A34309	ON when the data could not be written because there was insufficient free memory.
File Read Error Flag	A34310	ON when a file could not be read because its data was corrupted or if it contains the wrong data type.
File Missing Flag	A34311	ON when data could not be read because the specified file does not exist.
File Memory Operation Flag	A34313	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.

Name	Address	Operation
Accessing File Flag	A34314	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

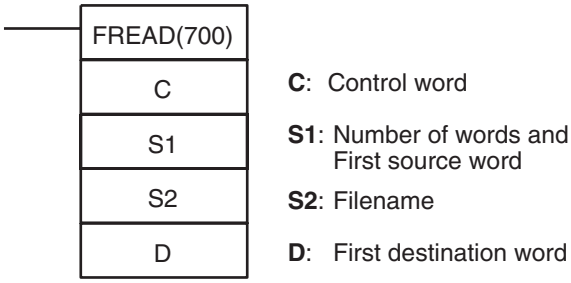
Name	Address	Operation
EM File Memory Format Error Flag	A34306	ON when there is a format error in the starting bank of EM file memory.
EM File Format Starting Bank	A344	Contains the starting bank number of the EM Area that has been formatted for use as EM file memory. Contains FFFF when none of the EM Area has been formatted. To convert the EM Area for use as file memory, the PLC Setup's EM File Memory setting must be set to 1 and the EM File Memory Starting Bank (0 to C) must be set. All EM banks from the starting bank to the last bank will then be formatted for use as file memory.

3-26-2 READ DATA FILE: FREAD(700)

Purpose

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.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	FREAD(700)
	Executed Once for Upward Differentiation	@FREAD(700)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

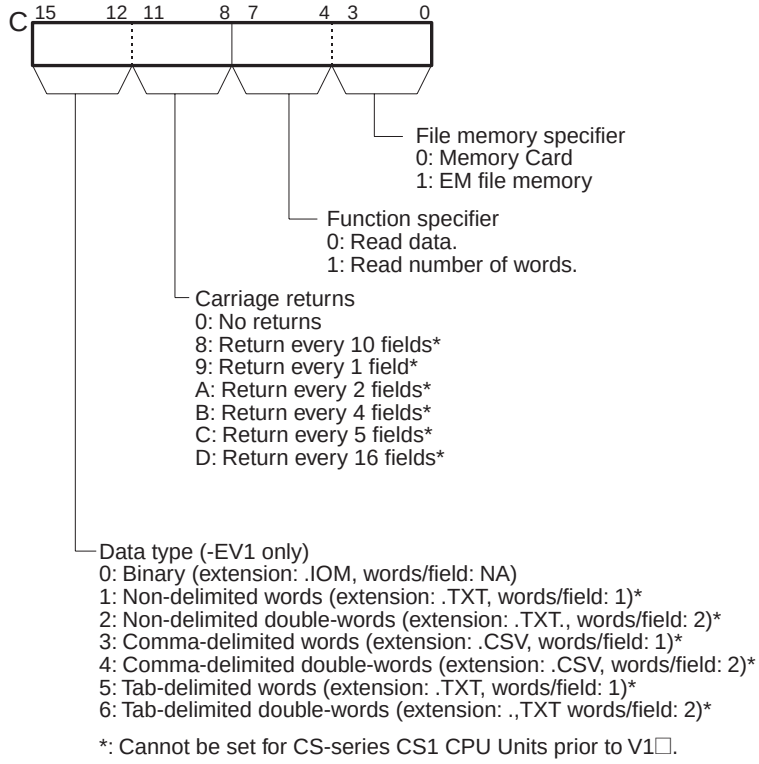
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word

As shown in the following diagram, the first digit indicates whether the source

file is in the Memory Card or EM file memory, the second digit of the control word indicates whether the actual data or the number of words of data is to be read, the third digits indicates the presence of carriage returns, and the fourth digit indicates the data type.



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 D00001 and 5678 in D00000.

S1 and S1+1: Number of Read Items

The 8-digit hexadecimal value in S1 and S1+1 specifies how many words or fields to read from file memory. 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.

S1+1 contains the leftmost 4 digits and S1 contains 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

Data type	Bits 12 to 15 of C	Contents of S1 and S1+1
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.

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

1. 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).
2. S1 to S1+3 must be in the same data area.
3. S1 to S1+3 are used only when reading data.
4. If the specified starting word exceeds the number of words in the data file, the File Read Error Flag (A34310) will be turned ON and the file data will not be read.

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. Do not include the filename extension; the .IOM extension will be added automatically.

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.

Note

1. Be sure that the character string containing the path name and file name does not exceed the end of the data area.
2. If the specified file or directory does not exist, the File Missing Flag (A34311) will be turned ON and the file data will not be read.

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	

D: First Destination Word

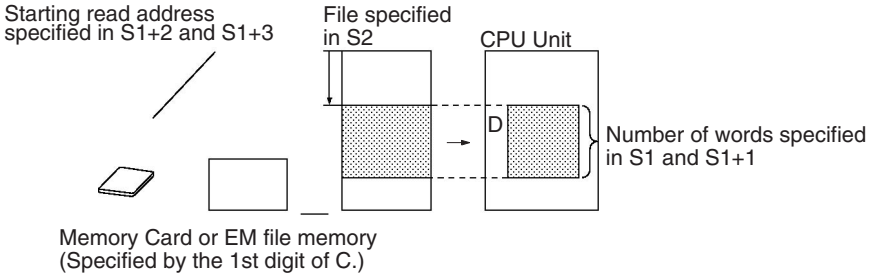
When data is being read, D specifies the starting address where the data read from file memory will be stored.

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.

Description

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.

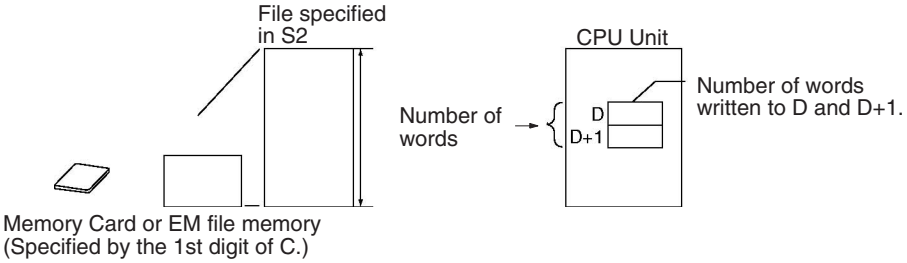


Note Data is stored in order by absolute internal memory addresses, so the output data will overwrite data in the next data area if it exceeds the capacity of the data area specified in D. See *Precautions* for more details.

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.

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.



Operand Specifications

Area	C	S1	S2	D
CIO Area	CIO 0000 to CIO6143	CIO 0000 to CIO 6140	CIO 0000 to CIO 6143	
Work Area	W000 to W511	W000 to W508	W000 to W511	
Holding Bit Area	H000 to H511	H000 to 508	H000 to W511	
Auxiliary Bit Area	A000 to A959	A000 to A444 A448 to A956	A000 to A447 A448 to A959	A448 to A959
Timer Area	T0000 to T4095	T0000 to T4092	T0000 to T4095	
Counter Area	C0000 to C4095	C0000 to C4092	C0000 to C4095	
DM Area	D00000 to D32767	D00000 to D32764	D00000 to D32767	
EM Area without bank	E00000 to E32767	E00000 to E32764	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32764 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	–	@D00000 to @D32767 @E00000 to @E32767 @En_00000 to @En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	–	∗D00000 to *D32767 ∗E00000 to *E32767 ∗En_00000 to *En_32767 (n = 0 to C)		
Constants	Specified values only	–		
Data Registers	–			
Index Registers	–			
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

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.

The following table shows relevant flags in the Auxiliary Area.

Name	Address	Operation
Memory Card Type	A34300 to A34302	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	A34307	ON when the Memory Card is not formatted or a formatting error has occurred.
File Read Error Flag	A34310	ON when a file could not be read because its data was corrupted or if it contains the wrong data type.
File Missing Flag	A34311	ON when data could not be read because the specified file does not exist.
File Memory Operation Flag	A34313	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	A34314	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.
Memory Card Detected Flag	A34315	ON when a Memory Card has been detected.
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. To convert the EM Area for use as file memory, the PLC Setup's EM File Memory setting must be set to 1 and the EM File Memory Starting Bank (0 to C) must be set. All EM banks from the starting bank to the last bank will then be formatted for use as file memory.

Name	Address	Operation
EM File Memory Format Error Flag	A34306	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.

Precautions

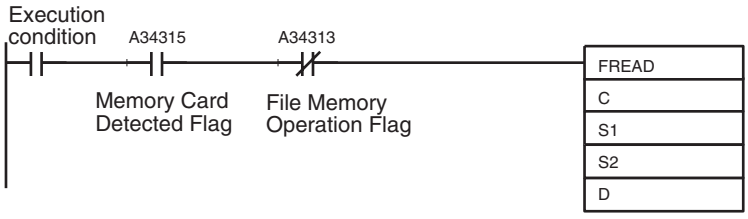
During normal instruction processing, FREAD(700) is used only to start reading file memory. The instruction execution times given toward the end of this manual are thus the times required to start reading, not to complete it. Actual reading (transfer) is performed by the file access processing in peripheral servicing. Therefore, once FREAD(700) has been executed, reading is continuously executed even if the execution condition is OFF in following cycles. When transfer has been completed, the File Memory Operation Flag (A34313) will turn OFF. This flag can be used for exclusive control of file memory instructions.

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.

The File Memory Operation Flag (A34313) 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.

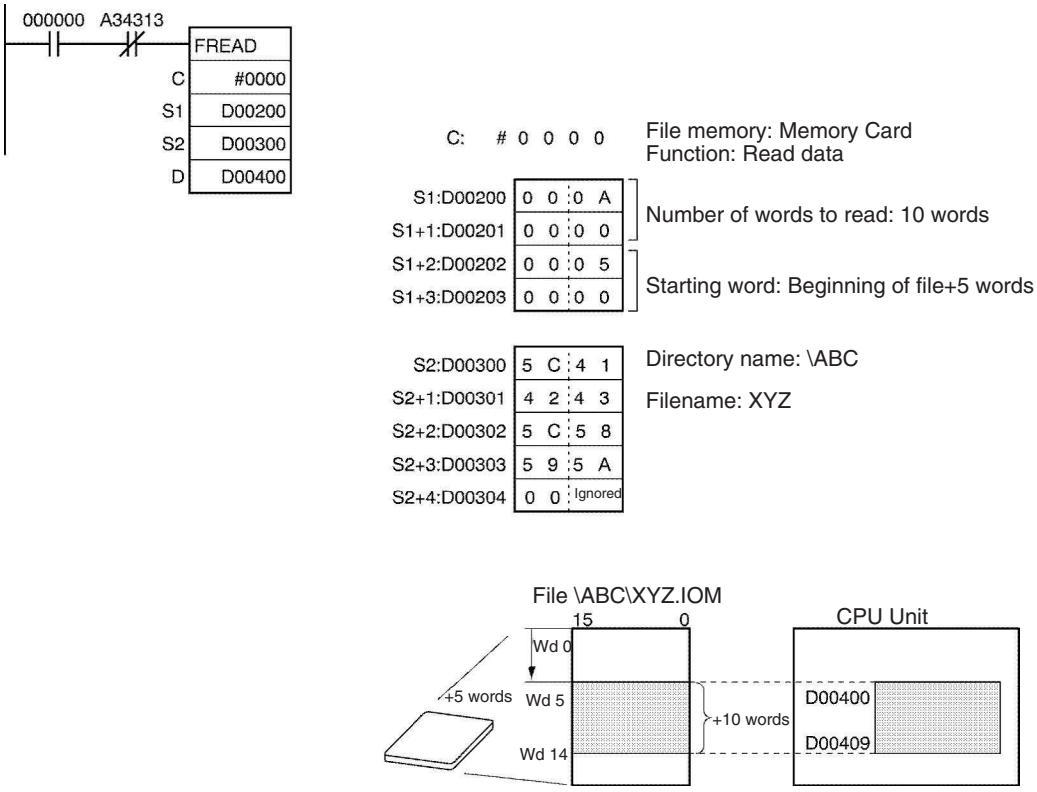
The File Read Error Flag (A34310) will be turned ON and the instruction will not be executed if the specified file contains the wrong data type or the file data is corrupted. For text or CSV files, the character code must be hexadecimal data and delimiters must be every 4 digits for word data and every 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 (A34315) in a NO input condition as shown below to be sure that the Memory Card has been detected.



Examples

When CIO 000000 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 D00400 through D00409.

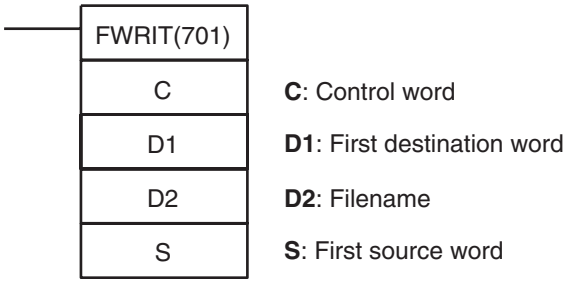


3-26-3 WRITE DATA FILE: FWRIT(701)

Purpose

Overwrites or appends data in the specified data file in file memory with the specified data from the data area in the CPU Unit. If the specified file does not exist, a new file is created with that filename. Data can be written as binary, text, or CSV format data.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	FWRIT(701)
	Executed Once for Upward Differentiation	@FWRIT(701)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

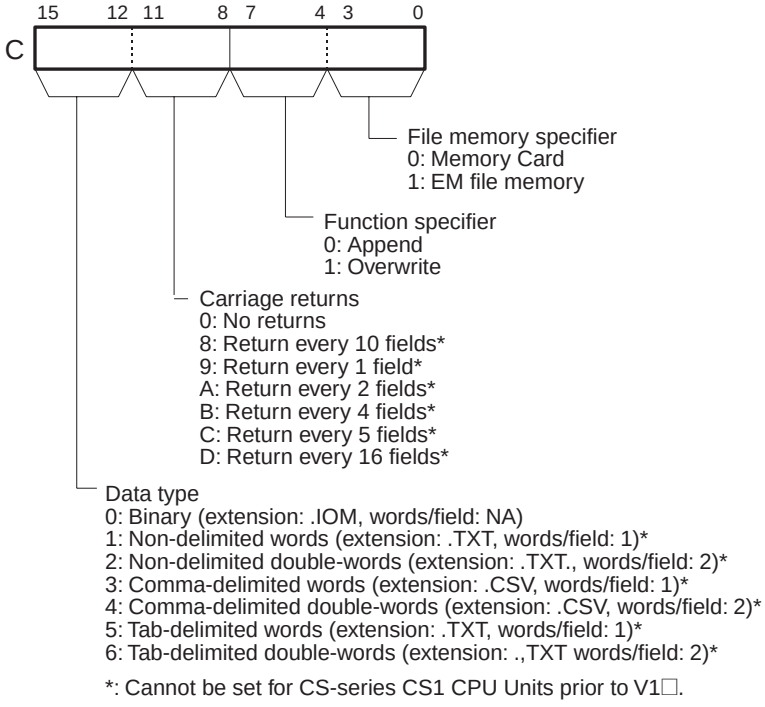
Applicable Program Areas

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

Operands

C: Control Word

As shown in the following diagram, the third digit of the control word indicates whether to append or overwrite data in the data file and the fourth digit indicates whether the destination file is in the Memory Card or EM file memory.



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.
3. 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. (The code for a comma is added for comma-delimiting and the code for a tab is added for tab-delimiting.)
4. If non-delimited words or double-words are specified, the data for all fields is written continuously without any delimiters.
5. 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.

D1 and D1+1: Number of Write Items

The 8-digit hexadecimal value in D1 and D1+1 specifies how many words or fields to write to file memory.

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

1. 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).
2. D1 to D1+3 must be in the same data area.
3. If the specified starting word exceeds the number of words in the data file, the File Write Error Flag (A34308) will be turned ON and the data will not be written.

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. Do not include the filename extension; the .IOM, .TXT, or .CSV extension is added automatically.

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.

Note

1. Be sure that the character string containing the pathname and filename does not exceed the end of the data area.
2. If the specified directory does not exist, the File Missing Flag (A34311) will be turned ON and the file data will not be written.

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	5C	41
D2+1	'B'	'C'	→ D2+1	42	43
D2+2	'\'	'X'	D2+2	5C	58
D2+3	'Y'	'Z'	D2+3	59	5A
D2+4	NUL		→ D2+4	00	

For information on creating directories from the ladder program, refer to *Section 5 File Memory Functions* in the *SYSMAC CS/CJ Series Programmable Controllers Programming Manual* (W394).

S: First Source Word

S specifies the starting address containing the data that will be written to the file memory. Data is read by absolute PLC memory addresses, so FWRIT(701) will continue reading source data from the next data area if the number of words being read exceeds the end of the data area specified in S.

Description

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. When transfer has been completed, the File Memory Operation Flag (A34313) will turn OFF. This flag can be used for exclusive control of file memory instructions.

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.

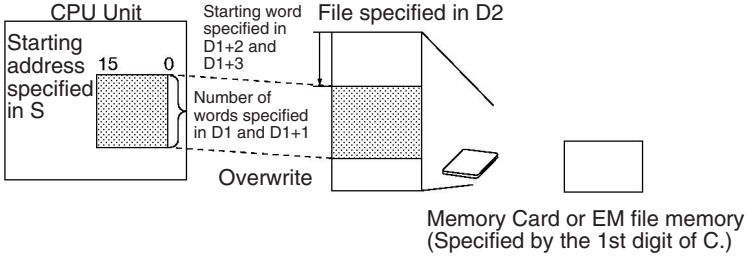
The source data is read from absolute internal memory addresses in RAM, so the entire block of data will be read even if the data spans two or more data areas. For example, if the first destination address is in the Work Area but the amount of data exceeds the capacity of this area, FWRIT(701) will continue reading data at the beginning of the next area (in this case, the Timer Area). Refer to *Appendix D* in the *CS/CJ-series Programmable Controllers Operation Manual* (W339) for a memory map showing the location of data areas in RAM.

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.

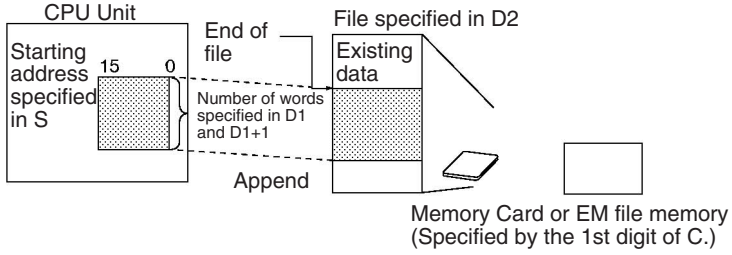
Overwriting Data in an Existing File (Third Digit of C=1)

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.



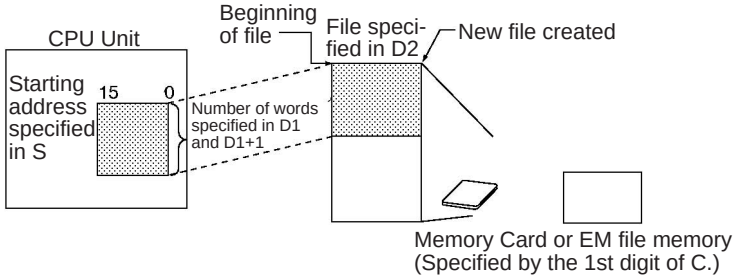
Appending Data to an Existing File (Third Digit of C=0)

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.



Operand Specifications

Area	C	D1	D2	S
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6140	CIO 0000 to CIO 6143	
Work Area	W000 to W511	W000 to W508	W000 to W511	
Holding Bit Area	H000 to H511	H000 to 508	H000 to H511	

Area	C	D1	D2	S
Auxiliary Bit Area	A000 to A959	A000 to A444 A448 to A956	A000 to A447 A448 to A959	
Timer Area	T0000 to T4095	T0000 to T4092	T0000 to T4095	
Counter Area	C0000 to C4095	C0000 to C4092	C0000 to C4095	
DM Area	D00000 to D32767	D00000 to D32764	D00000 to D32767	
EM Area without bank	E00000 to E32767	E00000 to E32764	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	En_00000 to En_32764 (n = 0 to C)	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	–	@D00000 to @D32767 @E00000 to @E32767 @En_00000 to @En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	–	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	Specified values only	–		
Data Registers	–			
Index Registers	–			
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

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.

The following table shows relevant flags in the Auxiliary Area.

Name	Address	Operation
Memory Card Type	A34300 to A34302	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	A34307	ON when the Memory Card is not formatted or a formatting error has occurred.
File Write Error Flag	A34308	ON when an error occurred when writing to the file.
File Write Impossible Flag	A34309	ON when the data could not be written because the file was write-protected or there was insufficient free memory.
No File Flag	A34311	ON when the specified directory does not exist when writing a file.
File Memory Operation Flag	A34313	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	A34314	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.
Memory Card Detected Flag	A34315	ON when a Memory Card has been detected.
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. To convert the EM Area for use as file memory, the PLC Setup's EM File Memory setting must be set to 1 and the EM File Memory Starting Bank (0 to C) must be set. All EM banks from the starting bank to the last bank will then be formatted for use as file memory.
EM File Memory Format Error Flag	A34306	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.

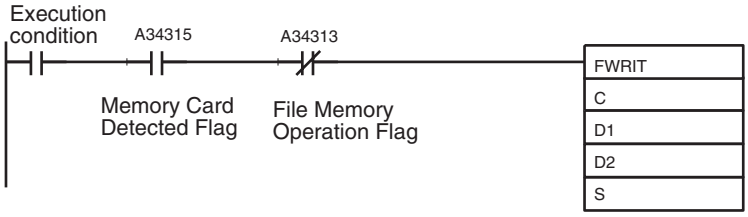
Precautions

The File Memory Operation Flag (A34313) is turned ON when FWRIT(701) is executed. An error will occur and the instruction will not be executed if A34313 is already ON.

The File Write Impossible Flag (A34309) 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 (A34308) 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 (A34315) 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.

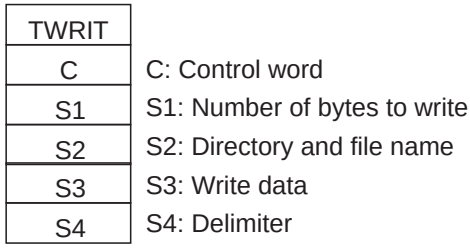
3-26-4 WRITE TEXT FILE: TWRIT(704)

Purpose

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). The data is stored in the TXT format.

This instruction is supported by CS/CJ-series CPU Units with unit version 4.0 or later only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	TWRIT(704)
	Executed Once for Upward Differentiation	@TWRIT(704)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Function block definitions	Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK	OK

Operand

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.

	S1+2	
S2	F1	F2
S2+1	F3	F4
	:	:
S2+38	F73	F74

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

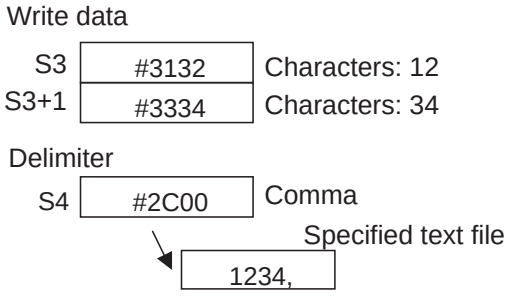
Area	C	S1	S2	S3	S4
CIO Area	CIO 0000 to CIO 6143				
Work Area	W000 to W511				
Holding Bit Area	H000 to H511				
Auxiliary Bit Area	A000 to A959				
Timer Area	T0000 to T4095				
Counter Area	C0000 to C4095				
DM Area	D00000 to D32767				
EM Area without bank	E00000 to E32767				

Area	C	S1	S2	S3	S4
EM Area with bank	En_00000 to En_32767 (n = 0 to C)				
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)				
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)				
Constants	#0000 to #0001	---			#0000 to #FFFF
Data Registers	---				
Index Registers	---				
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to 1–2048 to +2047 ,IR5 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15				

Description

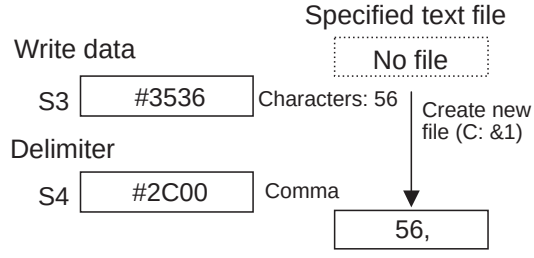
TWRIT(704) writes the number of bytes of data specified in S1, starting from the word specified in S3, to a text file (filename.TXT) in the Memory Card with the filename specified in S2.

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.



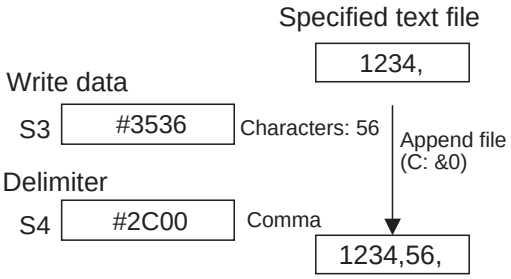
Creating a New File

Set C = 0001 and specify a new filename to create a new file.



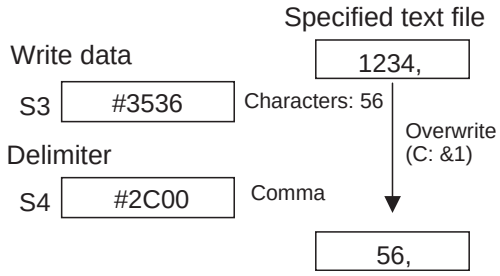
Appending an Existing File

Set C = 0000 to append data to an existing file.



Overwriting an Existing File

Set C = 0001 and specify an existing filename to overwrite an existing file.



Reference

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 (A34313) 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). Refer to *Appendix D* in the *CS/CJ-series Programmable Controllers Operation Manual* (W339) for a memory map showing the location of data areas in RAM.

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.

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 (A34311) will be turned ON and the file will not be overwritten.

Example: Writing to Directory \ABC and Filename XYZ

S2	'\'	'A'	→	S2	5C	41	Saved in ASCII.
S2+1	'B'	'C'		S2+1	42	43	
S2+2	'\'	'Y'		S2+2	5C	5B	
S2+3	'Y'	'Z'		S2+3	59	5A	
S2+4	NUL			S2+4	00		

Flags

Name	Label	Operation
Error Flag	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.

The following table shows relevant flags in the Auxiliary Area.

Name	Label	Operation
Memory Card Format Error Flag	A34307	ON when the Memory Card is not formatted or a formatting error has occurred.
File Write Error Flag	A34308	ON when an error occurred when writing to the file.
File Write Impossible Flag	A34309	ON when the data could not be written because the file was write-protected or there was insufficient free memory.

Name	Label	Operation
No File Flag	A34311	ON when the specified directory does not exist when writing a file.
File Memory Operation Flag	A34313	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	A34314	ON when file data is actually being accessed.
Memory Card Detected Flag	A34315	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.

Note 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. For details, refer to *Section 5 File Memory Functions* in the *SYSMAC CS/CJ Series Programmable Controllers Programming Manual* (W394).

Precautions

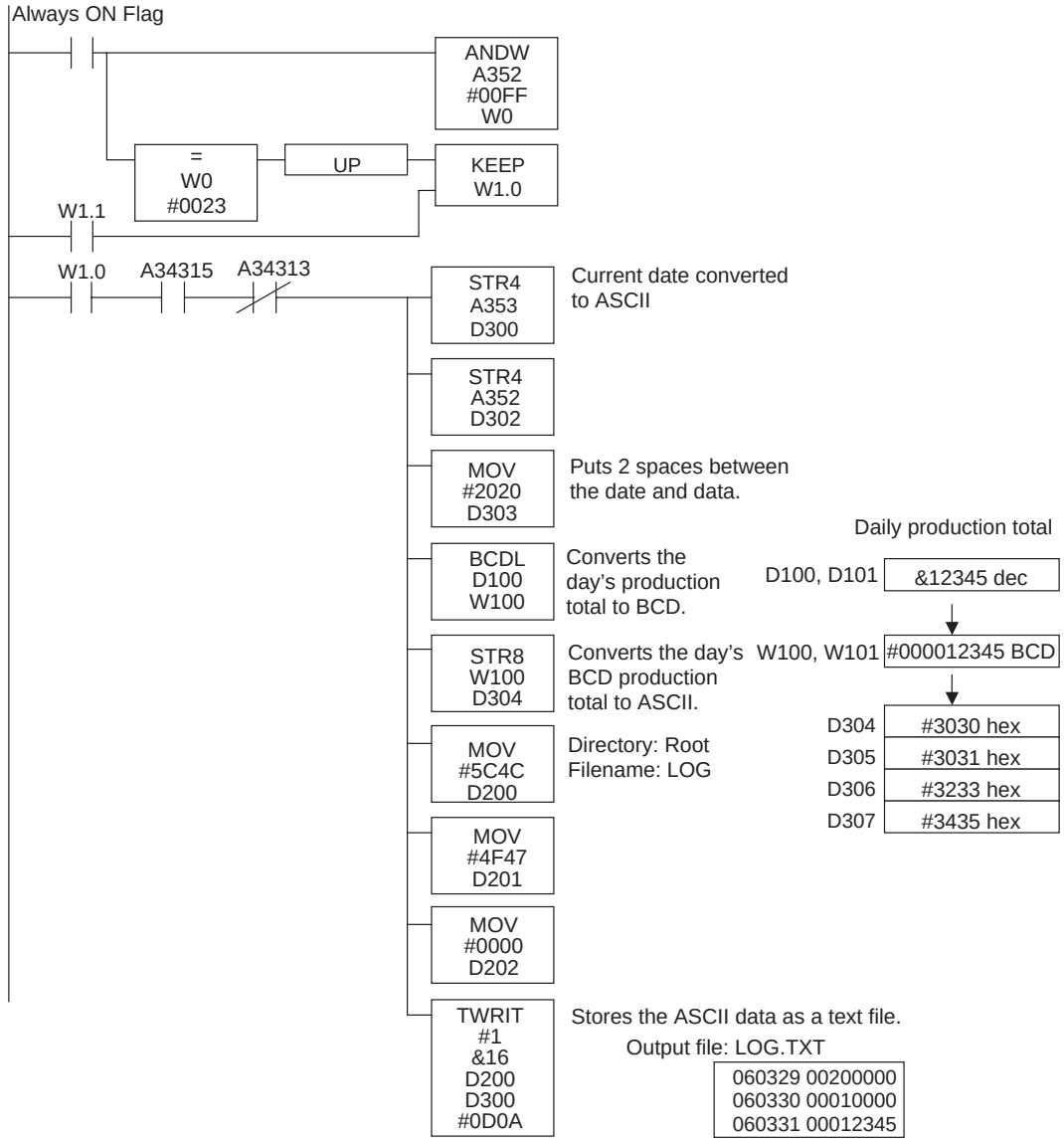
The File Memory Operation Flag (A34313) is turned ON when TWRIT(704) is executed. An error will occur and the instruction will not be executed if A34313 is already ON.

The File Write Impossible Flag (A34309) 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 (A34315) in a NO input condition as shown in the example below to be sure that the Memory Card has been detected.

Example

This example records the daily production total (number of units produced) in D00100 and D00101 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.

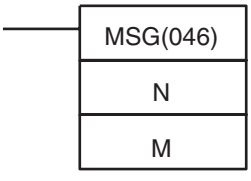


3-27 Display Instructions: DISPLAY MESSAGE: MSG(046)

Purpose

Reads the specified sixteen words of extended ASCII and displays the message on a Peripheral Device such as a Programming Console.

Ladder Symbol



N: Message number

M: First message word

Variations

Variations		
	Executed Each Cycle for ON Condition	MSG(046)
	Executed Once for Upward Differentiation	@MSG(046)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: Message number
The message number must be 0000 to 0007 hexadecimal (or 0 to 7 decimal).

M: First message word
When displaying a message, M specifies the address of the first of the words containing the ASCII message. When clearing a message, M can be any hexadecimal constant (0000 through FFFF).

Operand Specifications

Area	N	M
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	#0000 to #0007 (binary) or &0 to &7	#0000 to #FFFF (binary)
Data Registers	DR0 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

When the execution condition is ON, MSG(046) registers the 16 words of ASCII data (up to 32 characters including the null character) from M to M+15 for the message number specified by N. Once a message has been registered, 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.

Note Refer to *Appendix A* in the *CS/CJ-series Programming Consoles Operation Manual* (W341) for a table showing extended ASCII.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of S is not 0000 to 0007 hexadecimal. OFF in all other cases.

Precautions

Registered messages are updated each time MSG(046) is executed.

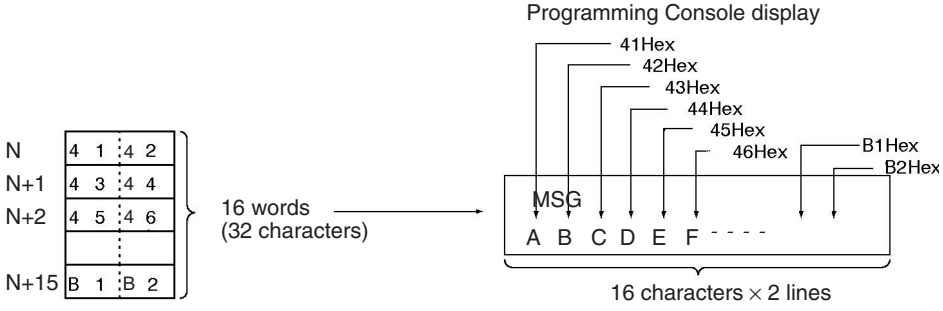
All message characters after the null character (00) are converted to spaces in the Programming Console display.

The character stored in the leftmost byte is displayed before the character in the rightmost byte.

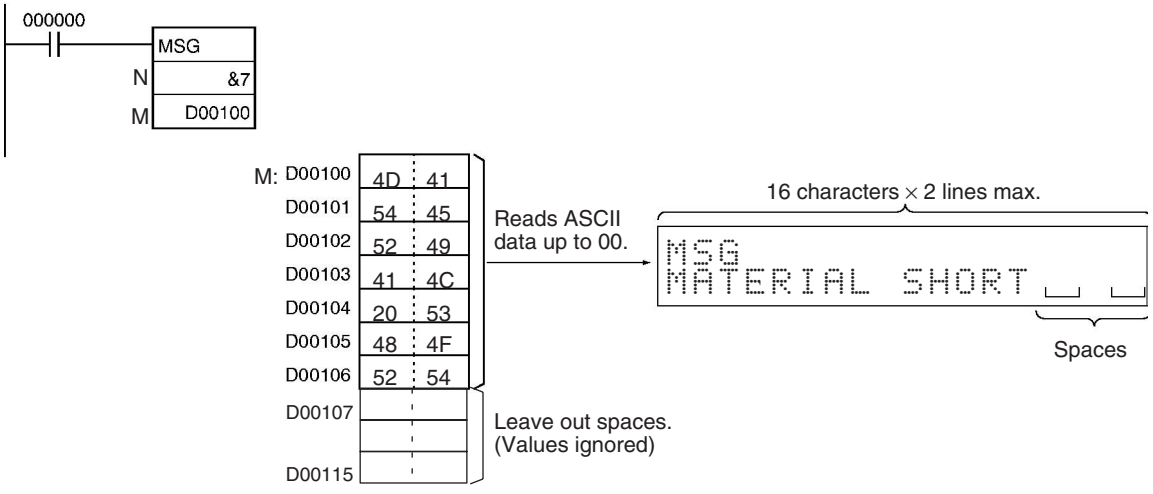
An error will occur and the Error Flag will turn ON if N is not between 0 and 7.

Examples

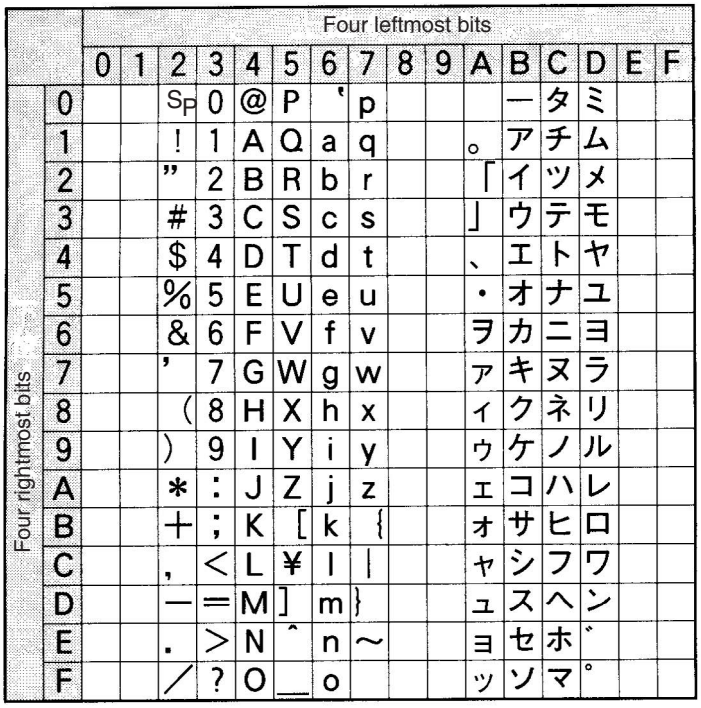
The following diagram shows how 16 words of hexadecimal data are converted to a message displayed on the Programming Console.



When CIO 000000 turns ON in the following example, the 16 words of data in D00100 through D00115 are read as the 32 characters of ASCII data for message number 7 and displayed at the Peripheral device.



ASCII



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

3-28 Clock Instructions

This section describes instructions used with the system clock.

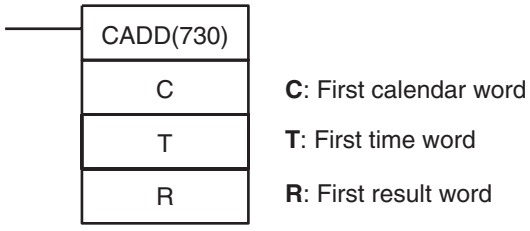
Instruction	Mnemonic	Function code	Page
CALENDAR ADD	CADD	730	1122
CALENDAR SUBTRACT	CSUB	731	1126
HOURS TO SECONDS	SEC	065	1129
SECONDS TO HOURS	HMS	066	1131
CLOCK ADJUSTMENT	DATE	735	1134

3-28-1 CALENDAR ADD: CADD(730)

Purpose

Adds time to the calendar data in the specified words.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	CADD(730)
	Executed Once for Upward Differentiation	@CADD(730)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

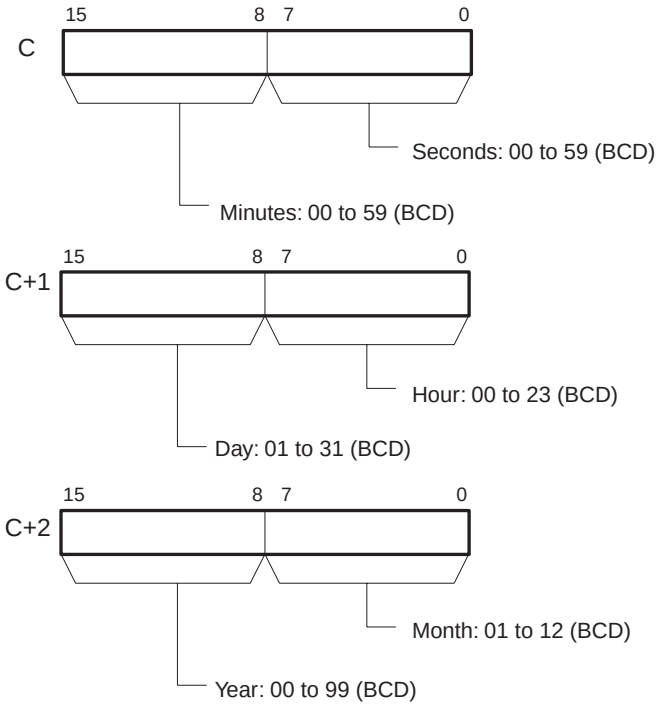
Applicable Program Areas

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

Operands

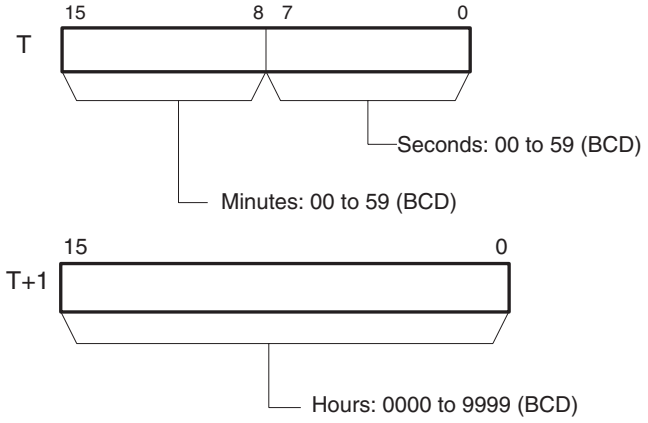
C through C+2: Calendar Data

Set the calendar data in C through C+2 as shown in the following diagram. C through C+2 must be in the same data area.



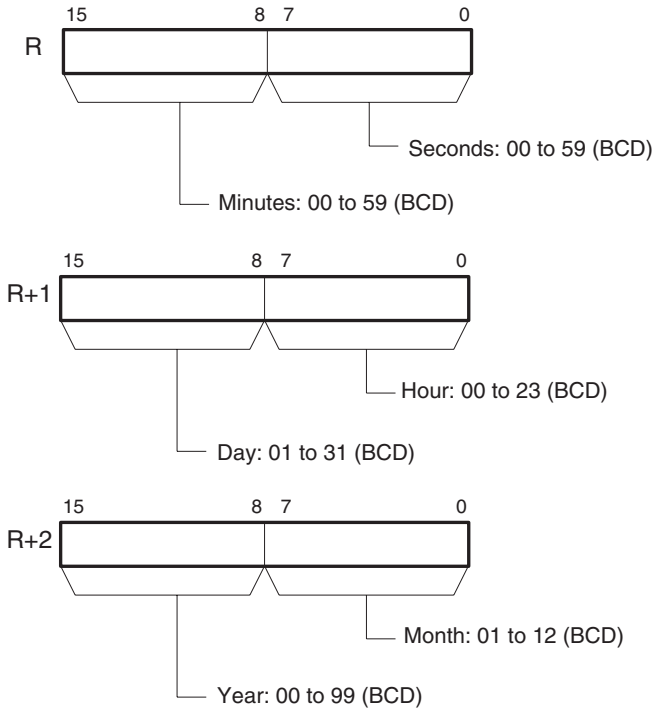
T and T+1: Time Data

Set the time data in T and T+1 as shown in the following diagram. T and T+1 must be in the same data area.



R through R+2: Result Data

R through R+2 contain the result of the addition. R through R+2 must be in the same data area.



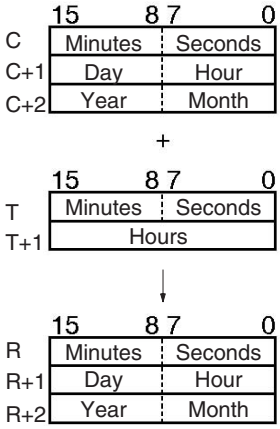
Operand Specifications

Area	C	T	R
CIO Area	CIO 0000 to CIO 6141	CIO 0000 to CIO 6142	CIO 0000 to CIO 6141
Work Area	W000 to W509	W000 to W510	W000 to W509
Holding Bit Area	H000 to H509	H000 to H510	H000 to H509
Auxiliary Bit Area	A000 to A957	A000 to A958	A448 to A957
Timer Area	T0000 to T4093	T0000 to T4094	T0000 to T4093
Counter Area	C0000 to C4093	C0000 to C4094	C0000 to C4093
DM Area	D00000 to D32765	D00000 to D32766	D00000 to D32765
EM Area without bank	E00000 to E32765	E00000 to E32766	E00000 to E32765
EM Area with bank	En_00000 to En_32765 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	En_00000 to 3En_2765 (n = 0 to C)
Indirect DM/EM addresses in binary	@D00000 to @D32767 @E00000 to @E32767 @En_00000 to @En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	---
Data Registers	---		

Area	C	T	R
Index Registers	–		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR005+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

CADD(730) adds the calendar data (words C through C+2) to the time data (words T and T+1) and outputs the resulting calendar data to R through R+2.

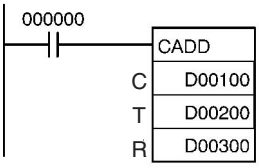


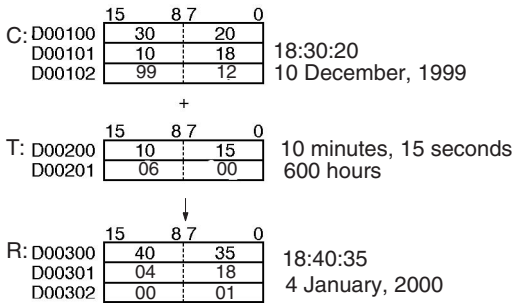
Flags

Name	Label	Operation
Error Flag	ER	ON if the calendar data in C through C+2 is not within the specified ranges. ON if the time data in T and T+1 is not within the specified ranges. OFF in all other cases.

Examples

When CIO 000000 turns ON in the following example, the calendar data in D00100 through D00102 (year, month, day, hour, minutes, seconds) is added to the time data in D00200 and D00201 (hours, minutes, seconds) and the result is output to D00300 through D00302.



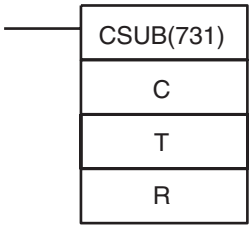


3-28-2 CALENDAR SUBTRACT: CSUB(731)

Purpose

Subtracts time from the calendar data in the specified words.

Ladder Symbol



C: First calendar word

T: First time word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	CSUB(731)
	Executed Once for Upward Differentiation	@CSUB(731)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

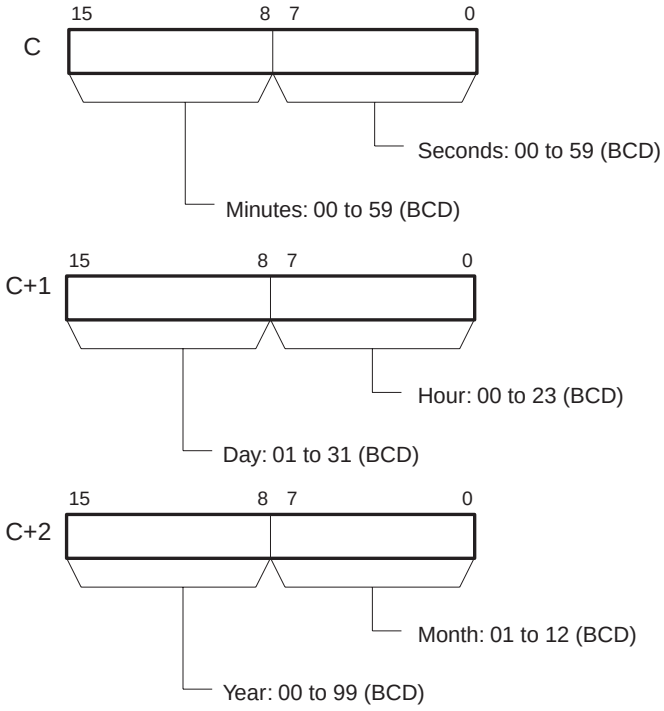
Applicable Program Areas

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

Operands

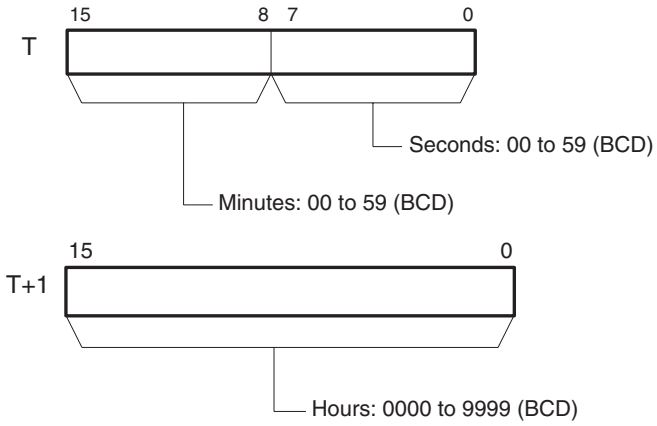
C through C+2: Calendar Data

Set the calendar data in C through C+2 as shown in the following diagram. C through C+2 must be in the same data area.



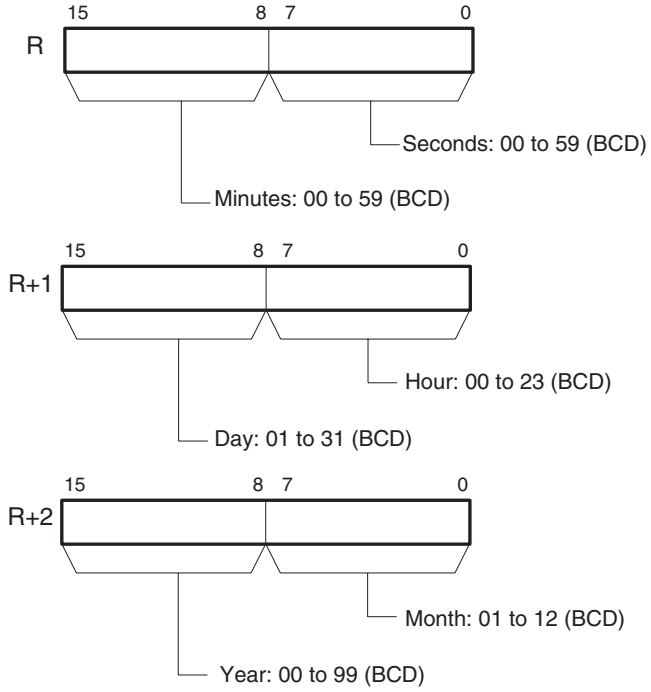
T and T+1: Time Data

Set the time data in T and T+1 as shown in the following diagram. T and T+1 must be in the same data area.



R through R+2: Result Data

R through R+2 contain the result of the addition. R through R+2 must be in the same data area.



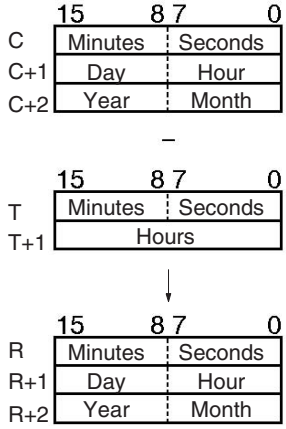
Operand Specifications

Area	C	T	R
CIO Area	CIO 0000 to CIO 6141	CIO 0000 to CIO 6142	CIO 0000 to CIO 6141
Work Area	W000 to W509	W000 to W510	W000 to W509
Holding Bit Area	H000 to H509	H000 to H510	H000 to H509
Auxiliary Bit Area	A000 to A957	A000 to A958	A448 to A957
Timer Area	T0000 to T4093	T0000 to T4094	T0000 to T4093
Counter Area	C0000 to C4093	C0000 to C4094	C0000 to C4093
DM Area	D00000 to D32765	D00000 to D32766	D00000 to D32765

Area	C	T	R
EM Area without bank	E00000 to E32765	E00000 to E32766	E00000 to E32765
EM Area with bank	En_00000 to En_32765 (n = 0 to C)	En_00000 to En_32766 (n = 0 to C)	En_00000 to 3En_2765 (n = 0 to C)
Indirect DM/EM addresses in binary	@D00000 to @D32767 @E00000 to @E32767 @En_00000 to @En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	Specified values only	---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR005+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

CSUB(731) subtracts the time data (words T and T+1) from the calendar data (words C through C+2) to and outputs the resulting calendar data to R through R+2.

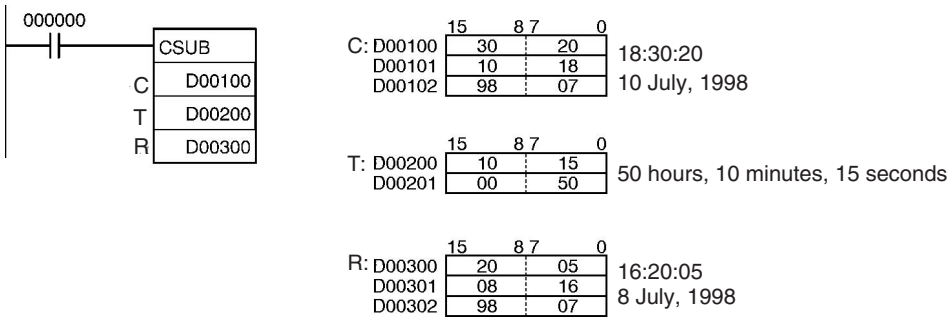


Flags

Name	Label	Operation
Error Flag	ER	ON if the calendar data in C through C+2 is not within the specified ranges. ON if the time data in T and T+1 is not within the specified ranges. OFF in all other cases.

Examples

When CIO 000000 turns ON in the following example, the time data in D00200 and D00201 (hours, minutes, seconds) is subtracted from the calendar data in D00100 through D00102 (year, month, day, hour, minutes, seconds) and the result is output to D00300 through D00302.

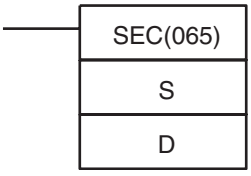


3-28-3 HOURS TO SECONDS: SEC(065)

Purpose

Converts time data in hours/minutes/seconds format to an equivalent time in seconds only.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	SEC(065)
	Executed Once for Upward Differentiation	@SEC(065)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

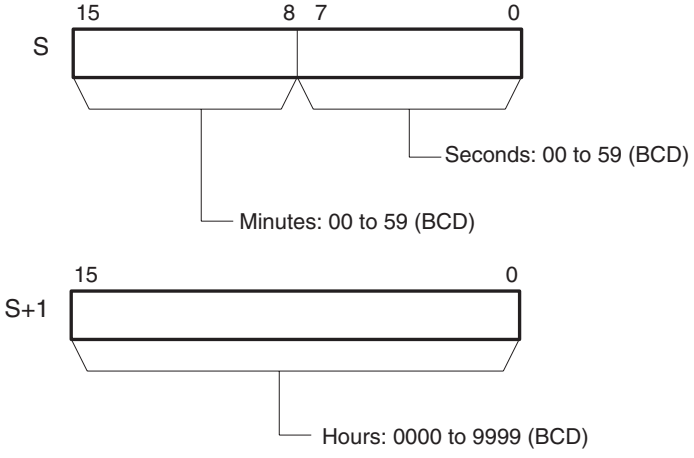
Applicable Program Areas

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

Operands

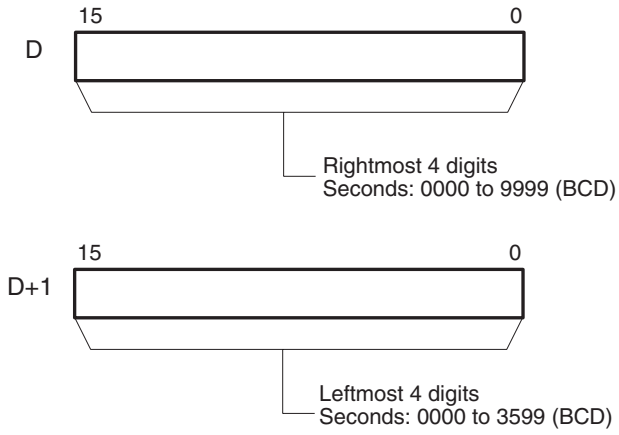
S and S+1: Source Data

Set the hours/minutes/seconds source data in S and S+1, as shown in the following diagram. S and S+1 must be in the same data area.



D and D+1: Result Data

D and D+1 contain the result data in seconds-only format. D and D+1 must be in the same data area.

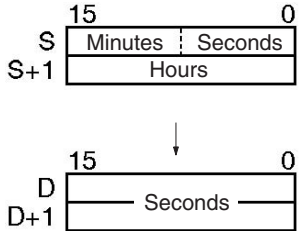


Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	Specified values only	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

SEC(065) converts the 8-digit BCD hours/minutes/seconds data in S and S+1 to 8-digit BCD seconds-only data and outputs the result to D and D+1.



Flags

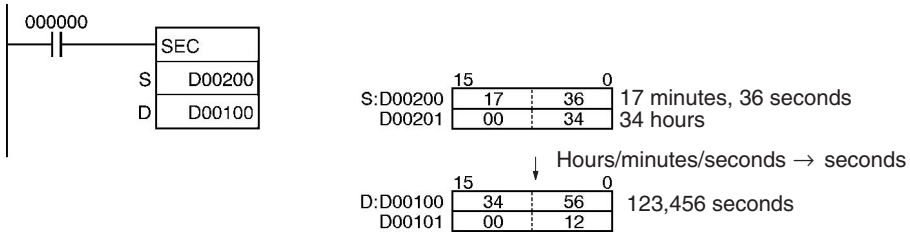
Name	Label	Operation
Error Flag	ER	ON if the minutes data in S (bits 08 to 15) is not BCD and in the range 00 to 59. ON if the seconds data in S (bits 00 to 07) is not BCD and in the range 00 to 59. OFF in all other cases.
Equals Flag	=	ON if the content of D is 0000 after the operation. OFF in all other cases.

Precautions

The maximum value for the source data is 9,999 hours, 59 minutes, and 59 seconds (35,999,999 seconds).

Examples

When CIO 000000 turns ON in the following example, the hours/minutes/seconds data in D00200 and D00201 (34 hours, 17 minutes, and 36 seconds) is converted to seconds-only data and the result is output to D00100 and D00101.

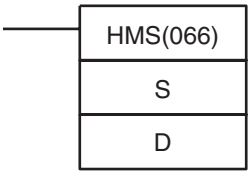


3-28-4 SECONDS TO HOURS: HMS(066)

Purpose

Converts seconds data to an equivalent time in hours/minutes/seconds format.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	HMS(066)
	Executed Once for Upward Differentiation	@HMS(066)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

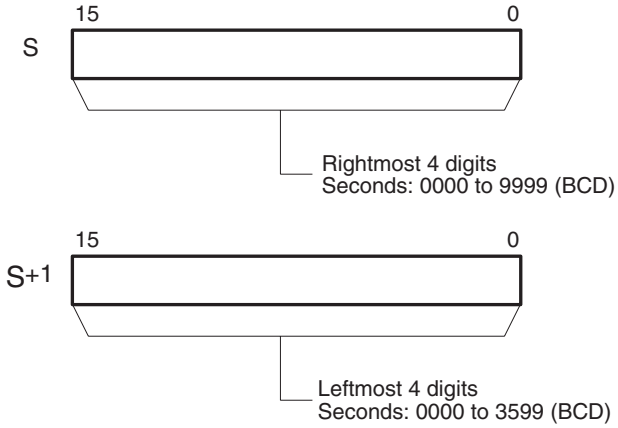
Applicable Program Areas

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

Operands

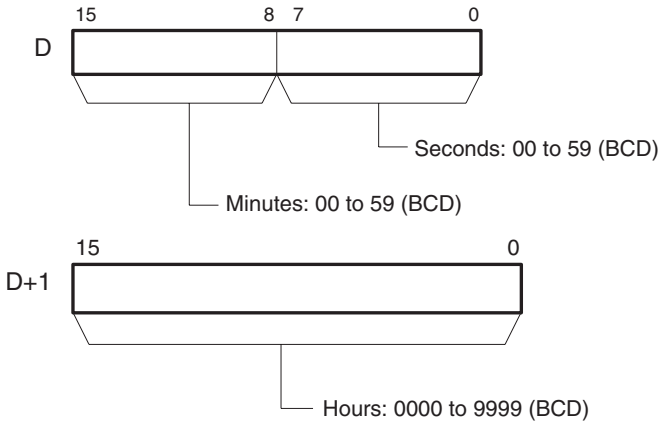
S and S+1: Source Data

Set the seconds source data in S and S+1, as shown in the following diagram. S and S+1 must be in the same data area.



D and D+1: Result Data

D and D+1 contain the result data in hours/minutes/seconds format. D and D+1 must be in the same data area.

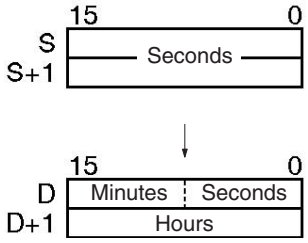


Area	S	D
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W510	
Holding Bit Area	H000 to H510	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T4094	
Counter Area	C0000 to C4094	
DM Area	D00000 to D32766	
EM Area without bank	E00000 to E32766	
EM Area with bank	En_00000 to En_32766 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	

Area	S	D
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	00000000 to 35999999 (BCD)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

HMS(066) converts the 8-digit BCD seconds-only data in S and S+1 to 8-digit BCD hours/minutes/seconds data and outputs the result to D and D+1.



Flags

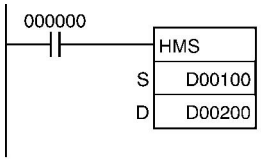
Name	Label	Operation
Error Flag	ER	ON if the seconds data in S and S+1 is not BCD and in the range 0 to 35,999,999. OFF in all other cases.
Equals Flag	=	ON if the content of D is 0000 after the operation. OFF in all other cases.

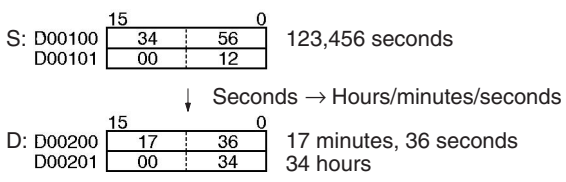
Precautions

The maximum value for the source data is 35,999,999 seconds (9,999 hours, 59 minutes, and 59 seconds).

Examples

When CIO 000000 turns ON in the following example, the seconds data in D00100 and D00101 (123,456 seconds) is converted to hours/minutes/seconds data and the result is output to D00200 and D00201.





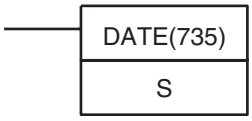
3-28-5 CLOCK ADJUSTMENT: DATE(735)

Purpose

Changes the internal clock setting to the setting in the specified source words.

Note The internal clock setting can also be changed from a Peripheral Device or the CLOCK WRITE FINS command (0702).

Ladder Symbol



S: First source word

Variations

Variations	**Executed Each Cycle for ON Condition**	DATE(735)
	Executed Once for Upward Differentiation	@DATE(735)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

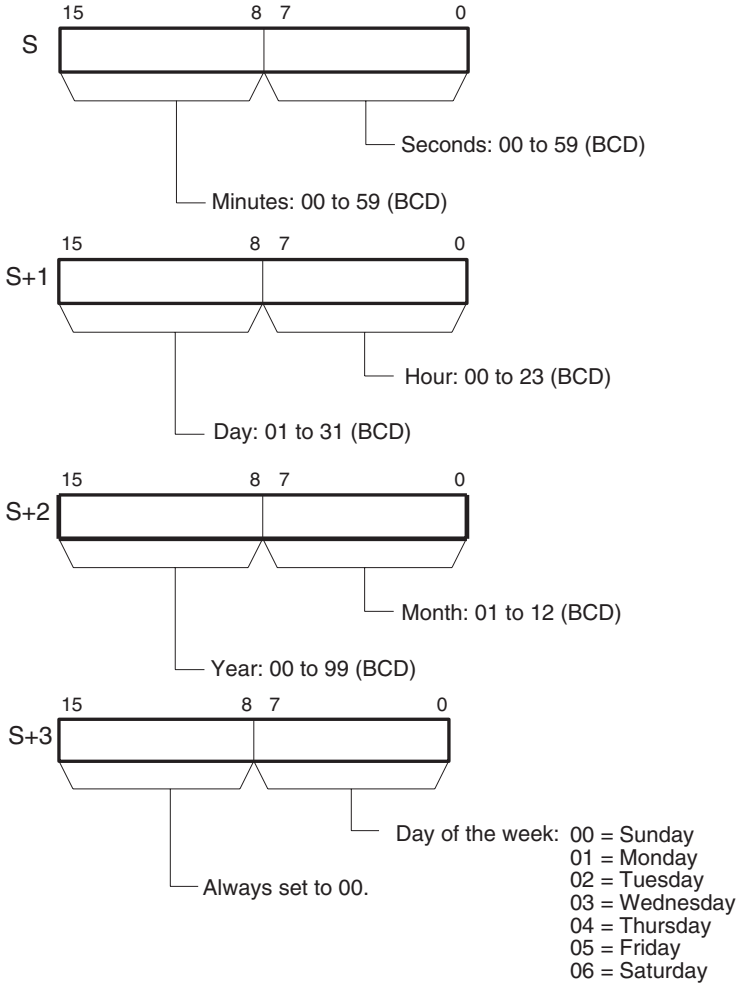
Applicable Program Areas

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

Operands

S through S+3: New Clock Setting

Set the new clock setting in S through S+3 as shown in the following diagram. S through S+3 must be in the same data area.



The following table shows the structure of the Calendar/Clock Area.

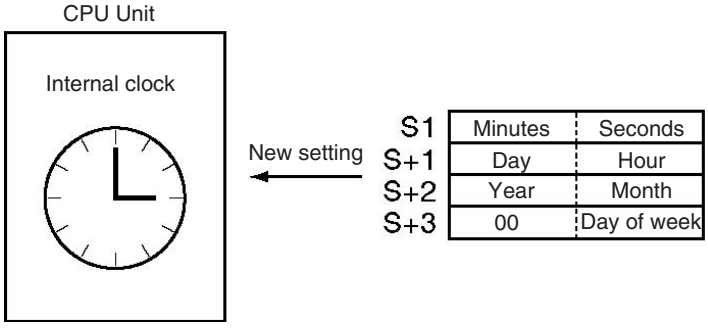
Addresses	Contents
A35100 to A35107	Second (00 to 59, BCD)
A35108 to A35115	Minute (00 to 59, BCD)
A35200 to A35207	Hour (00 to 23, BCD)
A35208 to A35215	Day of month (01 to 31, BCD)
A35300 to A35307	Month (01 to 12, BCD)
A35308 to A35315	Year (00 to 99, BCD)
A35400 to A35407	Day of week (00 to 06 = Sunday to Saturday, hexadecimal)
A35408 to A35415	Always set to 00.

Operand Specifications

Area	S
CIO Area	CIO 0000 to CIO 6140
Work Area	W000 to W508
Holding Bit Area	H000 to H508
Auxiliary Bit Area	A000 to A956
Timer Area	T0000 to T4092
Counter Area	C0000 to C4092
DM Area	D00000 to D32764
EM Area without bank	E00000 to E32764
EM Area with bank	En_00000 to En_32764 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

DATE(735) changes the internal clock setting according to the clock data in the four source words. The new internal clock setting is immediately reflected in the Calendar/Clock Area (A351 to A354).



Flags

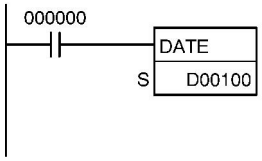
Name	Label	Operation
Error Flag	ER	ON if the new clock setting in S through S+3 is not within the specified range. OFF in all other cases.

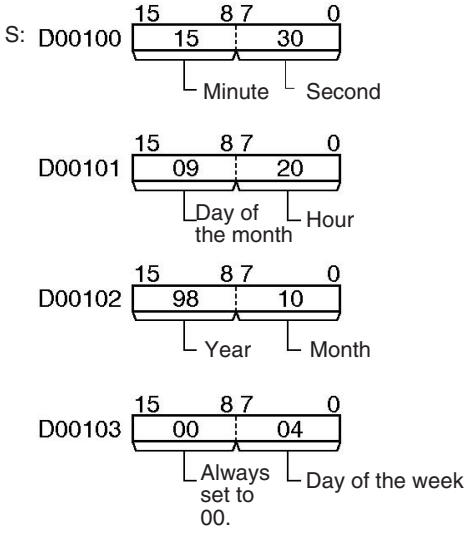
Precautions

An error will not be generated even if the internal clock is set to a non-existent date (such as November 31).

Examples

When CIO 000000 turns ON in the following example, the internal clock is set to 20:15:30 on Thursday, October 9, 1998.





3-29 Debugging Instructions

3-29-1 Trace Memory Sampling: TRSM(045)

Purpose

When TRSM(045) is executed, the status of a preselected bit or word is sampled and stored in Trace Memory. TRSM(045) can be used anywhere in the program, any number of times.

Ladder Symbol

Variations

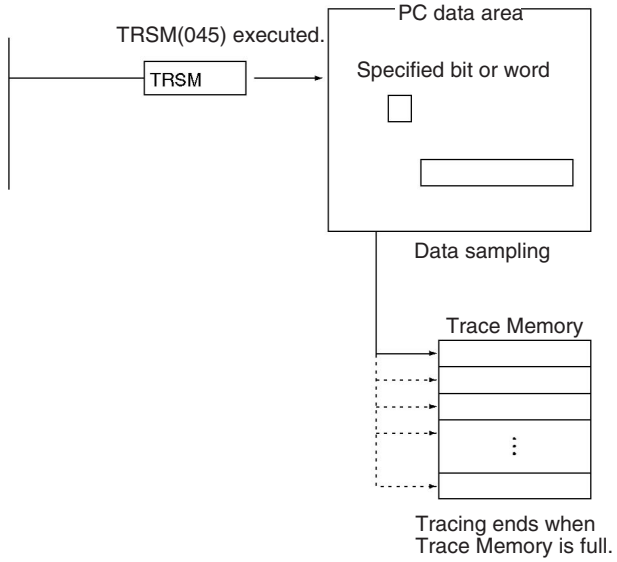
Variations	**Executed Each Cycle**	TRSM(045)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

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

Description

Before TRSM(045) is executed, the bit or word to be traced must be specified with 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.

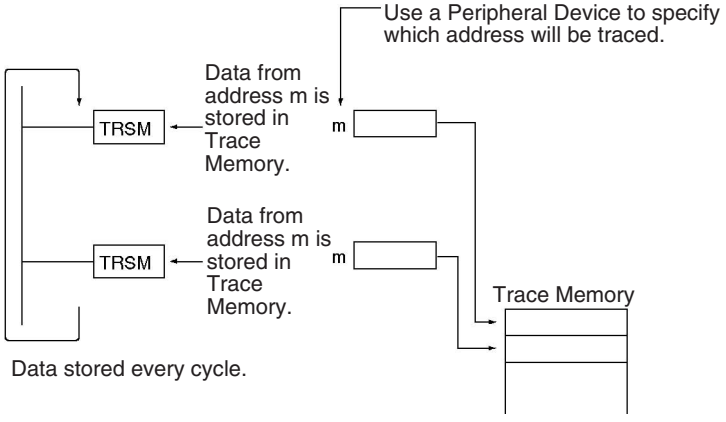


This instruction only indicates when the specified data will be sampled. All other settings and data trace operations are set with a Peripheral Device. The other two ways to control data sampling are sampling at the end of each cycle and sampling at a specified interval (independent of the cycle time).

TRSM(045) does not require an execution condition and is always executed as if it had an ON execution condition. Connect TRSM(045) directly to the left bus bar.

Use TRSM(045) to sample the value of the specified bit or word at the point in the program when the instruction's execution condition is ON. If the instruction's execution condition is ON every cycle, the specified bit or word's value will be stored in Trace Memory every cycle.

It is possible to incorporate two or more TRSM(045) instructions in a program. In this case, the value of the same specified bit or word will be stored in Trace Memory each time that one of the TRSM(045) instructions is executed.



Note Refer to the Peripheral Device's Operation Manual for details on data tracing.

The data-tracing operations performed with the Peripheral Device are summarized in the following list.

1,2,3... 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 (A50813) 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 (A50811) 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 (A50812) will be turned ON and the Trace Busy Flag (A50813) will be turned OFF.
5. Read the contents of Trace Memory with the Peripheral Device.

The following table shows relevant bits and flags in the Auxiliary Area. Only A50814 and A50815 are meant to be controlled by the user, and A00815 must not be turned ON from the program, i.e., it must be turned ON only from a Peripheral Device.

Name	Address	Operation
Trace Trigger Monitor Flag	A50811	This flag is turned ON when the trigger condition has been established with the Trace Start Bit. It is turned OFF when sampling is started for the next trace (by the Sampling Start Bit).
Trace Completed Flag	A50812	This flag is turned ON when trace samples have filled the Trace Memory. It is turned OFF the next time that the Sampling Start Bit goes from OFF to ON.
Trace Busy Flag	A50813	This flag is turned ON when the Sampling Start Bit goes from OFF to ON. It is turned OFF when the trace is completed.

Name	Address	Operation
Trace Start Bit	A50814	The trace trigger conditions are established when this bit is turned from OFF to ON. Samples will be recorded after the specified delay (positive delay) or the specified number of existing samples will be valid (negative delay).
Sampling Start Bit	A50815	When this bit is turned from OFF to ON from a Peripheral Device, data samples will start being stored in Trace Memory with one of the following three methods used to determine sampling: 1) Periodic sampling (10 to 2,550 ms intervals) 2) Sampling at TRSM(045) execution 3) Sampling at the end of each cycle This bit must be turned ON and OFF from a Peripheral Device.

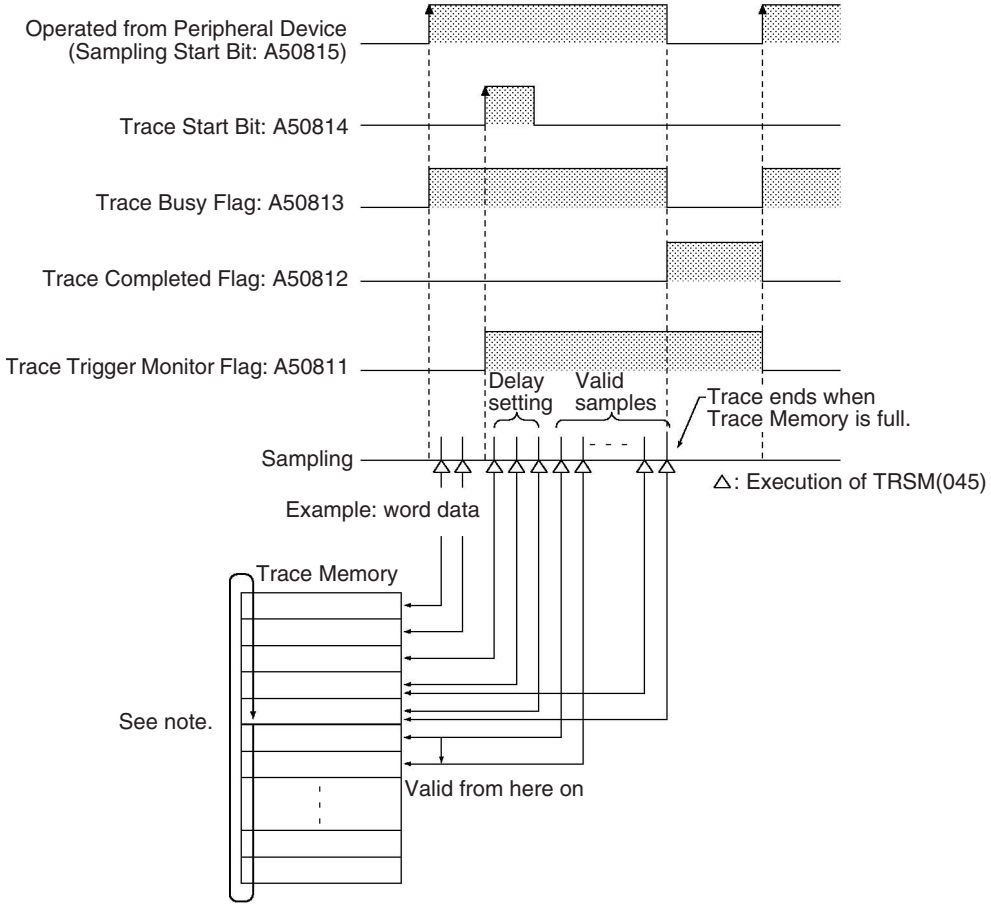
Precautions

TRSM(045) is processed as NOP(000) when data tracing is not being performed or when the sampling interval set in the parameters with a Peripheral Device is not set to sample on TRSM(045) instruction execution.

Do not turn the Sampling Start Bit (A50815) ON or OFF from the program. This bit must be turned ON and OFF from a Peripheral Device.

Example

The following example shows the overall data trace operation.



Note Trace Memory has a ring structure. Data is stored to the end of the Trace Memory area and then wraps to the beginning of the area, ending just before the first valid data sample.

3-30 Failure Diagnosis Instructions

This section describes instructions used to define and handle errors.

Instruction	Mnemonic	Function code	Page
FAILURE ALARM	FAL	006	1140
SEVERE FAILURE ALARM	FALS	007	1148
FAILURE POINT DETECTION	FPD	269	1156

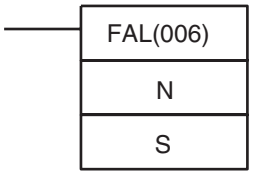
3-30-1 FAILURE ALARM: FAL(006)

Purpose

Generates or clears user-defined non-fatal errors. Non-fatal errors do not stop PLC operation.
With CS1-H, CJ1-H, and CJ1M CPU Units, FAL(006) can also be used to generate non-fatal system errors.

Ladder Symbol

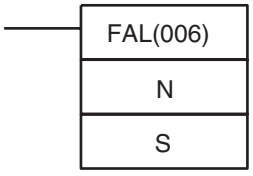
- Generating or Clearing User-defined Non-fatal Errors



N: FAL number

S: First message word or constant (0000 to FFFF)

- Generating Non-fatal System Errors (CS1-H, CJ1-H, CJ1M, or CS1D Only)



N: FAL number (value in A529)

S: First word containing the error code and error details

Variations

Variations	**Executed Each Cycle for ON Condition**	FAL(006)
	Executed Once for Upward Differentiation	@FAL(006)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

The function of the operands when FAL(006) is used to generate/clear user defined errors is slightly different from the function when FAL(006) is used to generate system errors (CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only).

Generating or Clearing User-defined Non-fatal Errors

The following table shows the function of the operands.

Note The value of operand N must be **different from** the content of A529 (the system-generated FAL/FALS number).

N	S	Function
0	#0001 to #01FF	Clears the non-fatal error with the corresponding FAL number.
	#FFFF	Clears all non-fatal errors.
	Other*	Clears the most serious non-fatal error.
1 to 511 (These FAL numbers are shared with FALS numbers.)	#0000 to #FFFF	Generates a non-fatal error with the corresponding FAL number (no message).
	Word address	Generates a non-fatal error with the corresponding FAL number. The 16-character ASCII message contained in S through S+7 will be displayed on the Programming Device.

Note *Other settings would be constants #0200 through #FFFE or a word address.

Generating Non-fatal System Errors (CS1-H, CJ1-H, CJ1M, or CS1D Only)

The following table shows the function of the operands.

Note The value of operand N must be **the same as** the content of A529 (the system-generated FAL/FALS number).

Operand	Function
N	1 to 511 (These FAL numbers are shared with FALS numbers.)
S	Error code that will be generated. (See *Description* below.)
S+1	Error details code that will be generated. (See *Description* below.)

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	0 to 511	#0000 to #FFFF (binary)
Data Registers	---	

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

Description

The operation of FAL(006) depends on the value of N. Set N to 0000 to clear an error and set N to 0001 to 01FF to generate an error. A system error will be generated if the value of N equals the content of A529 (CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only).

Generating Non-fatal User-defined Errors

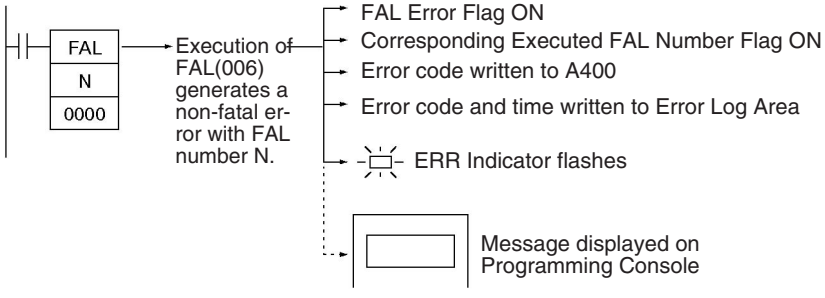
When FAL(006) is executed with N set to an FAL number (&1 to &511) that is not equal to the content of A529 (the system-generated FAL/FALS number), a non-fatal error will be generated with that FAL number and the following processing will be performed:

1,2,3...

1. The FAL Error Flag (A40215) will be turned ON. (PLC operation will continue.)
2. The Executed FAL Number Flag will be turned ON for the corresponding FAL number. Flags A36001 to A39115 correspond to FAL numbers 0001 to 01FF (1 to 511).
3. The error code will be written to A400. Error codes 4101 to 42FF correspond to FAL numbers 0001 to 01FF (1 to 511).

   Note If a fatal error or a more serious non-fatal error occurs at the same time as the FAL(006) instruction, the more serious error's error code will be written to A400.
4. The error code and the time that the error occurred will be written to the Error Log Area (A100 through A199).

   Note The error record will not be written to the Error Log Area if the *Don't register FAL to error log* Option in the PLC Setup is selected. (This option is supported only by the CS1-H, CCJ1-H, CJ1M, and CS1D CPU Units.)
5. The ERR Indicator on the CPU Unit will flash.
6. If a word address has been specified in S, the message beginning at S will be registered (displayed on the Programming Device).



The following table shows the error codes and FAL Error Flags for FAL(006).

FAL number	FAL error codes	Executed FAL Number Flags
1 to 511 decimal	4101 to 42FF	A36001 to A39115

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.

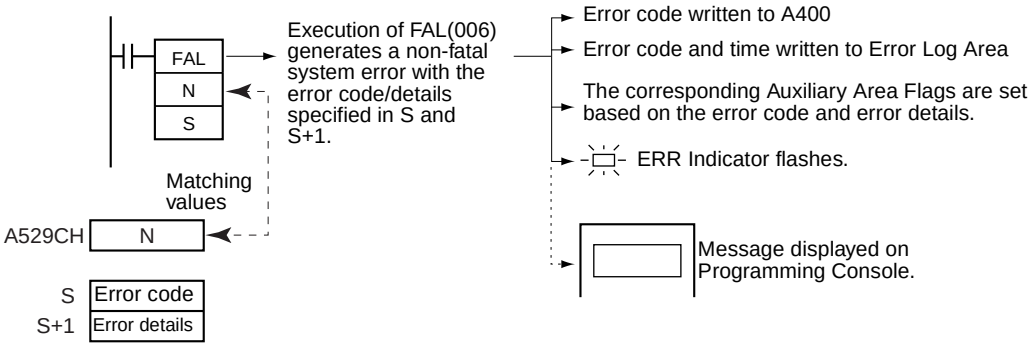
An ASCII message up to 16 characters long can be stored in S through S+7. The leftmost (most significant) byte in each word is displayed first.

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.

Generating Non-fatal System Errors (CS1-H, CJ1-H, CJ1M, or CS1D Only)

When FAL(006) is executed with N set to an FAL number (&1 to &511) that is equal to the content of A529 (the system-generated FAL/FALS number), a non-fatal error will be generated with the error code and error details code specified in S and S+1. The following processing will be performed at the same time:



1,2,3...

1. The specified error code will be written to A400.
2. The error code and the time that the error occurred will be written to the Error Log Area (A100 through A199).
3. The appropriate Auxiliary Area Flags are set based on the error code and error details.
4. The ERR Indicator on the CPU Unit will flash and PLC operation will continue.
5. The non-fatal error message for the specified system error will be displayed on the 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 (A36001 to A39115) or the error code.
   When it is necessary to generate two or more system errors (fatal and/or non-fatal errors), different errors can be generated by executing the FAL/

FALS/FPD instructions more than once with the same values in A529 and N, but different values in S and S+1.

3. If a more serious error (including a system-generated fatal error or FALS(007) error) occurs at the same time as the FAL(006) instruction, the more serious error's error code will be written to A400.
4. To clear a system error generated by FAL(006), turn the PLC OFF and then ON again. The PLC can be kept ON, but the same processing will be required to clear the error as if the specified error had actually occurred.

The following table shows how to specify error codes and error details in S and S+1.

Error name	S	S+1
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
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
CS1 CPU Bus Unit Error	0200 hex	CS1 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)
CS1 CPU Bus Unit Setup Error	0400 hex	CS1 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

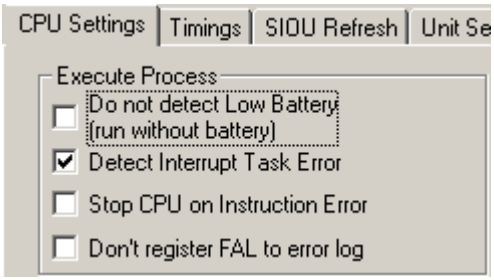
Disabling Error Log Entries of User-defined Errors (CS1-H, CJ1-H, CJ1M, or CS1D Only)

Normally when FAL(006) generates a user-defined error, the error code and the time that the error occurred are written to the Error Log Area (A100 through A199). It is possible to set the PLC Setup so that user-defined errors generated by FAL(006) are not recorded in the Error Log.

Even though the error will not be recorded in the Error Log, the FAL Error Flag (40215) will be turned ON, the corresponding flag in the Executed FAL Number Flags (A36001 to A39115) 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.



The following table shows the PLC Setup setting from the Programming Console.

Item	Setting	
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)	

Even if PLC Setup word 129 bit 15 is set to 1 (Do not record FAL Errors in Error Log.), the following errors will be recorded:

- Fatal errors generated by FALS(007)
- Non-fatal errors from the system
- Fatal errors from the system
- Non-fatal errors from the system generated intentionally with FAL(006) or FPD(269)
- Fatal errors from the system generated intentionally with FALS(007)

Clearing Non-fatal Errors without a Programming Device

1. Clearing User-defined Non-fatal Errors
   When FAL(006) is executed with N set to 0, non-fatal errors can be cleared. The value of S will determine the processing, as shown in the following table.

S	Process
&1 to &511 (0001 to 01FF hex)	The FAL error of the specified number will be cleared.
FFFF hex	All non-fatal errors (including system errors) will be cleared.
0200 to FFFE hex or word specification	The most serious non-fatal error (even if it is a non-fatal system error) that has occurred. When more than one FAL error has occurred, the FAL error with the smallest FAL number will be cleared.

2. Clearing Non-fatal System Errors (CS1-H, CJ1-H, CJ1M, and CS1D CPU Units Only)
   There are two ways to clear non-fatal system errors generated with FAL(006).
   - Turn the PLC OFF and then ON again.

- When keeping the PLC ON, the system error must be cleared as if the specified error had actually occurred.

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0 to 511 decimal. ON if a non-fatal system error is being generated (CS1-H/CJ1-H/CJ1M/CS1D Only), but the specified error code or error details code is incorrect. OFF in all other cases.

The following tables show relevant words and flags in the Auxiliary Area.

- Auxiliary Area Words/Flags for User-defined Errors Only

Name	Address	Operation
FAL Error Flag	A40215	ON when an error is generated with FAL(006).
Executed FAL Number Flags	A36001 to A39115	When an error is generated with FAL(006), the corresponding flag will be turned ON. Flags A36001 to A39115 correspond to FAL numbers 0001 to 01FF.

- Auxiliary Area Words/Flags for System Errors Only (CS1-H, CJ1-H, CJ1M, and CS1D CPU Units Only)

Name	Address	Operation
System-generated FAL/FALS number	A529	A dummy FAL/FALS number is used when a system error is generated with FAL(006). Set the same dummy FAL/FALS number in this word (0001 to 01FF hex, 1 to 511 decimal).

- Auxiliary Area Words/Flags for both User-defined and System Errors

Name	Address	Operation
Error Log Area	A100 to A199	The Error Log Area contains the error codes and time/date of occurrence for the most recent 20 errors, including errors generated by FAL(006).
Error code	A400	When an error occurs its error code is stored in A400. The error codes for FAL numbers 0001 to 01FF are 4101 to 42FF, respectively. If two or more errors occur simultaneously, the error code of the most serious error will be stored in A400.

Precautions

N must between 0000 and 01FF. An error will occur and the Error Flag will be turned ON if N is outside of the specified range.

Examples

Generating a Non-fatal Error

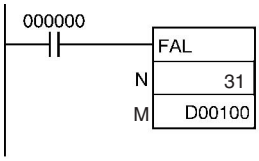
When CIO 000000 is ON in the following example, FAL(006) will generate a non-fatal error with FAL number 31 and execute the following processes.

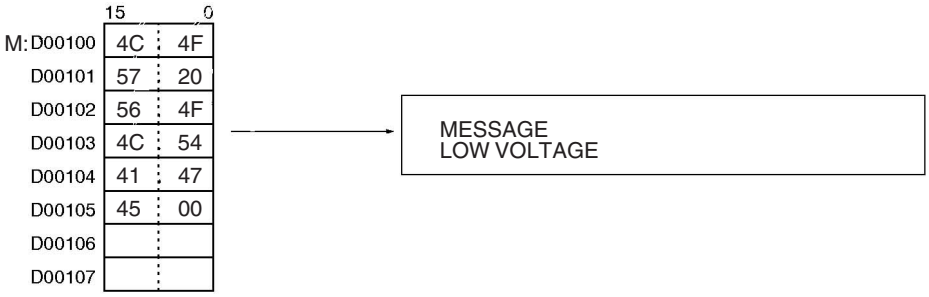
1,2,3...

1. The FAL Error Flag (A40215) will be turned ON.
2. The corresponding Executed FAL Number Flag (A36114) will be turned ON.
3. The corresponding error code (411F) will be written to A400.

Note If two or more errors occur at the same time, the error code of the most serious error (with the highest error code) will be stored in A400.

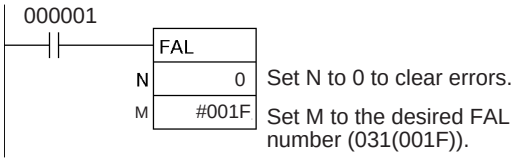
4. The error code and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
5. The ERR Indicator on the CPU Unit will flash.
6. The ASCII message in D00100 to D00107 will be displayed at the Peripheral Device. (If a message is not required, specify a constant for S.)





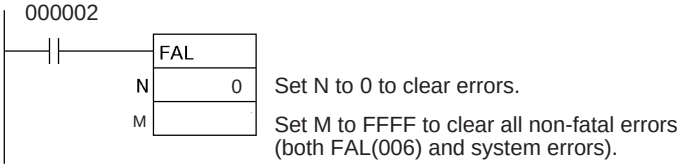
Clearing a Particular Non-fatal Error

When CIO 000001 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 (A36114), and turn OFF the FAL Error Flag (A40215).



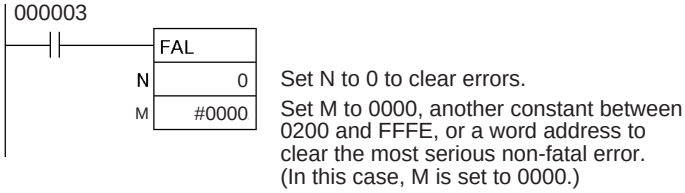
Clearing All Non-fatal Errors

When CIO 000002 is ON in the following example, FAL(006) will clear all of the non-fatal errors, turn OFF the Executed FAL Number Flags (A36001 to A39115), and turn OFF the FAL Error Flag (A40215).



Clearing the Most Serious Non-fatal Error

When CIO 000003 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 (A40215) will be turned OFF.

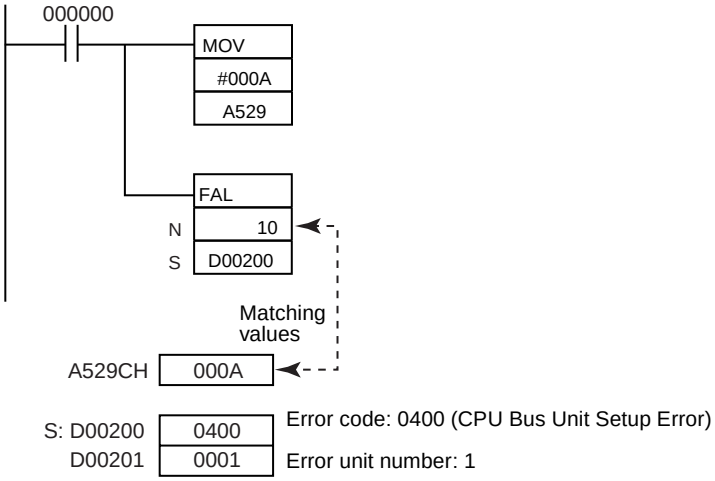


Generating a Non-fatal System Error (CS1-H, CJ1-H, CJ1M, or CS1D Only)

When CIO 000000 is ON in the following example, FAL(006) will generate a CPU Bus Unit Setup Error for unit number 1. In this case, dummy FAL number 10 is used and the corresponding value (000A hex) is stored in A529.

1,2,3...

1. The specified error code (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 (A40203) and CPU Bus Unit Setup Error Flag for unit number 1 (A42701) 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.



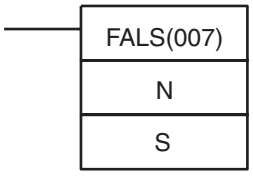
3-30-2 SEVERE FAILURE ALARM: FALS(007)

Purpose

Generates user-defined fatal errors. Fatal errors stop PLC operation.
With CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, FALS(007) can also be used to generate fatal system errors.

Ladder Symbol

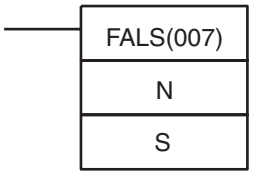
- Generating User-defined Fatal Errors



N: FALS number

S: First message word or constant (0000 to FFFF)

- Generating Fatal System Errors (CS1-H, CJ1-H, CJ1M, or CS1D Only)



N: FALS number (value in A529)

S: First word containing the error code and error details

Variations

Variations	**Executed Each Cycle for ON Condition**	FALS(007)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

Generating User-defined Fatal Errors

The following table shows the function of the operands.

Note The value of operand N must be **different from** the content of A529 (the system-generated FAL/FALS number).

Operand	Function
N	1 to 511 (These FALS numbers are shared with FAL numbers.)
S	Specifies the first of eight words containing an ASCII message to be displayed on the Programming Device. Specify a constant (0000 to FFFF) if a message is not required.

Generating Fatal Errors from the System (CS1-H, CJ1-H, CJ1M, or CS1D Only)

The following table shows the function of the operands.

Note The value of operand N must be **the same as** the content of A529 (the system-generated FAL/FALS number).

Operand	Function
N	1 to 511 (These FALS numbers are shared with FAL numbers.)
S	Error code that will be generated. (See *Description* below.)
S+1	Error details code that will be generated. (See *Description* below.)

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	Specified values only	#0000 to #FFFF (binary)
Data Registers	---	

Area	N	S
Index Registers	---	
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR+(++)0 to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

FALS(007) generates a fatal error. In CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, FALS(007) can also be used to generate fatal system errors as well as fatal user-defined errors. (A system error will be generated if the value of N equals the content of A529.)

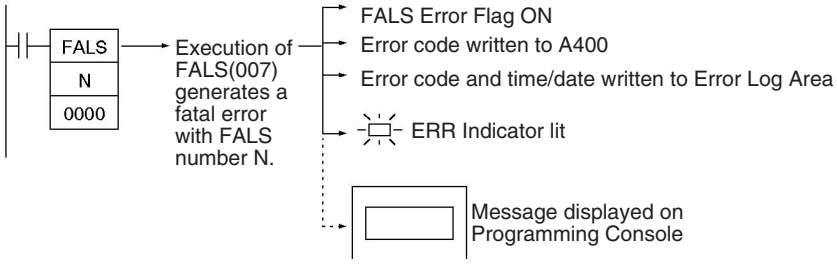
Generating Fatal User-defined Errors

When FALS(007) is executed with N set to an FALS number (1 to 511) that is not equal to the content of A529 (the system-generated FAL/FALS number), a fatal error will be generated with that FALS number and the following processing will be performed:

1,2,3...

1. The FALS Error Flag (A40106) 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).



The following table shows the error codes for FALS(007).

FALS number	FALS error codes
1 to 511	C101 TO C2FF

Note The input method for the FALS number, N, is different for the CX-Programmer and a Programming Console. Input #1 to #511 on the CX-Programmer and input 001 to 511 on a Programming Console.

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.)

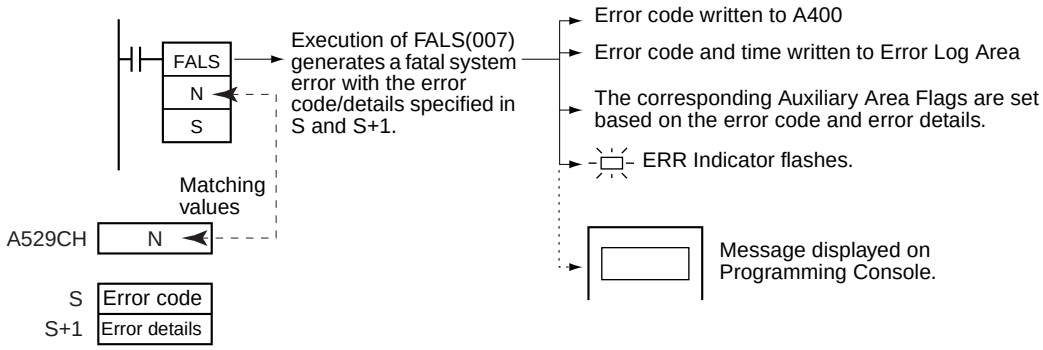
The message beginning at S will be registered when FALS(007) is executed. Once the message is registered, it will be displayed when a Programming Console is connected.

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.

Generating Non-fatal System Errors (CS1-H, CJ1-H, CJ1M, or CS1D Only)



When FALS(007) is executed with N set to an FAL number (1 to 511) that is equal to the content of A529 (the system-generated FAL/FALS number), a fatal error will be generated with the error code and error details code specified in S and S+1. The following processing will be performed at the same time:

1,2,3...

1. The specified error code will be written to A400.
2. The error code and the time that the error occurred will be written to the Error Log Area (A100 through A199).
3. The appropriate Auxiliary Area Flags are set based on the error code and error details.
4. The ERR Indicator on the CPU Unit will light and PLC operation will be stopped.
5. The fatal error message for the specified system error will be displayed on the Programming 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* for details.

4. The following table shows how the IOM Hold Bit affects the status of I/O memory and the status of outputs on Output Units after a fatal system error has been generated with FALS(007).

IOM Hold Bit (A50012)	Status of I/O memory	Status of outputs on Output Units
ON	Retained	OFF
OFF	Cleared	OFF

Note Unlike user-defined fatal errors, system errors generated by FALS(007) will clear I/O memory if the IOM Hold Bit is OFF. The following areas will be cleared: CIO Area, Work Area, Timer Flags and PVs, Index Registers, and Data registers.

The following table shows how to specify error codes and error details in S and S+1.

Error name	S	S+1
	Error code	Error details
Memory Error	80F1 hex	• Bits 00 to 09: Memory Error Location Bit 00: User program Bit 04: PLC Setup Bit 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

Error name	S	S+1
	Error code	Error details
Too Many I/O Points Error	80E1 hex	Bits 13 to 15: Error Cause Bits 00 to 12: Details • 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) • 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	--- (Not fixed.)
Program Error	80F0 hex	• Bits 08 to 15: Error Cause Bit 15: UM overflow error Bit 14: Illegal instruction error Bit 13: Differentiation overflow error Bit 12: Task error Bit 11: No END error Bit 10: Illegal access error Bit 09: Indirect DM/EM BCD error Bit 08: Instruction error • Bits 00 to 07: Invalid
Cycle Time Overrun Error	809F hex	--- (Not fixed.)

Clearing FALS(007) Fatal System Errors (CS1-H, CJ1-H, CJ1M, and CS1D CPU Units Only)

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.

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 (CS1-H/CJ1-H/CJ1M/CS1D Only), but the specified error code or error details code is incorrect. OFF in all other cases.

The following tables show relevant words and flags in the Auxiliary Area.

- Auxiliary Area Words/Flags for User-defined Errors Only

Name	Address	Operation
FALS Error Flag	A40106	ON when an error is generated with FALS(007).

- Auxiliary Area Words/Flags for System Errors Only (CS1-H, CJ1-H, CJ1M, and CS1D CPU Units Only)

Name	Address	Operation
System-generated FAL/FALS number	A529	A dummy FAL/FALS number is used when a system error is generated with FALS(007). Set the same dummy FAL/FALS number in this word (0001 to 01FF hex, 1 to 511 decimal).

- Auxiliary Area Words/Flags for both User-defined and System Errors

Name	Address	Operation
Error Log Area	A100 to A199	The Error Log Area contains the error codes and time/date of occurrence for the most recent 20 errors, including errors generated by FALS(007).
Error code	A400	When an error occurs its error code is stored in A400. The error codes for FALS numbers 0001 to 01FF (1 to 511 decimal) are C101 to C2FF, respectively. If two or more errors occur simultaneously, the error code of the most serious error will be stored in A400.

Precautions

The end code for the message is the null character (00 hexadecimal). All 16 characters in words S to S+7 will be displayed if the null character is omitted.

N must between 0001 and 01FF. An error will occur and the Error Flag will be turned ON if N is outside of the specified range.

Examples

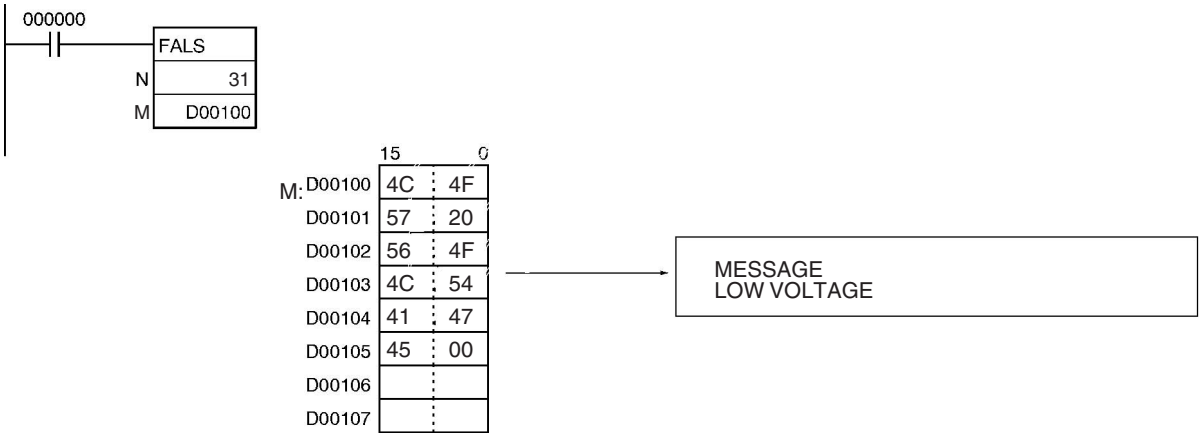
Generating a User-defined Error

When CIO 000000 is ON in the following example, FALS(007) will generate a fatal error with FAL number 31 and execute the following processes.

1,2,3...

1. The FALS Error Flag (A40106) will be turned ON.
2. The corresponding error code (C11F) will be written to A400.

   Note A400 will contain the error code of the most serious of all of the errors that have occurred, including non-fatal and fatal system errors, as well as errors generated by FAL(006) and FAL(007).
3. The error code and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
4. The ERR Indicator on the CPU Unit will be lit.
5. The ASCII message in D00100 to D00107 will be displayed at the Peripheral Device. (If a message is not required, specify a constant for S.)

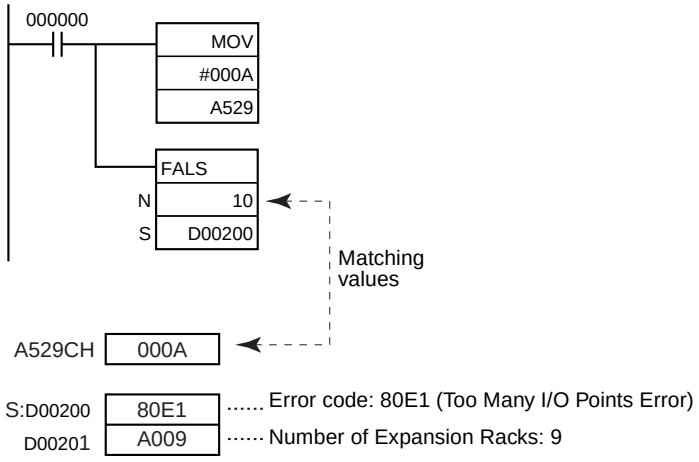


Generating a Non-fatal System Error (CS1-H, CJ1-H, CJ1M, and CS1D CPU Units Only)

When CIO 000000 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.

1,2,3...

1. The specified error code (80E1) will be written to A400 if it is the most serious error.
2. The error code and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
3. The Too Many I/O Points Flag (A40111) 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.

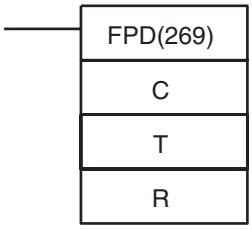


3-30-3 FAILURE POINT DETECTION: FPD(269)

Purpose

Diagnoses a failure in an instruction block by monitoring the time between execution of FPD(269) and execution of a diagnostic output and finding which input is preventing an output from being turned ON.

Ladder Symbol



C: Control word

T: Monitoring time

R: First register word

Variations

Variations		
	Executed Each Cycle for ON Condition	FPD(269)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

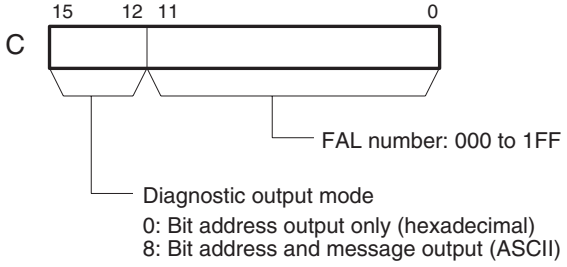
Applicable Program Areas

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

Operands

C: Control Word

C must be a constant between 0000 and 01FF or between 8000 and 81FF. The following diagram shows the function of the digits in the control word.



T: Monitoring Time

T must be between 0 and 9,999 decimal (between 0000 and 270F hex). A value of 0 disables time monitoring; values in the range of 1 to 270F set the monitoring time from 0.1 to 999.9 seconds.

R: First Register Word

The functions of the register words are described on page 1159.

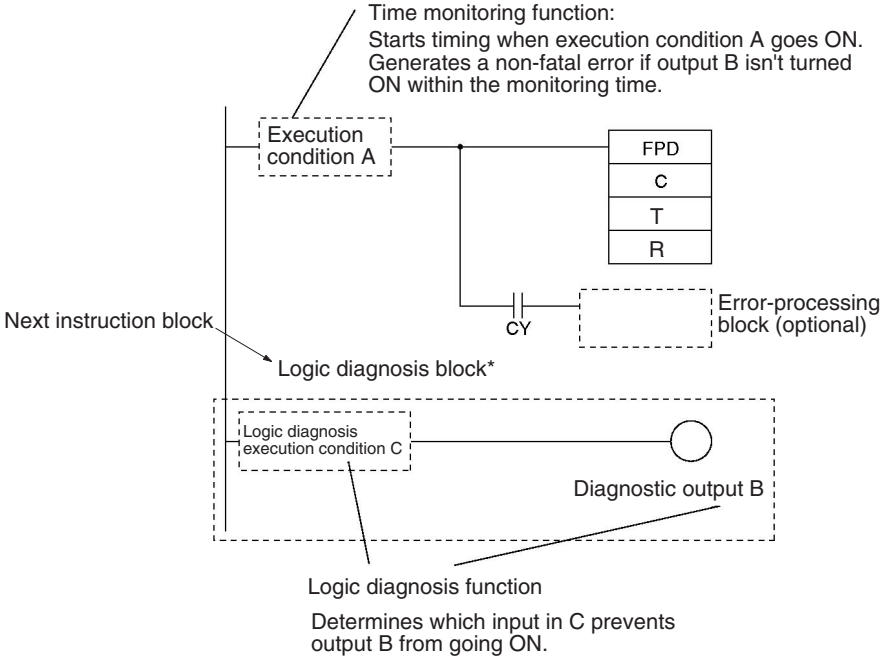
Operand Specifications

Area	C	T	R
CIO Area	---	CIO 0000 to CIO 6143	
Work Area	---	W000 to W511	
Holding Bit Area	---	H000 to H511	
Auxiliary Bit Area	---	A000 to A447 A448 to A959	A448 to A959
Timer Area	---	T0000 to T4095	
Counter Area	---	C0000 to C4095	
DM Area	---	D00000 to D32767	

Area	C	T	R
EM Area without bank	---	E00000 to E32767	
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	Specified values only	#0000 to #270F (binary)	---
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	

Description

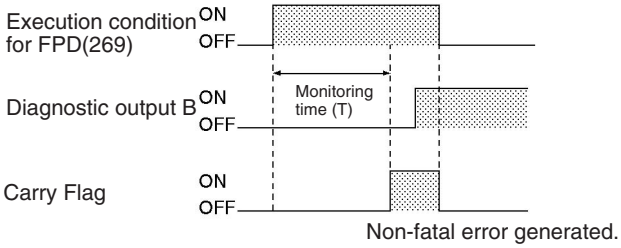
FPD(269) performs time monitoring and logic diagnosis. The time monitoring function generates a non-fatal error with the specified FAL number if the diagnostic output is not turned ON within the specified monitoring time. The logic diagnosis function indicates which input is preventing the output from being turned ON.



Note *The logic diagnosis block begins with the first LD (not LD TR) or LD NOT instruction after FPD(269) and ends with the first OUT (not OUT TR) or other right-hand instruction.

Time Monitoring Function

FPD(269) starts timing when it is executed (when execution condition A goes ON); it will generate a non-fatal error and turn ON the Carry Flag if the diagnostic output is not turned ON within the specified monitoring time.



Note The diagnostic output must go ON within the monitoring time. The teaching function can be used set the monitoring time automatically.

The following processing will be performed when the Carry Flag is turned ON. (This processing will not be performed if the FAL number is set to 000 in C.)

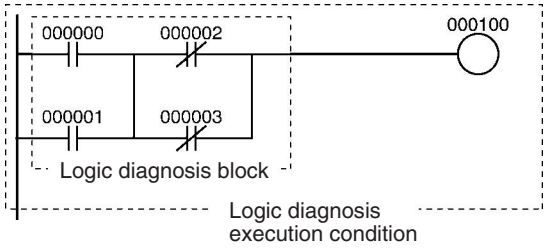
1,2,3...

1. The FAL Error Flag (A40215) will be turned ON. (PLC operation continues.)
2. The Executed FAL Number Flag for the specified FAL number will be turned ON. (Flags A36001 to A39115 correspond to FAL numbers 001 to 1FF.)
3. The corresponding error code will be written to A400. Error codes 4101 to 42FF correspond to FAL numbers 001 to 1FF.
   (If a more serious error has occurred (one with a higher error code) at the same time, the error code of the more serious error will be stored in A400.)
4. The error code and the time/date that the error occurred will be written to the Error Log Area (A100 through A199).
5. The ERR Indicator on the CPU Unit will flash.
6. If the output mode has been set for bit address and message output (leftmost digit of C set to 8), the ASCII message stored in R+2 through R+10 will be displayed as a non-fatal error message.

Logic Diagnosis Function

Every cycle that the execution condition for FPD(269) is ON, FPD(269) determines which input bit is causing the diagnostic output to be OFF and writes the bit's address to the register area beginning at R.

If input bits CIO 000000 through CIO 000003 are all ON in the following example, FPD(269) would determine that the normally closed CIO 000002 condition is causing output CIO 000100 to remain OFF. FPD(269) would turn ON the Bit Address Found Flag (bit 15 of R) and write the bit address to register words R+2 to R+4.



The logic diagnosis function is executed every cycle as long as the execution condition for FPD(269) is ON. The operation of the logic diagnosis function is independent of the time monitoring function.

When two or more input bits are preventing the diagnostic output from being turned ON, the address of the first input bit in the execution condition (on the highest instruction line and nearest the left bus bar) will be output to R+2 through R+4.

Input bits in LD, LD NOT, AND, AND NOT, OR, and OR NOT instructions (including differentiated and immediate-refreshing variations) will be checked by the logic diagnosis function. Input bits in other instructions and operands addressed indirectly through Index Registers will not be checked.

The logic diagnosis block begins with the first LD (not LD TR) or LD NOT instruction after FPD(269) and ends with the first OUT (not OUT TR) or other right-hand instruction.

There are two diagnostic output modes, set with the leftmost digit of C.

1,2,3...

1. Bit address output mode (Leftmost digit of C = 0)

   Bit 15 of R (the Bit Address Found Flag) is turned ON when an input bit address has been found and bit 14 of R indicates whether the input is normally ON or normally OFF.
   The 8-digit hexadecimal PLC memory address of the input bit is output to R+3 and R+2.

2. Bit address and message output mode (Leftmost digit of C = 8)

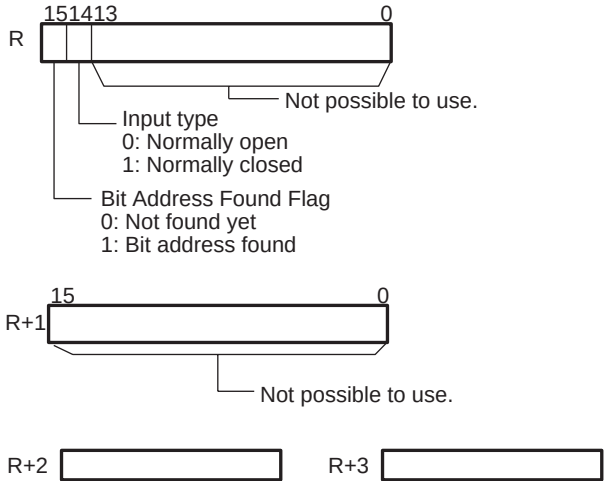
   Bit 15 of R (the Bit Address Found Flag) is turned ON when an input bit address has been found and bit 14 of R indicates whether the input is normally ON or normally OFF.
   The input bit's address is output to R+2 through R+4 as 6 ASCII characters.

Register Word Functions

The register words contain the results of the diagnostic function and can also contain an ASCII error message which is displayed when an error is generated by the time monitoring function. The function of the register words depends upon the diagnostic output mode which is set with the leftmost digit of C.

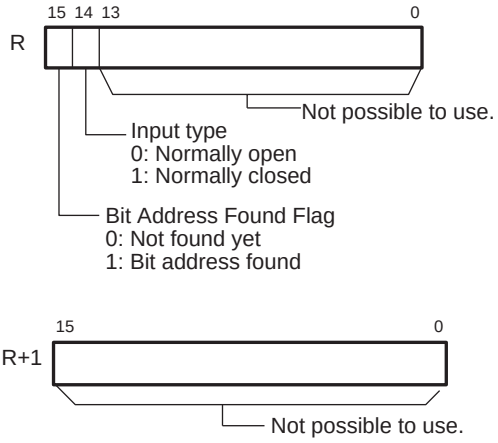
Bit Address Output (C=0□□□)

When the leftmost digit of C is set to 0, the 8-digit hexadecimal PLC memory address of the input bit is output to R+2 and R+3. R contains two flags which indicate whether an input bit has been found and whether it is used in a normally open or normally closed input condition.



Bit Address and Message Output (C=8□□□)

When the leftmost digit of C is set to 8, the ASCII address of the input bit is output to R+2 to R+4. R contains two flags which indicate whether an input bit has been found and whether it is used in a normally open or normally closed input condition.



Register words R+2 to R+4 indicate the address of the input which prevented the diagnostic output from being turned ON. The bit address is output to these words in ASCII. The following table shows the ASCII representations for each area.

Area	ASCII text	Notes
Auxiliary Area	A00000 to A95915	---
Holding Area	H00000 to H51115	---
Work Area	W00000 to W51115	---
CIO Area	000000 to 665515	---
Task Flags	TK0000 to TK0031	---
Timer Area	_T0000 to _T4095	The "_" represents an ASCII space. (Character code 20.)
Counter Area	_C0000 to _C4095	

	15	
R+2	W	5
R+3	1	1
R+4	1	5

Bit address written in ASCII

Register words R+2 through R+5 would have the following values for W51115:

Word	Bits 8 to 15	Bits 0 to 7
R+2	W	5
R+3	1	1
R+4	1	5
R+5	2D (hexadecimal)	Input type (hexadecimal) 30: Normally open 31: Normally closed

The user can store an ASCII message in register words R+6 to R+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).

	15 8	7 0
R+6		
R+7		
R+8		
R+9		
R+10		

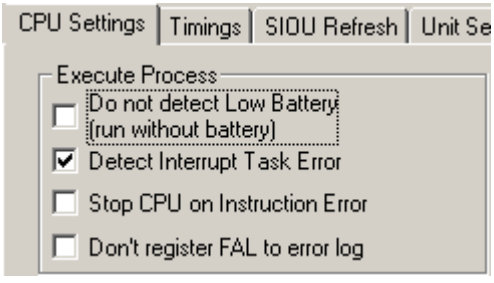
Disabling Error Log Entries of Non-fatal FPD(269) Errors (CS1-H, CJ1-H, CJ1M, or CS1D Only)

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). In CS1-H, CJ1-H, CJ1M, and CS1D CPU Units, 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.

Even though the error will not be recorded in the Error Log, the FAL Error Flag (40215) will be turned ON, the corresponding flag in the Executed FAL Number Flags (A36001 to A39115) 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.



The following table shows the PLC Setup setting from the Programming Console.

Item	Setting	
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)	

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.

1,2,3...

1. Turn ON the FPD Teaching Bit (A59800).
2. FPD(269) will measure the time from the point when the execution condition for FPD(269) goes ON until the diagnostic output is turned ON.
3. If the measured time exceeds the monitoring time setting, a setting 1.5 times the measured time will be stored in T.

Flags

Name	Label	Operation
Error Flag	ER	ON if C is not within the specified range of 0000 to 01FF or 8000 to 81FF. ON if T is not within the specified range of 0000 to 270F. OFF in all other cases.
Carry Flag	CY	ON if the diagnostic output is still OFF after the monitoring time has elapsed. OFF in all other cases.

The following table shows relevant words and flags in the Auxiliary Area.

Name	Address	Operation
FAL Error Flag	A40215	ON when a non-fatal (FAL) error is registered in time monitoring.
Executed FAL Number Flags	A36001 to A39115	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).

Name	Address	Operation
Error code	A400	When an error occurs its error code is stored in A400. The error codes for FAL numbers 0001 to 01FF are 4101 to 42FF, respectively. If two or more errors occur simultaneously, the error code of the most serious error will be stored in A400.
FPD Teaching Bit	A59800	Turn this bit ON when you want the monitoring time to be set automatically (teaching function) when FPD(269) is executed.

Precautions

When the time monitoring function is being used, the execution condition for FPD(269) must remain ON for the entire monitoring time set in T.

The execution condition for FPD(269) must be made up of a combination of normally open and normally closed inputs.

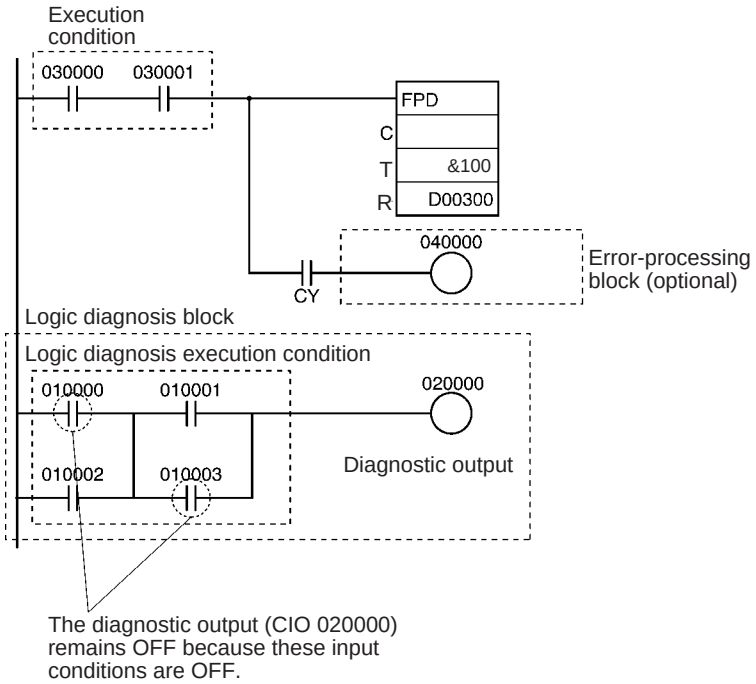
The error-processing block is optional. When an error-processing block is included, be sure to use outputs or other right-hand instructions. LD and LD NOT cannot be used at this point.

FPD(269) can be used more than once in the program, but each instruction must have a unique register (R) setting.

The monitoring time is refreshed only when FPD(269) is executed. If the cycle time is longer than 100 ms, the monitoring time will not be refreshed normally and FPD(269) will not operate correctly because the monitoring time is updated in units of 100 ms.

Examples

The following program example is used to demonstrate the operation of the time monitoring function and logic diagnosis function. In this example, the diagnostic output (CIO 020000) does not go ON because CIO 010000 and CIO 010003 remain OFF in the logic diagnosis execution condition.



Time Monitoring Function

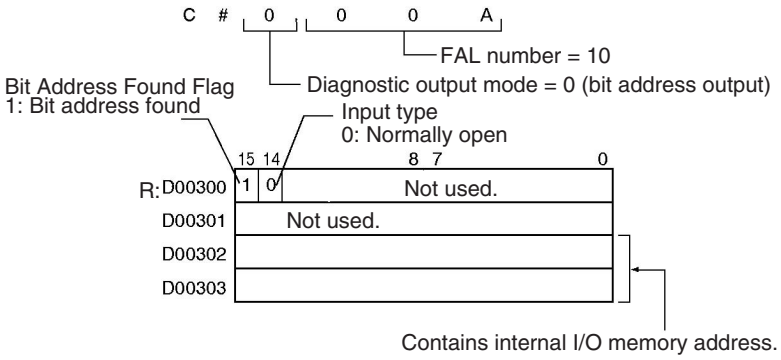
If the diagnostic output (CIO 020000) does not go ON within 10 seconds after CIO 030000 and CIO 030001 are both ON, a non-fatal error will be generated and the following processing will be performed.

1,2,3... 1. The Carry Flag is turned ON.

2. When the rightmost 3 digits of C specify an FAL number of 00A hex (10), the corresponding Executed FAL Number Flag (A36010) will be turned ON, the corresponding error code (410A) is written in A400, and the FAL Error Flag (A40215) is turned ON.

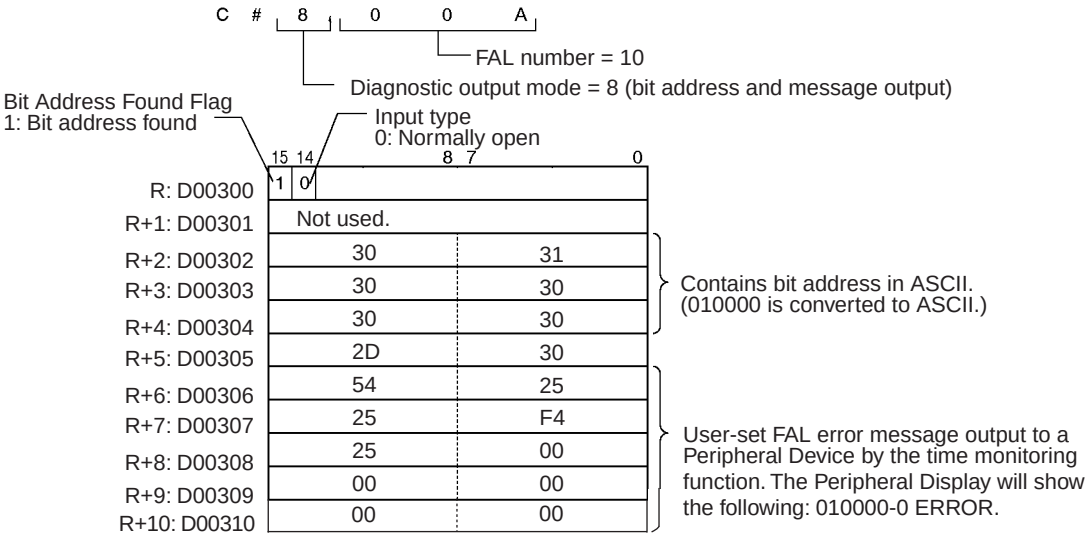
Logic Diagnosis Function (C=000A)

Since the leftmost digit of C is 0 (bit address output mode) the PLC memory address of CIO 010000 is output to D00303 and D00302. (CIO 010000 is on a higher instruction line than CIO 010003.)



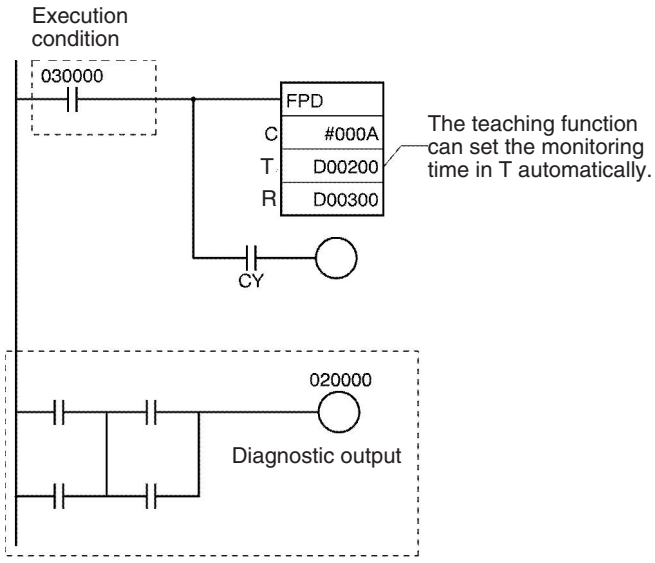
Logic Diagnosis Function (C=800A)

Since the leftmost digit of C is 8 (bit address and message output mode) the address of CIO 010000 (010000) is output to D00302 through D00304 in ASCII.

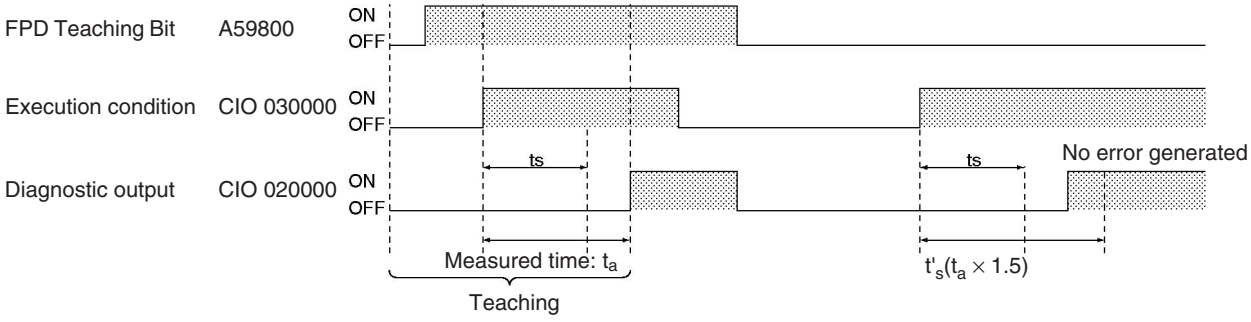


Setting the Monitoring Time with the Teaching Function

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 A59800 (the FPD Teaching Bit). While A59800 is ON, FPD(269) measures how long it takes for the diagnostic output (CIO 020000) to go ON after the execution condition (CIO 030000) goes ON. If the measured time exceeds the monitoring time in T, the measured time is multiplied by 1.5 and that value is stored in T as the new monitoring time.



t_s: Initial setting in T
t_a: Measured time
t'_s: New setting in T after teaching
(When $t_a > t_s$, $t'_s = t_a \times 1.5$)

3-31 Other Instructions

This section describes instructions for manipulating the Carry Flag, selecting the EM bank, and extending the maximum cycle time.

Instruction	Mnemonic	Function code	Page
SET CARRY	STC	040	1166
CLEAR CARRY	CLC	041	1166
SELECT EM BANK	EMBC	281	1167
EXTEND MAXIMUM CYCLE TIME	WDT	094	1169
SAVE CONDITION FLAGS	CCS	282	1171
LOAD CONDITION FLAGS	CCL	283	1173
CONVERT ADDRESS FROM CV	FRMCV	284	1174
CONVERT ADDRESS TO CV	TOCV	285	1179

Instruction	Mnemonic	Function code	Page
DISABLE PERIPHERAL SERVICING	IOSP	287	1183
ENABLE PERIPHERAL SERVICING	IORS	288	1185

3-31-1 SET CARRY: STC(040)

Sets the Carry Flag (CY).

Ladder Symbol

Variations

Variations	**Executed Each Cycle for ON Condition**	STC(040)
	Executed Once for Upward Differentiation	@STC(040)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Description

When the execution condition is ON, STC(040) turns ON the Carry Flag (CY). Although STC(040) turns the Carry Flag ON, the flag will be turned ON/OFF by the execution of subsequent instructions which affect the Carry Flag.

Flags

Name	Label	Operation
Error Flag	ER	Unchanged (See note.)
Equals Flag	=	Unchanged (See note.)
Carry Flag	CY	ON
Negative Flag	N	Unchanged (See note.)

Note In 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.

Precautions

ROL(027), ROLL(572), ROR(028), and RORL(573) make use of the Carry Flag in their rotation shift operations. When using any of these instructions, use STC(040) and CLC(041) to set and clear the Carry Flag.

3-31-2 CLEAR CARRY: CLC(041)

Purpose

Turns OFF the Carry Flag (CY).

Ladder Symbol

Variations

Variations	**Executed Each Cycle for ON Condition**	CLC(041)
	Executed Once for Upward Differentiation	@CLC(041)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Description

When the execution condition is ON, CLC(040) turns OFF the Carry Flag (CY). Although CLC(040) turns the Carry Flag OFF, the flag will be turned ON/OFF by the execution of subsequent instructions which affect the Carry Flag.

Flags

Name	Label	Operation
Error Flag	ER	Unchanged (See note.)
Equals Flag	=	Unchanged (See note.)
Carry Flag	CY	OFF
Negative Flag	N	Unchanged (See note.)

Note In 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.

Precautions

+C(402), +CL(403), +BC(406), and +BCL(407) make use of the Carry Flag in their addition operations. Use CLC(041) just before any of these instructions to prevent any influence from other preceding instructions.

–C(412), –CL(413), –BC(416), and –BCL(417) make use of the Carry Flag in their subtraction operations. Use CLC(041) just before any of these instructions to prevent any influence from other preceding instructions.

ROL(027), ROLL(572), ROR(028), and RORL(573) make use of the Carry Flag in their rotation shift operations. When using any of these instructions, use STC(040) and CLC(041) to set and clear the Carry Flag.

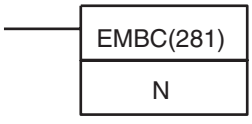
Note The +(400), +L(401), +B(404), +BL(405), –(410), –L(411), –B(414), and –BL(415) instructions do no include the Carry Flag in their addition and subtraction operations. In general, use these instructions when performing addition or subtraction.

3-31-3 SELECT EM BANK: EMBC(281)

Purpose

Changes the current EM bank.

Ladder Symbol



N: EM bank number

Variations

Variations		
	Executed Each Cycle for ON Condition	EMBC(281)
	Executed Once for Upward Differentiation	@EMBC(281)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

N: EM Bank Number
Specifies the new EM bank number in hexadecimal (0000 to 000C).

Operand Specifications

Area	N
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A000 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767
Constants	#0000 to #000C (binary)
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

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.

There are up to 13 banks (0 to C) 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.

1,2,3...

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 = 0 to C), to indicate addresses in a particular EM bank.

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the range 0000 to 000C. 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.) OFF in all other cases.

The following table shows relevant flags in the Auxiliary Area.

Name	Address	Operation
Current EM Bank	A301	Contains the current EM bank number in hexadecimal (0000 to 000C).

Precautions

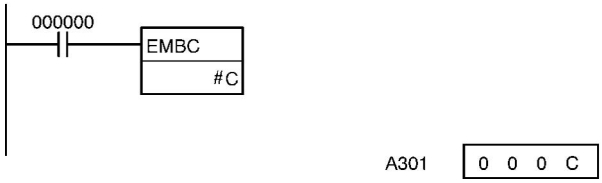
The current EM bank number changed in a cyclic task is retained when operation is switched between tasks. For example, if EMBC(281) is used in task 1 to change the current EM bank from bank B to bank C, bank C will remain the current EM bank for all cyclic tasks even when operation is switched to task 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.

An error will occur if the specified EM bank has been registered as file memory in the PLC Setup.

Examples

When CIO 000000 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.

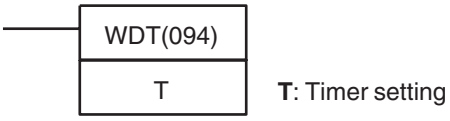


3-31-4 EXTEND MAXIMUM CYCLE TIME: WDT(094)

Purpose

Extends the maximum cycle time, but only for the cycle in which the instruction is executed. WDT(094) can be used to prevent errors for long cycle times when a longer cycle time is temporarily required for special processing.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	WDT(094)
	Executed Once for Upward Differentiation	@WDT(094)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

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

Operands

T: Timer Setting
Specifies the watchdog timer setting between 0000 and 0F9F hexadecimal or between &0000 and &3999 decimal.

Operand Specifications

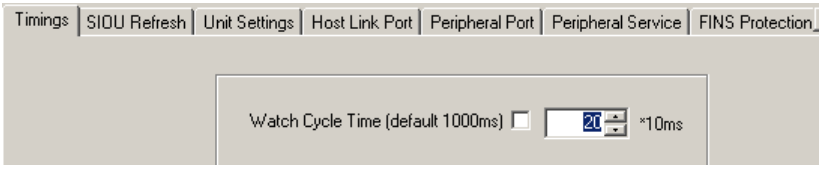
Area	T
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---

Area	T
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	0000 to 0F9F (binary)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Description

WDT(094) extends the maximum cycle time for the cycle in which this instruction is executed. The watchdog timer setting in the PLC Setup is extended by an interval of T × 10 ms (0 to 39,990 ms).

The following screen capture shows the PLC Setup setting from the CX-Programmer.



The following table shows the watchdog timer settings in the PLC Setup. The default value for the maximum cycle time is 1,000 ms, although it can be set anywhere from 1 to 40,000 ms in 10-ms units.

Name	Function	Settings
Watch cycle time	A Cycle Time Too Long error (fatal error) will be registered if the cycle time exceeds the maximum setting.	0: Default setting (1,000 ms) 1: User time setting
	Sets the maximum cycle time. (This setting is valid only when the first setting has been set to 1.)	0001 to 0FA0 (1 to 40,000 ms, 10-ms units)

Flags

Name	Label	Operation
Error Flag	ER	ON if the watchdog timer setting exceeds 40 seconds. OFF in all other cases.

The following table shows relevant flags and words in the Auxiliary Area.

Name	Address	Operation
Cycle Time Too Long Flag	A40108	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.

Precautions

WDT(094) can be used more than once in a cycle. When WDT(094) is executed more than once the cycle time extensions are added together, although

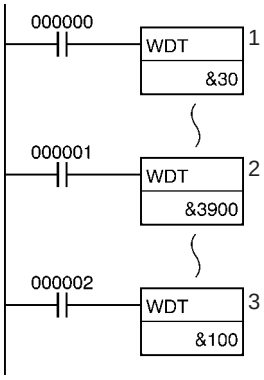
the total must not exceed 40,000 ms. If WDT(094) cannot be executed again if the cycle has already been extended to 40,000 ms.

Examples

The default maximum cycle time (1,000 ms) is used in this example.

1,2,3...

1. When CIO 000000 turns ON, the first WDT(094) instruction extends the maximum cycle time by 300 ms (30 × 10 ms). Thus, the maximum cycle time is 1,300 ms at this point.
2. When CIO 000001 turns ON, the second WDT(094) instruction attempts to extend the maximum cycle time by another 39,000 ms. Since the new maximum cycle time (40,300 ms) exceeds the upper limit of 40,000 ms, the extra 300 ms is ignored. As a result, the second WDT(094) instruction actually extends the maximum cycle time by 38,700 ms.
3. When CIO 000002 turns ON, the third WDT(094) instruction attempts to extend the maximum cycle time by another 1,000 ms. Since the maximum cycle time has already reached the upper limit of 40,000 ms, the third WDT(094) instruction is not executed.

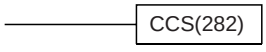


3-31-5 SAVE CONDITION FLAGS: CCS(282)

Saves the current status of the Condition Flags in a separate area within the CPU Unit. The current status of the Flags is preserved so that it can be read (restored) with CCL(283) at a different location in the program, in a different task, or even in a later cycle.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	CCS(282)
	Executed Once for Upward Differentiation	@CCS(282)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

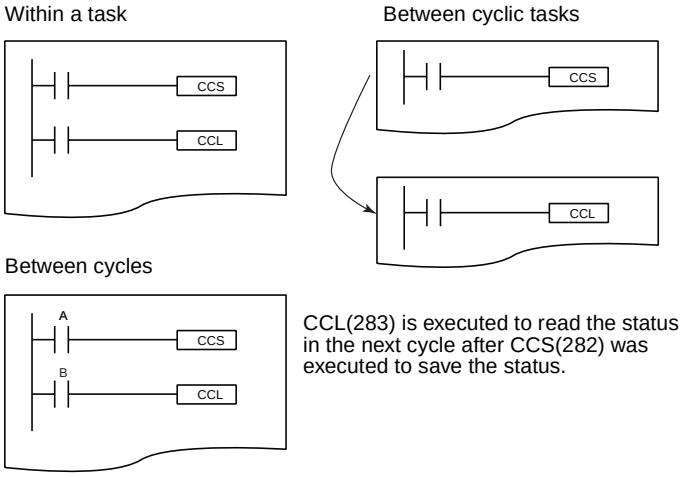
Description

When the execution condition is ON, CCS(282) stores the current status of the Condition Flags (except for the ALWAYS ON and ALWAYS OFF Flags) in

a separate area in the CPU Unit. The Status of the following Condition Flags will be preserved: ER, CY, >, =, <, N, OF, UF, >=, <>, and <=.

The preserved status of the Condition Flags can be read (restored) later only with CCL(283), the LOAD CONDITION FLAGS instruction. The status can be read in any of the following cases:

- Within a task
- Between different cyclic tasks
- Between cycles



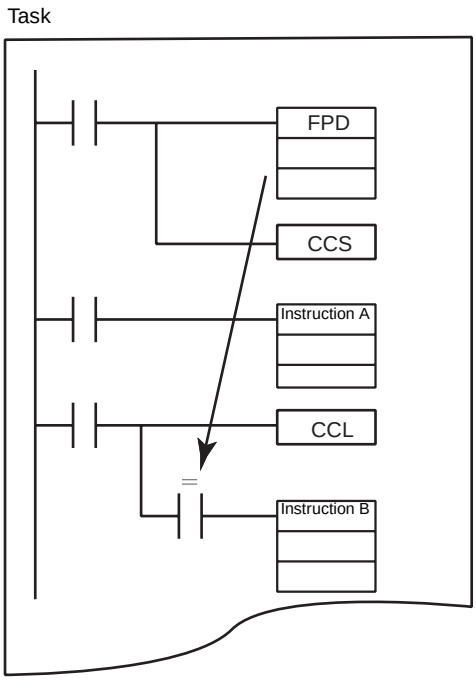
Note

1. The status of the Condition Flags cannot be saved/loaded between a cyclic task and interrupt task.
2. When CCS(282) is executed, it overwrites the previous Condition Flag information that was saved.

All of the Condition Flags are cleared when operation switches from one task to another. Use the CCS(282) and CCL(283) instructions to save and load the Condition Flag status between tasks or cycles.

For example, the CCS(282) and CCL(283) instructions make it possible to use the CY Flag status (time monitoring diagnosis error) from the execution of

FPD(269) at a later point in the program, not immediately after execution of the instruction.



The results of the comparison are stored in the Condition Flags. (In this case, the results of the COMPARE Instruction can be used in instruction B even if those results are affected by execution of instruction A.)

Preserves the status of the Condition Flags in a separate location in the CPU Unit.

Restores the status of the Condition Flags.

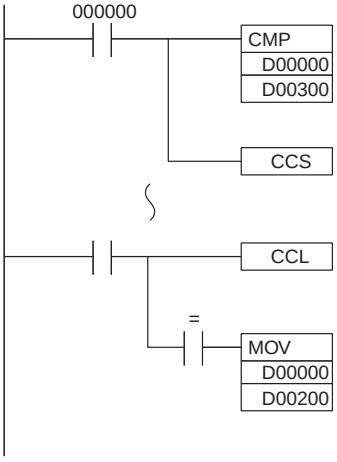
The Equals Flag will reflect the result of the COMPARE instruction, not the result of instruction A.

Flags

There are no flags affected by these instructions.

Examples

In the following example, CCS(282) preserves the results of a Comparison so that this result can be used as an execution condition later in the program.



When CIO 000000 is ON in the following example, CMP(020) will compare the contents of D00000 and D00300 and those results will be preserved by CCS(282).

The preserved Condition Flags are restored by CCL(283).

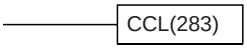
This MOV(021) instruction is executed if the result of the CMP(020) instruction caused the Equals flag to be turned ON.

3-31-6 LOAD CONDITION FLAGS: CCL(283)

Restores the status of the Condition Flags that were saved in a separate area within the CPU Unit by CCS(282). It is also possible to use CCL(283) independently to clear the Condition Flags.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	CCL(283)
	Executed Once for Upward Differentiation	@CCL(283)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

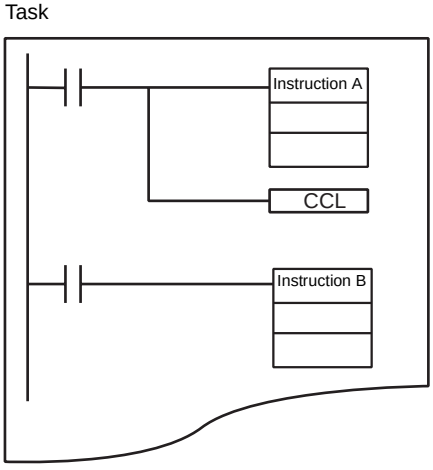
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description

When the execution condition is ON, CCL(283) restores (reads) the status of the Condition Flags (except for the ALWAYS ON and ALWAYS OFF Flags). The Status of the following Condition Flags will be restored (read): ER, CY, >, =, <, N, OF, UF, >=, <>, and <=.

Condition Flags are shared by all instructions, so the status of these Flags may change many times during the PLC cycle as each instruction is executed. Previously, it was necessary to place conditions using the Condition Flags immediately after the controlling instruction so that the status of the Condition Flags would not be affected by intervening instructions. The CCS(282) and CCL(283) instructions allow the controlling instruction to be separated from the execution conditions that rely on the result.

For example, CCS(282) can store the status of the Equals Flag after execution of a Comparison Instruction and the result can be restored later. The result does not have to be used immediately after execution of the instruction.



CCL(283) is used alone to clear the Condition Flags after execution of instruction A so that those results do not affect instruction B and later instructions.

Refer to *3-31-5 SAVE CONDITION FLAGS: CCS(282)* for more examples showing how to use CCS(282) and CCL(283).

Flags

There are no flags affected by these instructions.

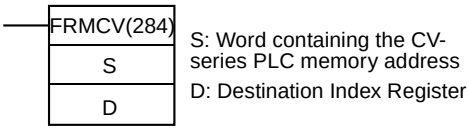
3-31-7 CONVERT ADDRESS FROM CV: FRMCV(284)

Purpose

Converts a CV-series PLC memory address to its corresponding CS/CJ-series PLC memory address. FRMCV(284) can be useful when converting CV-series programs that use PLC memory addresses so that they are compatible with CS/CJ-series PLCs.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	FRMCV(284)
	Executed Once for Upward Differentiation	@FRMCV(284)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

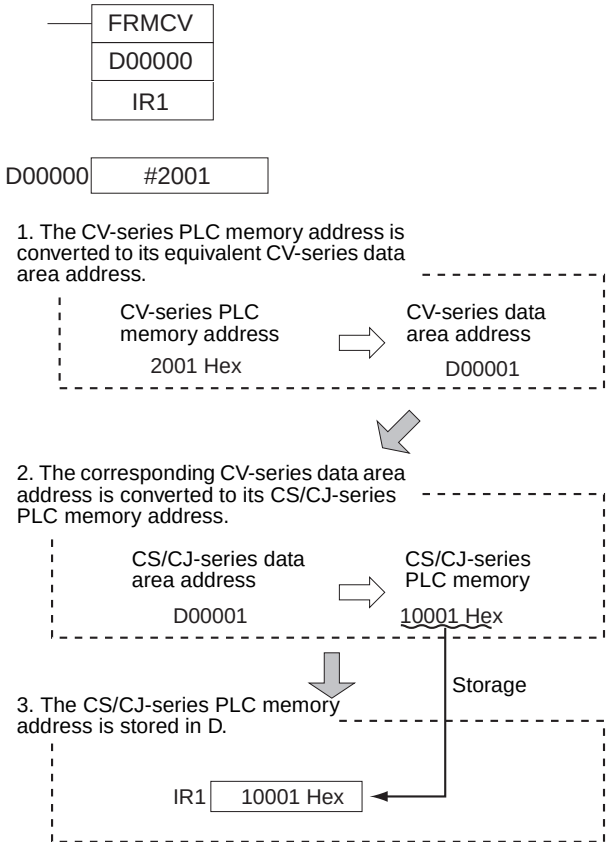
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

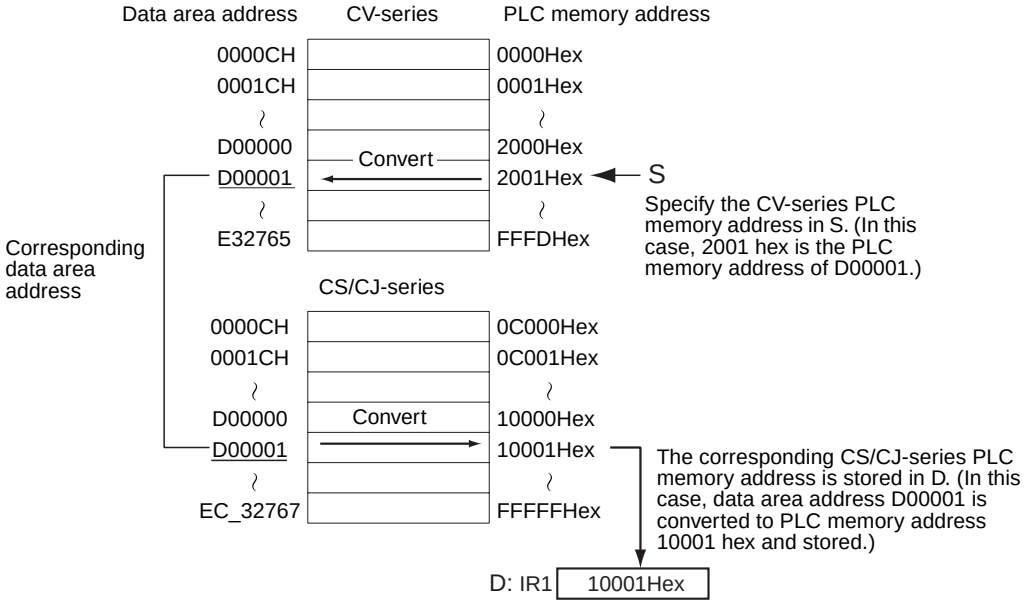
Description

When the execution condition is ON, FRMCV(284) executes the following operations.

1. The CV-series PLC memory address specified in S is converted to its equivalent CV-series data area address.
2. FRMCV(284) determines the 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.)

The following example shows FRMCV(284) used to convert the CV-series PLC memory address for D00001.





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.

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))

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	---
Work Area	W000 to W511	---
Holding Bit Area	H000 to H511	---
Auxiliary Bit Area	A448 to A959	---
Timer Area	T0000 to T4095	---
Counter Area	C0000 to C4095	---
DM Area	D00000 to D32767	---
EM Area without bank	E00000 to E32767	---
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	---

Area	S	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	---
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	---
Constants	Any constant except 09FF hex, 0A00 to 0AFF hex, or 0D00 to 0E3F hex	---
Data Registers	DR0 to DR15	---
Index Registers	---	IR0 to IR15
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15	---

Flags

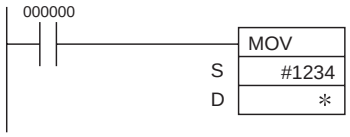
Name	Label	Operation
Error Flag	ER	ON if S specifies one of the following PLC memory addresses that do not exist in the CS/CJ-series: Temporary Relay (TR) Area (09FF hex) CPU Bus Link (G) Area (0A00 to 0AFF hex) SFC Areas (0D00 to 0E3F hex) OFF in all other cases.

Examples

Example 1: Converting a CV-series Program with ✳DM Indirect Binary Mode DM Addressing

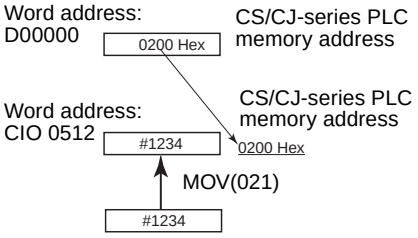
In this FRMCV(284) example, a DM word is specified in S, the PLC memory address there is stored in an Index Register, and the Index Register is used for indirectly addressed.

- CV-series program (Program using indirect DM binary mode addressing)



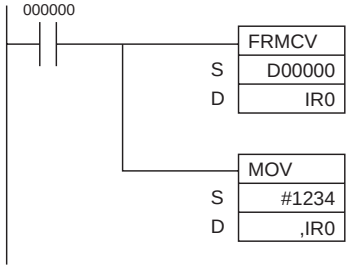
PLC Setup
Indirect DM data:
When indirect DM addresses are in binary, the content of the DM word is treated as a PLC memory address and specifies the corresponding address in I/O memory.

In this case, the value in D00000 is 0200 hex. The corresponding data area address is CIO 0512, so #1234 is transferred to CIO 0512.

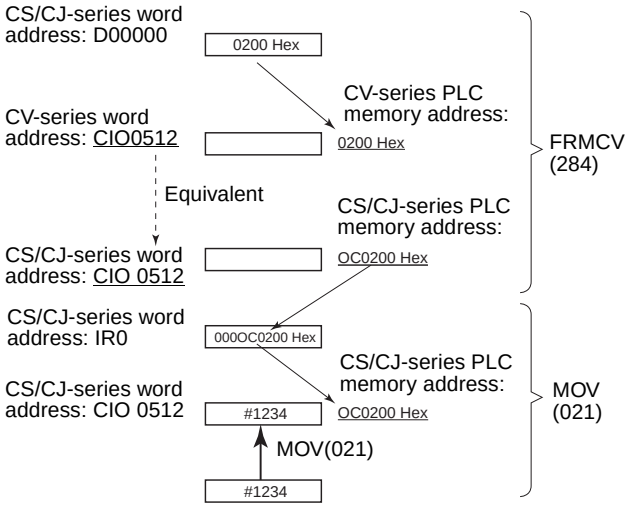


Equivalent program

- CS/CJ-series program



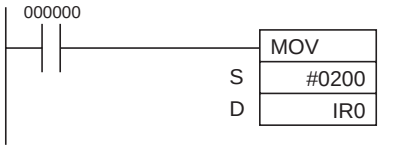
In this case, the value in D00000 is 0200 hex. The corresponding CV-series data area address is CIO 0512. The CS/CJ-series PLC memory address for CIO 0512 is 0000C200 hex, so this value is stored in IR0. The destination operand in MOV(021) indirectly addresses the content of IR0, so #1234 is transferred to CIO 0512.



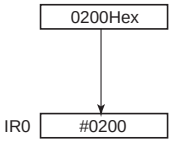
Example 2: Converting a CV-series Program with PLC Memory Addresses Stored directly in Index Registers

In this FRMCV(284) example, the CV-series PLC memory address is specified directly in S.

- CV-series program
  (Program using PLC memory addresses stored directly in IR)

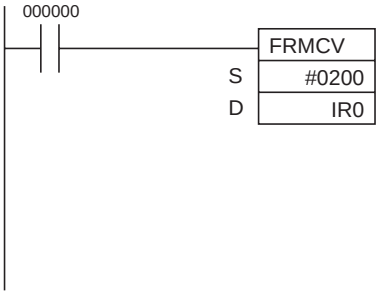


In this case, the PLC memory address 0200 hex is stored in Index Register IR0.

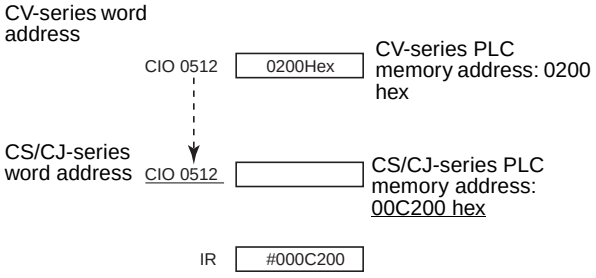


Equivalent program

- CS/CJ-series program



In this case, the CV-series PLC memory address 0200 hex corresponds to CIO 0512. The CS/CJ-series PLC memory address for CIO 0512 is 0000C200 hex, so this value is stored in IR0.



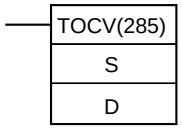
3-31-8 CONVERT ADDRESS TO CV: TOCV(285)

Purpose

Converts a CS/CJ-series PLC memory address to its corresponding CV-series PLC memory address. TOCV(285) can be useful when converting CS/CJ-series programs that use PLC memory addresses so that they are compatible with CV-series PLCs.

This instruction is supported by CS1-H, CJ1-H, CJ1M, and CS1D CPU Units only.

Ladder Symbol



S: Index Register containing the CS/CJ-series PLC memory address

D: Destination word

Variations

Variations	**Executed Each Cycle for ON Condition**	TOCV(285)
	Executed Once for Upward Differentiation	@TOCV(285)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

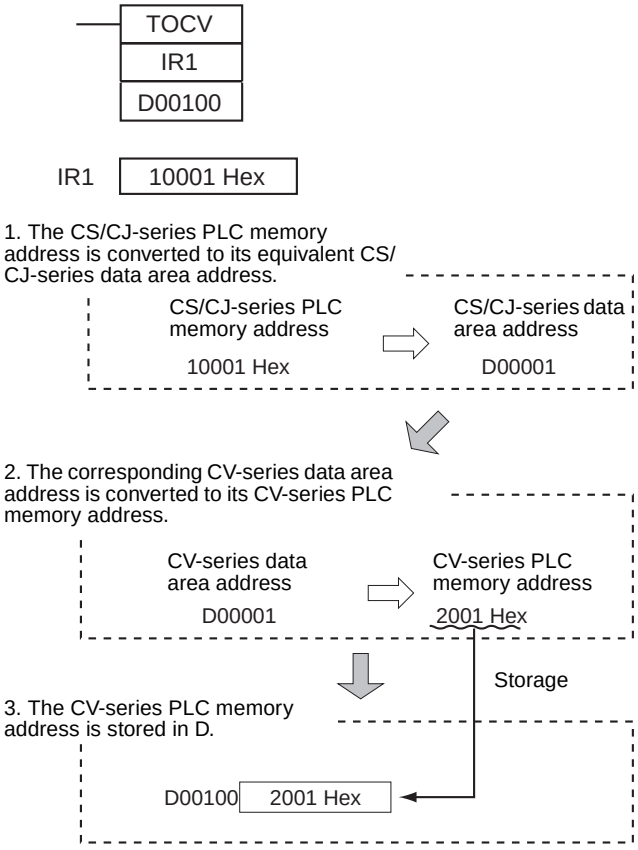
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

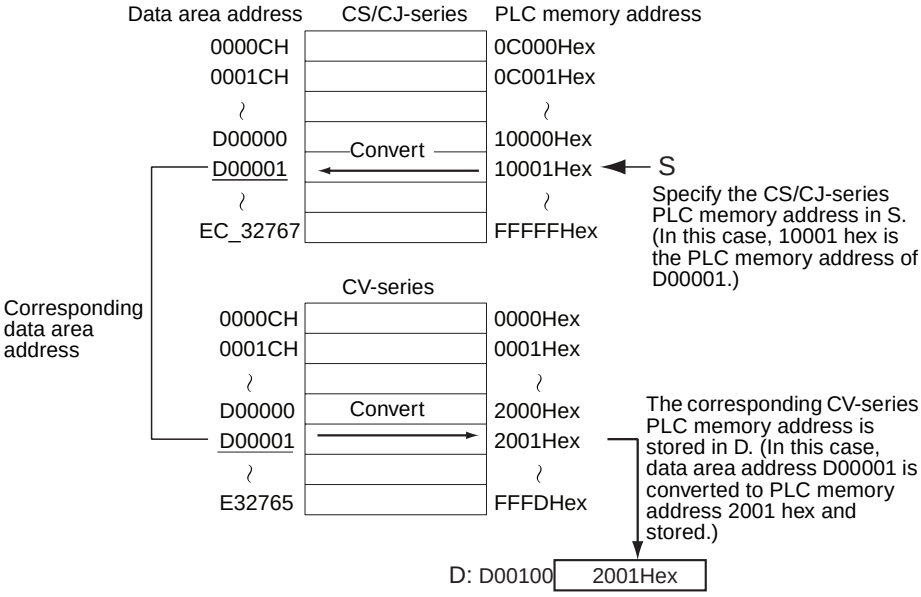
Description

When the execution condition is ON, TOCV(285) executes the following operations.

1. The 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 D00001.





Note

1. 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.
2. The CV-series PLC memory address data stored by TOCV(285) can be transferred to a CV-series PLC using CX-Programmer.
3. The same data area address that was used in the 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).

Operand Specifications

Area	S	D
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A448 to A959
Timer Area	---	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	See note 1.	---
Data Registers	---	DR0 to DR15

Area	S	D
Index Registers	IR0 to IR15	---
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15

Note

1. An error will occur and the Error Flag will be turned ON if S specifies one of the following PLC memory addresses that do not exist in the CV-series:

Area or addresses	PLC memory addresses
Task Flag Area	0000 B800 to 0000 B801 hex
A512 to A959	0000 BA40 to 0000 BBFF hex
CIO 2556 to CIO 6143	0000 C9FC to 0000 D7FF hex
T1024 to T4095	0000 BE40 to 0000 BEFF hex and 0000 E400 to 0000 EFFF hex
C1024 to C4095	0000 BF40 to 0000 BFFF hex and 0000 F400 to 0000 FFFF hex
HR Area	0000 D800 to 0000 D9FF hex
WR Area	0000 DE00 to 0000 DFFF hex
D24576 to D32767	0001 6000 to 0001 7FFF hex
EM bank specification	0001 8000 to 000F 7FFF hex
E32766 to D32767	000F FFFE to 000F FFFF hex

2. An error will occur and the Error Flag will be turned ON if an area other than the Index Register Area is specified for S.

Flags

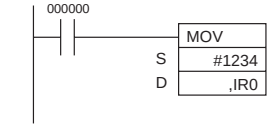
Name	Label	Operation
Error Flag	ER	ON if S specifies a PLC memory address that does not exist in the CV-series PLCs. ON if S is not a constant or Index Register. OFF in all other cases.

Example

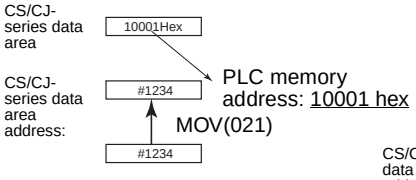
Converting a CS/CJ-series Program with Indirect Index Register Addressing

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.

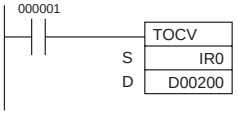
- CS/CJ-series program (Program using indirect Index Register addressing)



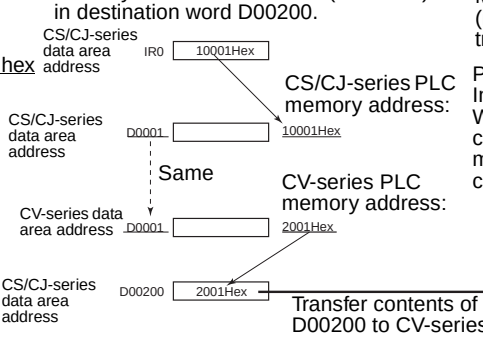
In this case, IR0 contains 10001 hex. The data area address corresponding to PLC memory address 10001 hex is D00001, so #1234 is transferred to D00001.



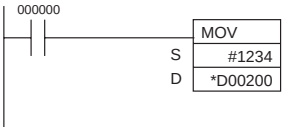
- CS/CJ-series program



In this case, IR0 contains 10001 hex. Since the data area address corresponding to CS/CJ-series PLC memory address 10001 hex is D00001, TOCV(285) stores the CV-series PLC memory address for D00001 (2001 hex) in destination word D00200.



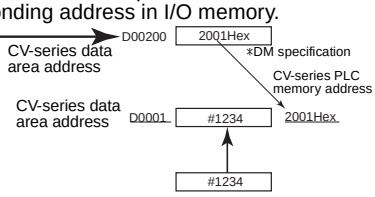
- CV-series program



Transfer contents of D00200 to CV-series.

In the CV-series PLC, the destination of the MOV(021) instruction is indirectly addressed (in binary mode) through D00200, so #1234 is transferred to D00001.

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.



3-31-9 DISABLE PERIPHERAL SERVICING: IOSP(287) (CS1-H/CJ1-H/ CJ1M Only)

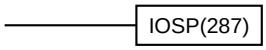
Purpose

Disables peripheral servicing during program execution in Parallel Processing Mode or Peripheral Servicing Priority Mode.

For details on the Parallel Processing Mode and Peripheral Servicing Priority Mode, refer to *Section 6 Advanced Functions* in the *CS/CJ PLC Programming Manual*.

Note This instruction is supported by CS1-H, CJ1-H, and CJ1M CPU Units only. It cannot be used with CS1, CJ1, or CS1D CPU Units.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	IOSP(287)
	Executed Once for Upward Differentiation	@IOSP(287)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

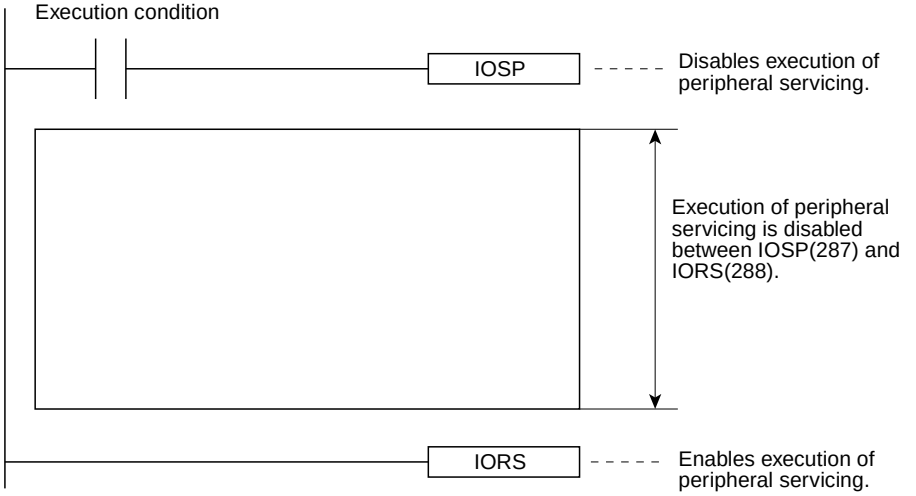
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	Not allowed

Description

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.



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.

Flags

Name	Label	Operation
Error Flag	ER	ON if IOSP(287) is executed in an interrupt task. OFF in all other cases.

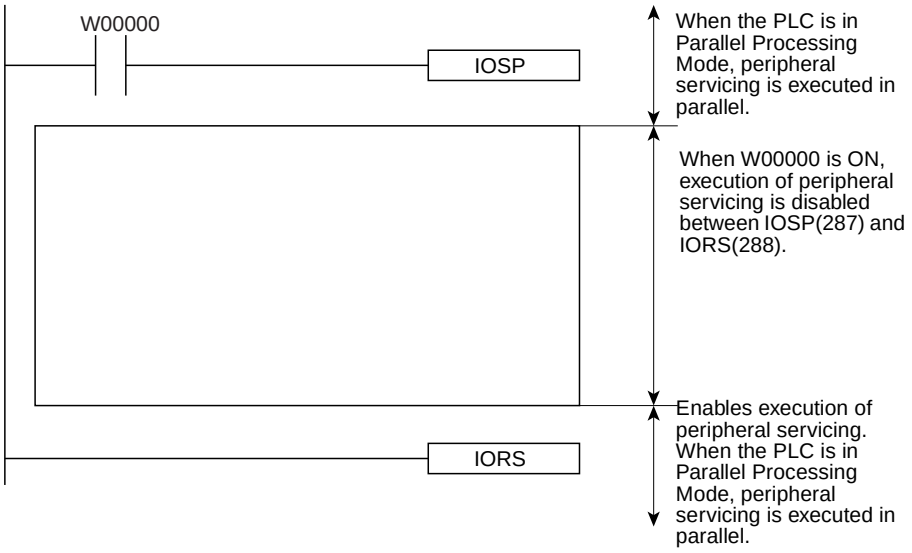
Precautions

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.

Example

The following example shows IOSP(287) and IORS(288) used to disable peripheral servicing in a program section.



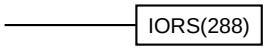
3-31-10 ENABLE PERIPHERAL SERVICING: IORS(288) (CS1-H/CJ1-H/ CJ1M Only)

Purpose

Enables the peripheral servicing during program execution in Parallel Processing Mode that was disabled by IOSP(287), the DISABLE PERIPHERAL SERVICING instruction.

This instruction is supported by CS1-H, CJ1-H, and CJ1M CPU Units only.

Ladder Symbol



Variations

Variations	**Executed Each Cycle for ON Condition**	IORS(288)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	Not allowed

Description

Use IORS(288) in a cyclic task to release the prohibition on peripheral servicing by IOSP(287), the DISABLE PERIPHERAL SERVICING instruction.

It is not necessary to program IORS(288) with an execution condition.

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.

Flags

Name	Label	Operation
Error Flag	ER	ON if IORS(288) is executed in an interrupt task. OFF in all other cases.

3-32 Block Programming Instructions

This section describes block programs and the block programming instructions.

Instruction	Mnemonic	Function code	Page
BLOCK PROGRAM BEGIN	BPRG	096	1191
BLOCK PROGRAM END	BEND	801	1191
BLOCK PROGRAM PAUSE	BPPS	811	1193
BLOCK PROGRAM RESTART	BPRS	812	1193
CONDITIONAL BLOCK EXIT (NOT)	EXIT (NOT)	806	1199
IF (NOT)	IF (NOT)	802	1196
ELSE	ELSE	803	1196
IF END	IEND	804	1196
ONE CYCLE AND WAIT (NOT)	WAIT (NOT)	805	1202
HUNDRED-MS TIMER WAIT	TIMW (BCD)	813	1206
	TIMWX (binary)	816	
COUNTER WAIT	CNTW (BCD)	814	1209
	CNTWX (binary)	818	
TEN-MS TIMER WAIT	TMHW (BCD)	817	1212
	TMHWX (binary)	815	
LOOP	LOOP	809	1215
LOOP END (NOT)	LEND (NOT)	810	1215

3-32-1 Introduction

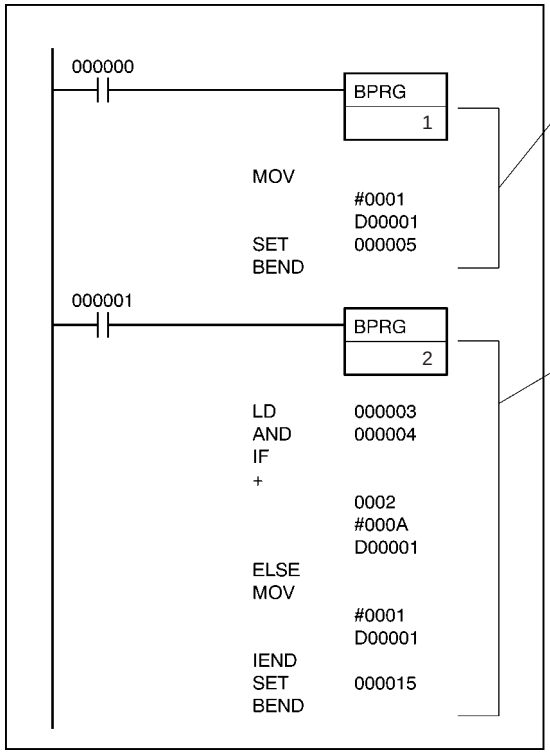
Block Programs

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.

The following example shows two block programs.



Block program area No. 1

When CIO 000000 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 000001 is ON, the contents of block program 2 will be executed. If CIO 000003 and CIO 000004 are both ON, the binary addition will be performed (CIO 0002 + #000A→D00001). If one or both of these bits is OFF, #0001 will be moved to D00001. CIO 000015 will then be set unconditionally and the block program will end.

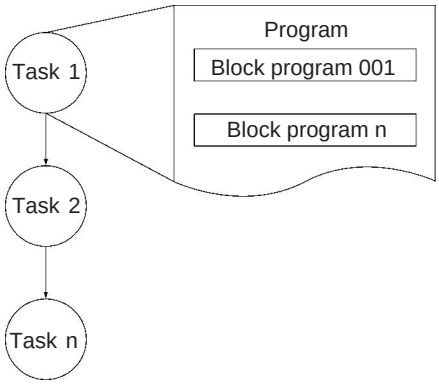
Tasks and Block Programs

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.

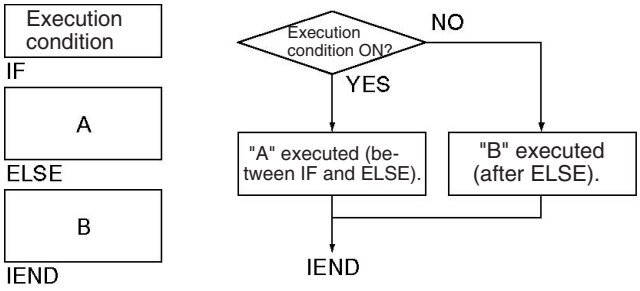
Block programs can be used either within cyclic tasks or interrupt tasks. Each block program number from 0 to 127 can be used only once and cannot be use again, even in a different task.



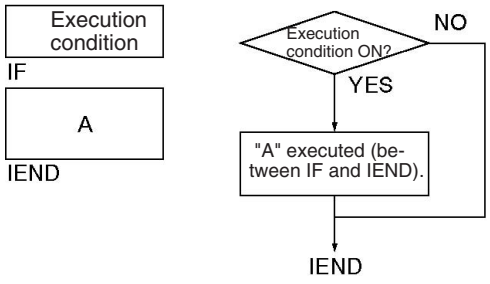
Using Block Programming Instructions

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

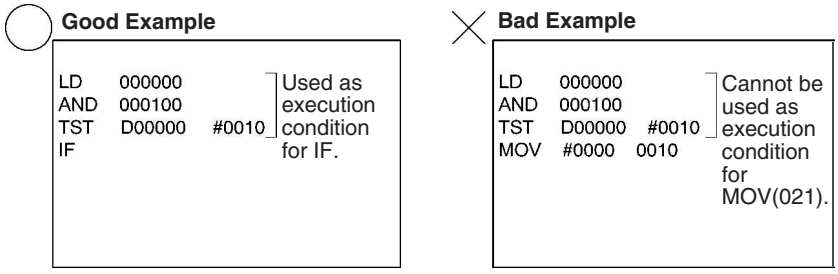
The following instruction can take execution conditions within a block program.

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.

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



Instructions Not Applicable in Block Programs

The instructions listed in the following table cannot be used within block programs.

Instruction group	Mnemonic	Name	Alternative
Sequence Output Instructions	OUT	OUTPUT	Use SET and RSET.
	OUT NOT	OUTPUT NOT	
	DIFU(013)	DIFFERENTIATE UP	None
	DIFD(014)	DIFFERENTIATE DOWN	None
	KEEP(011)	KEEP	None

Instruction group	Mnemonic	Name	Alternative
Sequence Control Instructions	FOR(512) and NEXT(513)	FOR-NEXT LOOPS	Use LOOP(809) and LEND(810) (NOT).
	BREAK(514)	BREAK LOOP	
	IL(002) and ILC(003)	INTERLOCK and INTERLOCK CLEAR	Divide the block program into smaller blocks.
	JMP(004)0 and JME(005) 0	Multiple JUMP and Multiple JUMP END	Use JMP(004 and JME(005) (but the jump will be made unconditionally).
	END(001)	END	Use BEND(801).
Timer and Counter Instructions	TIM 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 TIMHHX(552)	ONE-MS TIMER	
	TIMU(541) and TIMUX(556)	TENTH-MS TIMER (CJ1-H-R CPU Units only)	
	TIMUH(544) and TIMUHX (557)	HUNDREDTH-MS TIMER (CJ1-H-R CPU Units only)	
	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

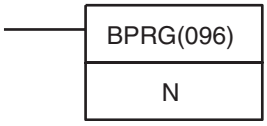
3-32-2 BLOCK PROGRAM BEGIN/END: BPRG(096)/BEND(801)

Purpose

Define a block programming area. For every BPRG(096) there must be a corresponding BEND(801).

Ladder Symbols

BLOCK PROGRAM BEGIN



N: Block program number

BLOCK PROGRAM END

BEND(801)

Variations

BPRG(096)

Variations	**Executed Each Cycle for ON Condition**	BPRG(096)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

BEND(801)

Variations	**Always Executed in Block Program**

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
(See note.)	OK	OK	OK

Note BPRG(096) is allowed only once at the beginning of each block program.

Operands

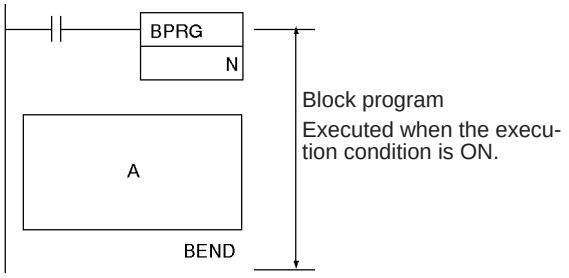
N: Block Program Number
The block program number must be between 0 and 127 decimal.

Operand Specifications (BPRG(096))

Area	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	0 to 127 (decimal)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Description

BPRG(096) executes the block program with the block number designated in N, i.e., the one immediately after it and ending with BEND(801). All instructions between BPRG(096) and BEND(801) are executed with ON execution conditions (i.e., unconditionally).



When the execution condition for BPRG(096) is OFF, the block program will not be executed and no execution time will be required for the instruction in the block program.

Execution of the block program can be stopped using BPPS(811) from within another block program even if the execution condition for BPRG(096) is ON.

Flags

BPRG(096)

Name	Label	Operation
Error Flag	ER	ON if BPRG(096) is already being executed. ON if N is not between 0 and 127. ON if the same block program number is used more than once. OFF in all other cases.

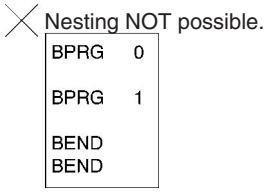
BEND(801)

Name	Label	Operation
Error Flag	ER	ON if a block program is not being executed. OFF in all other cases.

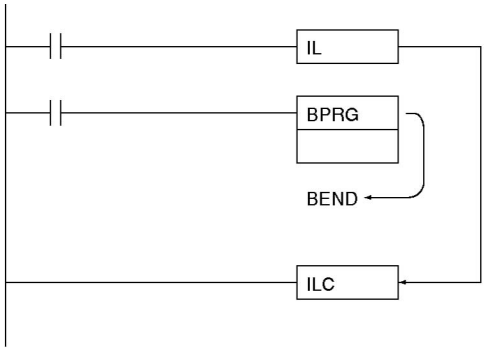
Precautions

Each block program number can be used only once within the entire user program.

Block programs cannot be nested.



If the block program is in an interlocked program section and the execution condition for IL(002) is OFF, the block program will not be executed.

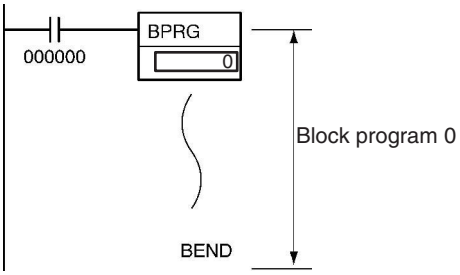


BPRG(096) and the corresponding BEND(801) must be in the same task.

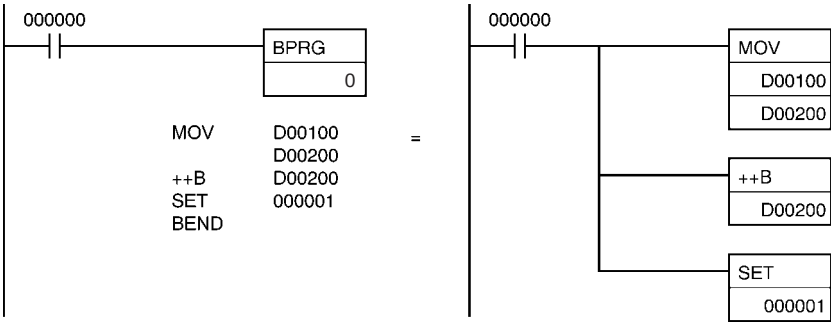
An error will occur and the Error Flag will turn ON if BPRG(096) is in the middle of a block program, BEND(801) is not in a block program, N is not between #0000 and #007F (binary), there is no block program, or if the same block program number is used more than once.

Examples

When CIO 000000 turns ON in the following example, block program 0 will be executed. When CIO 000000 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 000000 turns ON).



3-32-3 BLOCK PROGRAM PAUSE/RESTART: BPPS(811)/BPRS(812)

Purpose

Pause and restart the specified block program from another block program.

Ladder Symbol

BPPS(811)	N	**N**: Block program number
BPRS(812)	N	

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Note BPRG(096) and BPRS(812) must be used in block programming regions even within subroutines and interrupt tasks.

Operands

N: Block Program Number
The block program number must be between 0 and 127 decimal.

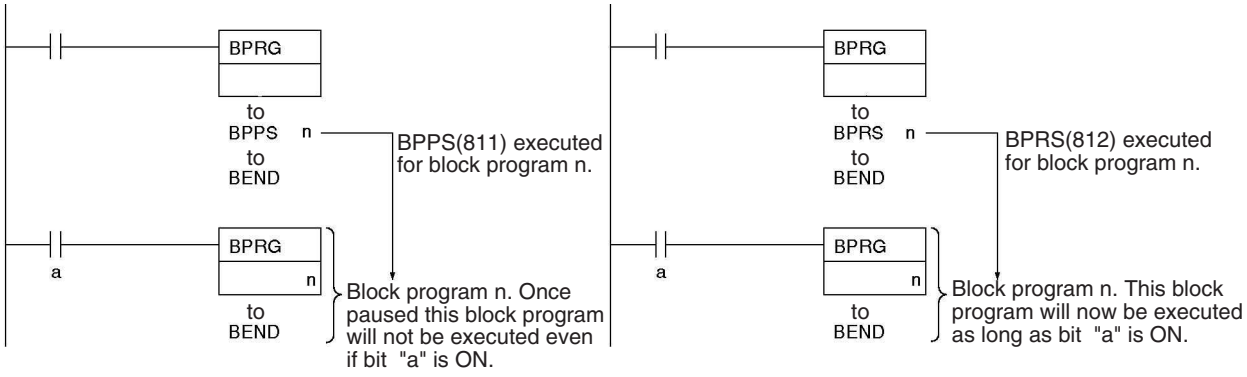
Operand Specifications

Area	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	0 to 127 (decimal)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Description

BPPS(811) is used inside one block program to pause the execution of another block program specified by N, the block program number. The block program that is paused with BPPS(811) even if the BPRG(096) for the block program has an ON execution condition. The block program will not be restarted until BPRS(812) is executed for it.

BPRS(812) restarts the block program specified by N, the block program number. Once restarted, the block program will be executed as long as the BPRG(096) for the block program has an ON execution condition.



Flags

Name	Label	Operation
Error Flag	ER	ON if BPPS(811) or BPRS(812) is not in a block program. ON if N is not between 0 and 127. OFF in all other cases.

Precautions

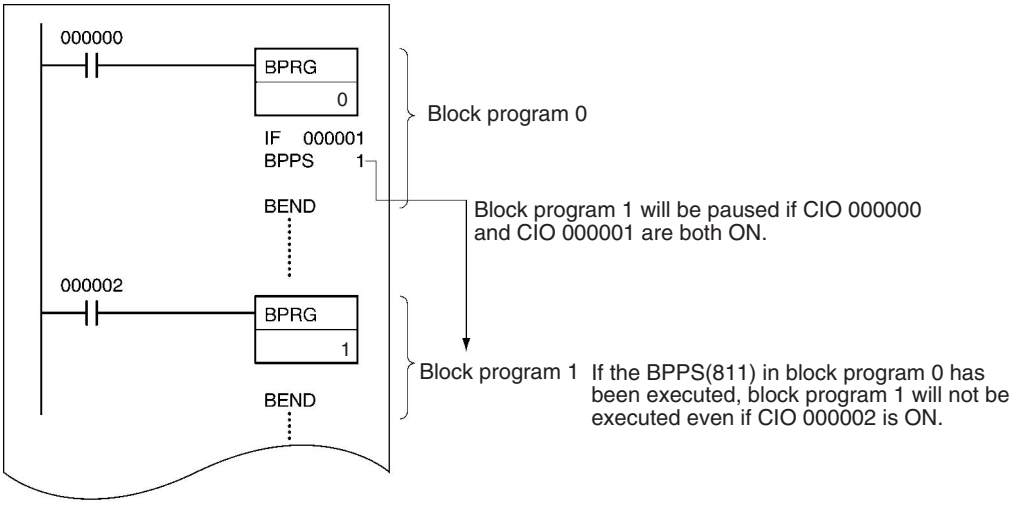
An error will occur and the Error Flag will turn ON if BPPS(811) or BPRS(812) is not in a block program or if N is not between #0000 and #007F (binary).

BPPS(811) can be used to pause the block program that contains it. When the block program is then restarted using BPRS(812) from another block program, the paused block program will restart from the next instruction after BPPS(811).

If a paused block program contains TIMW(813), TIMWX(816), TMHW(815), or TMHWX(817), the PV of the time will continue to elapse even while the block program is paused.

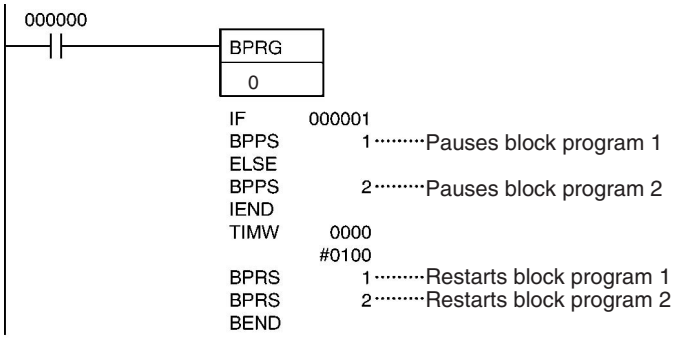
Examples

The following diagram shows a basic example of pausing a block program.



Note If the block program that is being paused appears after BPPS(811), it will not be executed. If the block program appears before BPPS(811), it will be paused starting the next cycle.

If CIO 000000 is ON, the following program pauses execution of either block program 1 or block program 2 depending on the status of CIO 000001. The block program that was paused is then restarted after 10 seconds.



Address	Instruction	Operands
000000	LD	000000
000001	BPRG(096)	00
000002	IF(802)	000001
000003	BPPS(811)	01
000004	ELSE(803)	
000005	BPPS(811)	02
000006	IEND(804)	
000007	TIMW(803)	0000
		# 0100
000008	BPRS(812)	1
000009	BPRS(812)	2
000010	BEND(801)	

3-32-4 Branching: IF(802), ELSE(803), and IEND(804)

Purpose

Branches the block program either based on an execution condition or on the status of an operand bit.

Ladder Symbol

IF(802) B

IF(802)

IF(802) NOT B

ELSE(803)

IEND(804)

B: Bit operand

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Note IF(802), ELSE(803), and IEND(804) must be used in block programming regions even within subroutines and interrupt tasks.

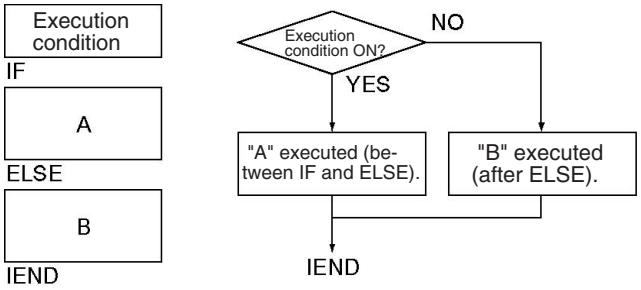
Operand Specifications

Area	B
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A44715 A44800 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flags	TK0000 to TK0031
Condition Flags	ER, CY, >, =, <, N, OF, UF, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

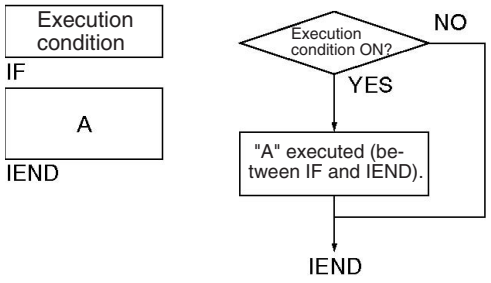
Description

Operation without an Operand for IF(802)

If an operand bit is not specified, an execution must be created before IF(802) starting with LD. If the execution condition is ON, the instructions between IF(802) and ELSE(803) will be executed and if the execution condition is OFF, the instructions between ELSE(803) and IEND(804) will be executed.

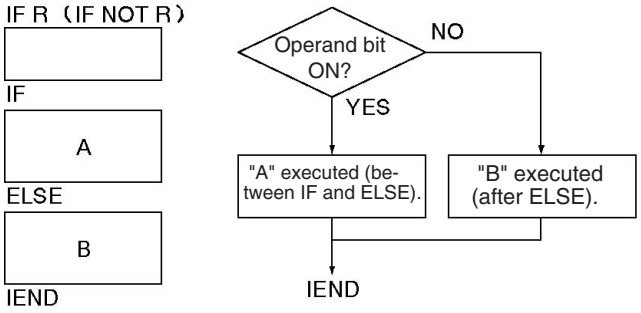


If the ELSE(803) instruction is omitted and the execution condition is ON, the instructions between IF(802) and IEND(804) will be executed and if the execution condition is OFF, only the instructions after IEND(804) will be executed.

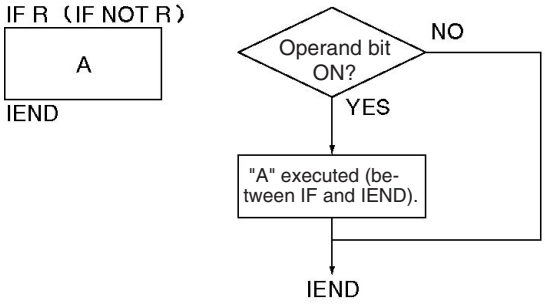


Operation with an Operand for IF(802) or IF NOT(802)

An operand bit, B, can be specified for IF(802) or IF NOT(802). If the operand bit is ON, the instructions between IF(802) and ELSE(803) will be executed. If the operand bit is OFF, the instructions between ELSE(803) and IEND(804) will be executed. For IF NOT(802), the instructions between IF(802) and ELSE(803) will be executed and if the operand bit is ON, the instructions be ELSE(803) and IEND(804) will be executed is the operand bit is OFF.



If the ELSE(803) instruction is omitted and the operand bit is ON, the instructions between IF(802) and IEND(804) will be executed and if the operand bit is OFF, only the instructions after IEND(804) will be executed. The same will happen for the opposite status of the operand bit if IF NOT(802) is used.



Flags

Name	Label	Operation
Error Flag	ER	ON if the branch instructions are not in a block program. ON if more than 254 branches are nested. OFF in all other cases.

Precautions

Instructions in block programs are generally executed unconditionally. Branching, however, can be used to create conditional execution based on execution conditions or operand bits.

Use IF A ELSE B IEND to branch between A and B.

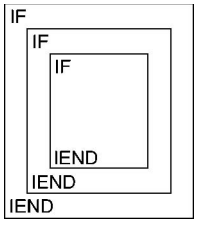
Use IF A IEND to branch between A and doing nothing.

Branches can be nested to up to 253 levels.

A error will occur and the Error Flag will turn ON if the branch instructions are not in a block program or if more than 254 branches are nested.

Nesting Branches

Up to 253 branches can be nested within the top level branch.

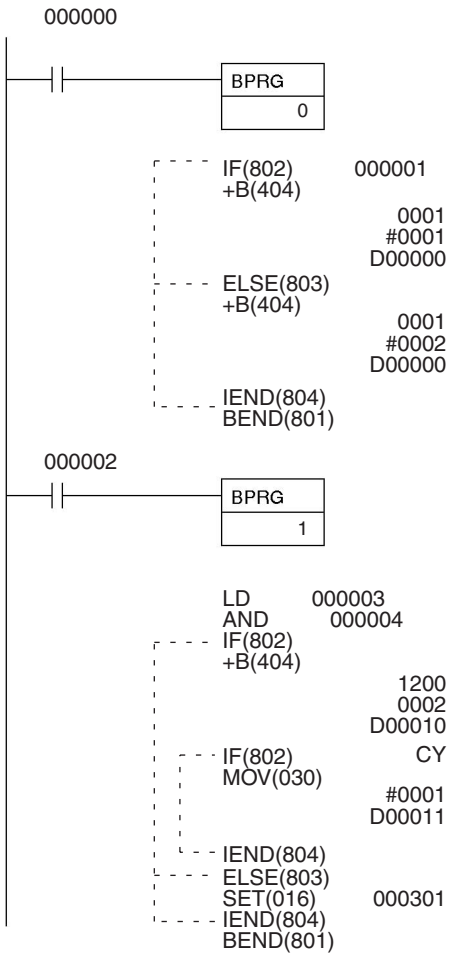


Examples

The following example shows two different block programs controlled by CIO 000000 and CIO 000002.

The first block executes one of two additions depending on the status of CIO 000001. This block is executed when CIO 000000 is ON. If CIO 000001 is ON, 0001 is added to the contents of CIO 0001. If CIO 000001 is OFF, 0002 is added to the contents of CIO 0001. In either case, the result is placed in D00000.

The second block is executed when CIO 000002 is ON and shows nesting two levels. If CIO 000003 and CIO 000004 are both ON, the contents of CIO 1200 and CIO 0002 are added and the result is placed in D00010 and then 0001 is moved into D00011 based on the status of CY. If either CIO 000003 or CIO 000004 is OFF, then the entire addition operation is skipped and CIO 000301 is turned ON.



Address	Instruction	Operands
000000	LD	000000
000001	BPRG(096)	00
000002	IF(802)	000001
000003	+B(404)	
		0001
		#0001
		D00000
000004	ELSE(803)	
000005	+B(404)	
		0001
		#0002
		D00000
000006	IEND(804)	
000007	BEND(801)	
000008	LD	000002
000009	BPRG(096)	1
000010	LD	000003
000011	AND	000004
000012	IF(802)	
000013	+B(404)	
		1200
		0002
		D00010
000014	IF(802)	A50004
000015	MOV(030)	
		#0001
		D00011
000016	IEND(804)	
000017	ELSE(803)	
000018	SET(016)	000301
000019	IEND(804)	
000020	BEND(801)	

3-32-5 CONDITIONAL BLOCK EXIT (NOT): EXIT (NOT)(806)

Purpose

Exists the block program (i.e., does not execute any other instruction in the block program through BEND(801) depending on the status of the operand bit or on the execution condition. EXIT(806) without an operand bit exits the program if the execution condition is ON. EXIT(806) with an operand bit exits the program if the bit is ON. EXIT NOT(806) must have an operand bit and exits the program if the bit is OFF.

Ladder Symbol

EXIT(806)

EXIT(806) B

EXIT NOT(806) B

B: Bit operand

Variations

Variations	Always Executed in Block Program	EXIT(806) EXIT(806) B EXIT NOT(806) B

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Note EXIT(806) and EXIT NOT(806) must be used in block programming regions even within subroutines and interrupt tasks.

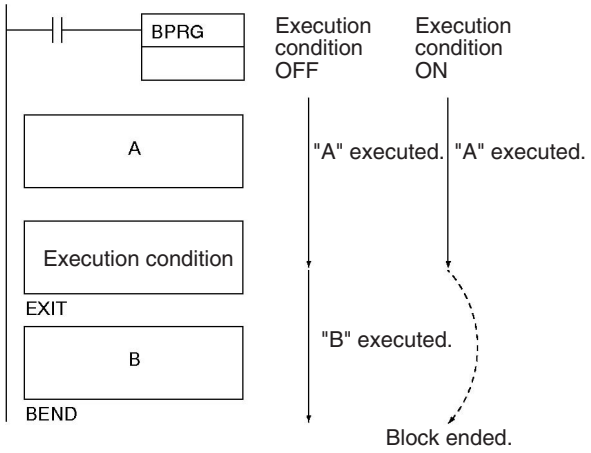
Operand Specifications

Area	B
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A44715 A44800 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flags	TK0000 to TK0031
Condition Flags	ER, CY, >, =, <, N, OF, UF, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

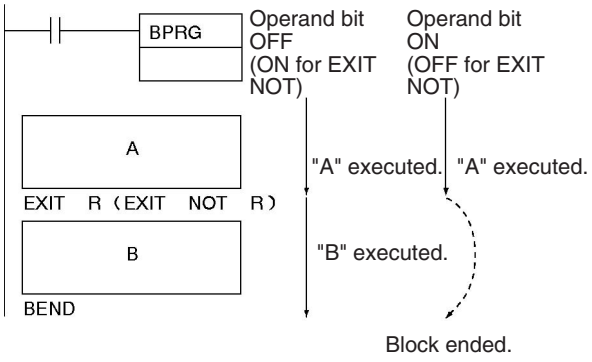
Operation without an Operand

EXIT(806) can be executed without an operand. If it is, then an execution condition must be created for it starting with LD. If the execution condition is OFF, the rest of the block program will be executed normally. If the execution condition is ON, the rest of the instructions in the block program through BEND(801) will not be executed.



Operation with an Operand

If the operand bit, B, is OFF for EXIT(806) the rest of the block program will be executed normally. If the operand bit is ON for EXIT(806), the rest of the instructions in the block program through BEND(801) will not be executed. For EXIT NOT(806), the rest of the block program will be executed for if the operand bit is ON and skipped if the operand bit is OFF.



Flags

Name	Label	Operation
Error Flag	ER	ON if EXIT(806) or EXIT NOT(806) is not in a block program. OFF in all other cases.

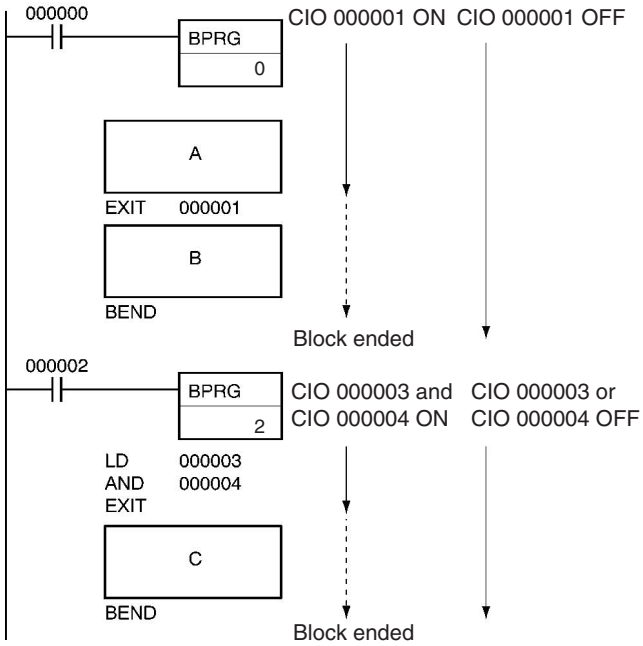
Precautions

An error will occur and the Error Flag will turn ON if EXIT(806) or EXIT NOT(806) is not in a block program.

Examples

When CIO 000000 is OFF, the block program is executed. If CIO 000001 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 000001 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.



3-32-6 ONE CYCLE AND WAIT (NOT): WAIT(805)/WAIT(805) NOT

Purpose

Stops execution of the rest of the block program until an execution condition turns ON or an operand bit turns ON or OFF.

Ladder Symbol

WAIT(805)

WAIT(805) B

WAIT(805) NOT B

B: Bit operand

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Note WAIT(805)/WAIT(805) NOT must be used in block programming regions even within subroutines and interrupt tasks.

Operand Specifications

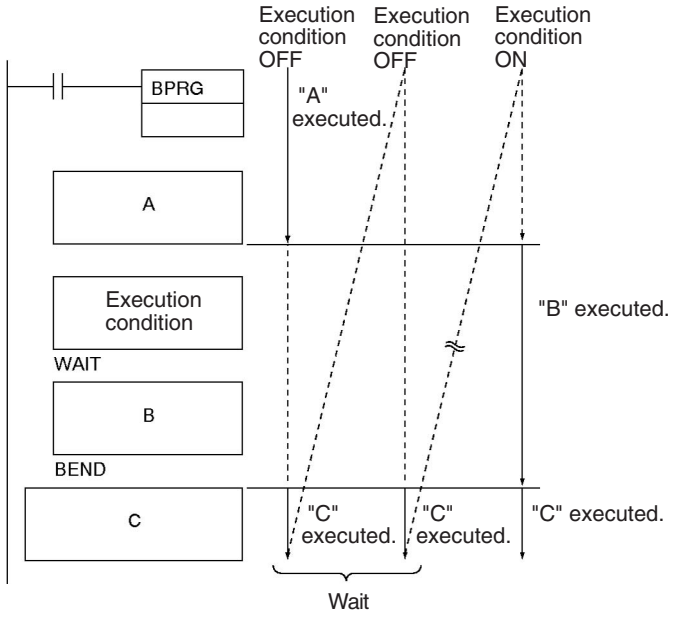
Area	B
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A44715 A44800 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flags	TK0000 to TK0031
Condition Flags	ER, CY, >, =, <, N, OF, UF, >=, <>, <=ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min

Area	B
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

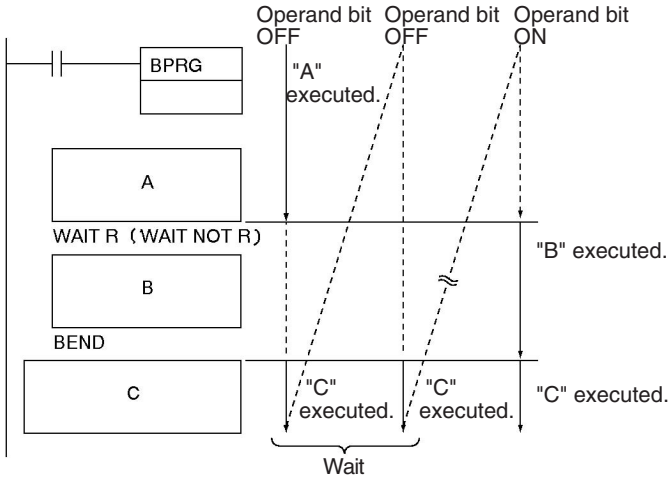
Operation without an Operand

If an operand bit is not specified, an execution must be created before WAIT(805)/WAIT(805 NOT starting with LD. If the execution condition is ON for WAIT(805), the rest of the instruction in the block program will be skipped. In the next cycle, none of the block program will be executed except for the execution condition for WAIT(805). When the execution condition goes ON, the instruction from WAIT(805) to the end of the program will be executed.



Operation with an Operand

An operand bit, B, can be specified for WAIT(805) or WAIT NOT(805). If the operand bit is OFF (ON for WAIT NOT(805)), the rest of the instructions in the block program will be skipped. In the next cycle, none of the block program will be executed except for the execution condition for WAIT(805) or WAIT(805) NOT. When the execution condition goes ON (OFF for WAIT(805) NOT), the instruction from WAIT(805) or WAIT(805) NOT to the end of the program will be executed.



Flags

Name	Label	Operation
Error Flag	ER	ON if WAIT(805) or WAIT(805) NOT is not in a block program. OFF in all other cases.

Precautions

WAIT(805) and WAIT(805) NOT can be used for step progressions inside block programs.

An error will occur and the Error Flag will turn ON if WAIT(805) or WAIT(805) NOT is not in a block program.

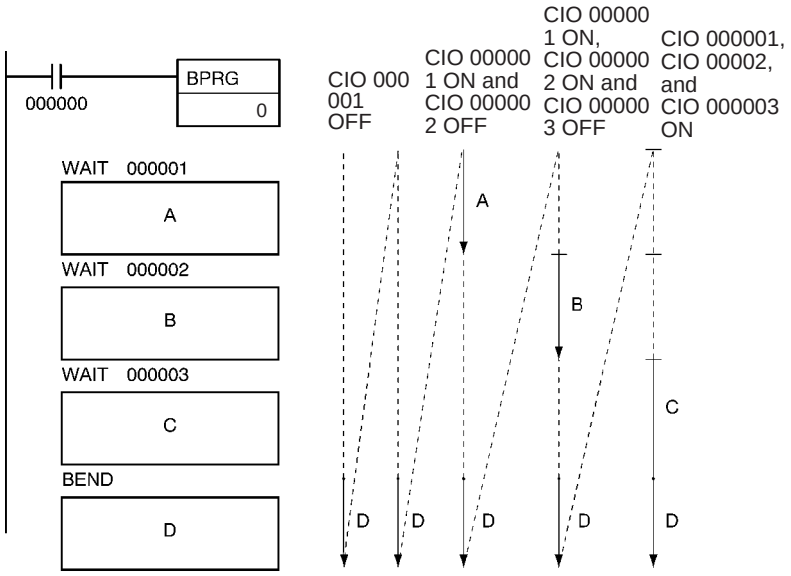
Note The program addresses of WAIT instructions with operands specified and the program addresses of the first instruction creating the execution conditions for WAIT instructions without operands are recorded in memory to enable execution to be continued based on the execution condition/bit operand. If online editing performed from a Peripheral Device, however, the WAIT status will be cleared and the block program will again be executed from the beginning.

Examples

When CIO 000000 is ON in the following example, block program 00 will be executed. Execution would proceed as follows:

1,2,3...

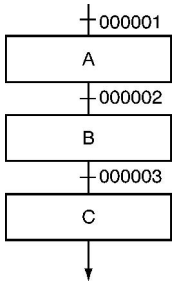
1. If CIO 000001 is OFF, none of the block program will be executed until CIO 000001 turns ON. When CIO 000001 turns ON, “A” will be executed.
2. If CIO 000002 is OFF after “A” is executed, the rest of the block program will not be executed until CIO 000002 turns ON. When CIO 000002 turns ON, “B” will be executed
3. If CIO 000003 is OFF after “B” is executed, the rest of the block program will not be executed until CIO 000003 turns ON. When CIO 000003 turns ON, “C” will be executed and the execution process will be repeated.



The following table shown the relationship between the operand bits and block program execution.

Operand bits			Program execution		
CIO 000001	**CIO 000002**	**CIO 000003**	**First cycle CIO 000000 is ON**	**Next cycle**	**Following cycles**
OFF	Any status	Any status	Nothing executed.	Nothing executed; waiting for CIO 000001.	When CIO 000001 turns ON “A” is executed and the status of CIO 000002 is checked.
ON	OFF	Any status	“A” executed.	Waiting for CIO 000002.	When CIO 000002 turns ON “B” is executed and the status of CIO 000003 is checked.
ON	ON	OFF	“A” and “B” executed.	Waiting for CIO 000003.	When CIO 000003 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.



Note 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.

3-32-7 HUNDRED-MS TIMER WAIT: TIMW(813) and TIMWX(816)

Purpose

Delays execution of the rest of the block program until the specified time has elapsed. Execution will be continued from the next instruction after TIMW(813)/TIMWX(816) when the timer times out.

Ladder Symbol

PV Refresh Method: BCD

TIMW(813)
N
SV

N: Timer number
SV: Set value

PV Refresh Method: Binary

TIMWX(816)
N
SV

N: Timer number
SV: Set value

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	Not allowed.

Note TIMW(813)/TIMWX(816) must be used in block programming regions even within subroutines.

Operands

N: Timer Number
BCD: 0 to 4095 (decimal)
Binary: 0 to 4095 (decimal)

S: Set Value
BCD: #0000 to #9999 (BCD)
Binary: &0 to &65535 (decimal)
#0000 to #FFFF (hex)

Operand Specifications

Area	N	SV
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A447 A448 to A959
Timer Area	0000 to 4095	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)

Area	N	SV
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

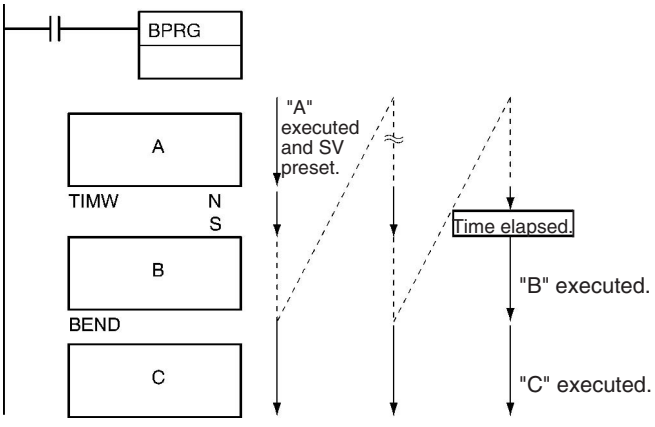
TIMW(813)/TIMWX(816) creates an ON-delay countdown timer (100-ms timer set in SV) between execution of the block program instruction preceding it and the instructions following. TIMW(813) can time from 0 to 999.9 s with a timer accuracy of 0 to 0.01 s. TIMWX(816) can time from 0 to 6,553.5 s with a timer accuracy of 0 to 0.01 s.

Note The timer accuracy for CS1D CPU Units is 10 ms + 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. Once the entire block program has been executed, the process will be repeated.

TIMW(813)/TIMWX(816) can be thought of as a WAIT instruction with a timer for the execution condition and it can thus be used for timed step progressions.



Flags

Name	Label	Operation
Error Flag	ER	ON if TIMW(813)/TIMWX(816) is not in a block program. ON if an indirect IR designation is used for N in BCD mode and the address is not for a timer present value. ON if in BCD mode and SV is not BCD. OFF in all other cases.

Precautions

The rest of the block program following timer will be executed if the Completion Flag for the timer is force set.

If the Completion Flag for the timer is force reset, only TIMW(813)/ TIMWX(816)) will be executed in the block program until the force reset status is cleared.

The present value of timers programmed with timer numbers 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 timer numbers are also used by the other timer instructions. Operation will not be predictable if the same timer number is used for more than one timer instruction. Use each timer number only once. The only way that the same timer number can be used dependably is if only one of the timers is ever operating at the same time. An error will occur in the program check if the same timer number is used in more than one timer instruction.

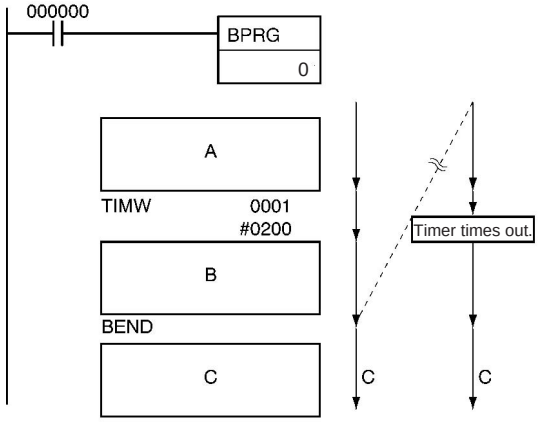
An error will occur and the Error Flag will turn ON if an indirect IR designation is used for N in BCD mode and the address is not for a timer present value or if SV is not BCD.

The timer will not operate correctly if the cycle time is 100 ms or longer.

Note 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.

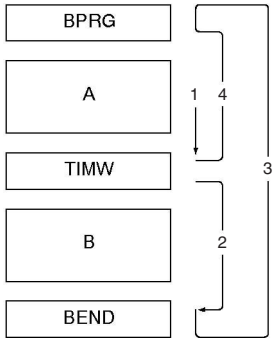
Examples

In the following example, “B” will be executed 20 seconds after “A” whenever CIO 000000 is ON.



Address	Instruction	Operand
000200	LD	000000
000201	BPRG	0
. .	A	. .
000210	TIMW	0001
		#0200
. .	B	. .
000220	BEND	---

Program execution will flow from 2 to 3 to 4 and back to 2 during the 20 s before “B” is executed, as shown in the following diagram.



3-32-8 COUNTER WAIT: CNTW(814) and CNTWX(818)

Purpose

Delays execution of the rest of the block program until the specified count has been achieved. Execution will be continued from the next instruction after CNTW(814)/CNTWX(818) when the counter counts out.

Ladder Symbol

PV Refresh Method: BCD

CNTW(814)	N	**N**: Counter number
	SV	**SV**: Set value
	I	**I**: Count input

PV Refresh Method: Binary

CNTWX(818)	N	**N**: Counter number
	SV	**SV**: Set value
	I	**I**: Count input

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Note CNTW(814)/CNTWX(818) must be used in block programming regions even within subroutines and interrupt tasks.

Operands

N: Counter Number
BCD: 0 to 4095 (decimal)
Binary: 0 to 4095 (decimal)

S: Set Value
BCD: #0000 to #9999 (BCD)
Binary: &0 to &65535 (decimal)
#0000 to #FFFF (hex)

Operand Specifications

Area	N	SV	I
CIO Area	---	CIO 0000 to CIO 6143	CIO 000000 to CIO 614315
Work Area	---	W000 to W511	W00000 to W51115
Holding Bit Area	---	H000 to H511	H00000 to H51115
Auxiliary Bit Area	---	A000 to A447 A448 to A959	A00000 to A44715 A44800 to A95915
Timer Area	---	T0000 to T4095	T0000 to T4095
Counter Area	C0000 to C4095	C0000 to C4095	C0000 to C4095
Task Flags	---		TK0000 to TK0031
Condition Flags	---		ER, CY, >, =, <, N, OF, UF, >=, <>, <=, ON,OFF, AER
Clock Pulses	---		0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
DM Area	---	D00000 to D32767	---
EM Area without bank	---	E00000 to E32767	---
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)	---
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	---
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	---
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)	---
Data Registers	---	DR0 to DR15	---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

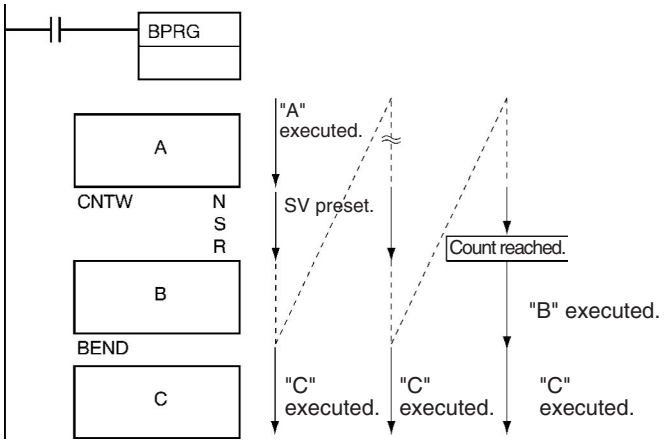
Description

CNTW(814)/CNTWX(818) creates a decrementing counter that delays execution of the instructions following it in the block program until the counter has counted out. The set value for CNTW(814) is specified in BCD between 0000 and 9999. The set value for CNTWX(818) is specified in binary between 0000 and FFFF hex.

The first part of the block program is executed the first time the block program is entered. When CNTW(814)/CNTWX(818) is reached, the Completion Flag is reset to 0, the counter is preset to SV, and execution of the rest of the block program will wait until the counter has counted out. The counter counts pulses (upward differentiation) on I, the counter input.

While the counter is counting down, only CNTW(814)/CNTWX(818) will be executed to update the counter. When the counter counts out, the Completion Flag will turn ON and the rest of the block program will be executed. Once the entire block program has been executed, the process will be repeated.

CNTW(814)/CNTWX(818) can be thought of as a WAIT instruction with a counter for the execution condition and it can thus be used for timed step progressions.



Flags

Name	Label	Operation
Error Flag	ER	ON if CNTW(814)/CNTWX(818) is not in a block program. ON if an indirect IR designation is used for N in BCD mode and the address is not for a counter present value. ON if SV is not BCD when BCD mode is set. OFF in all other cases.

Precautions

The rest of the block program following CNTW(814)/CNTWX(818) will be executed if the Completion Flag for the counter is force set.

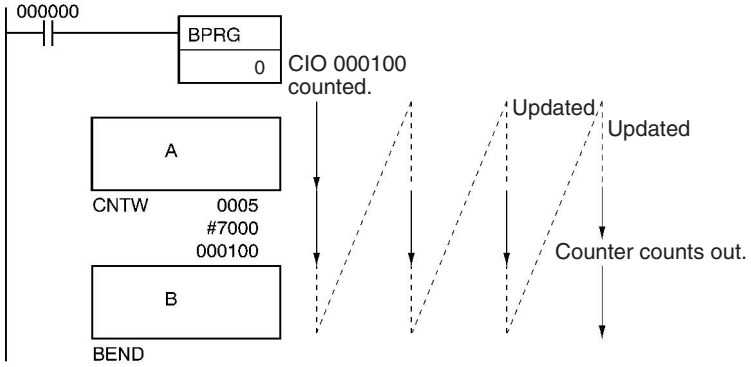
If the Completion Flag for the counter is force reset, the only CNTW(814)/CNTWX(818) will be executed in the block program until the force reset status is cleared.

The counter numbers are also used by the other counter instructions. Operation will not be predictable if the same counter number is used for more than one counter instruction. Use each counter number only once. The only way that the same counter number can be used dependably is if only one of the counters is ever operating at the same time. An error will occur in the program check if the same counter number is used in more than one counter instruction.

An error will occur and the Error Flag will turn ON if an indirect IR designation is used for N in BCD mode and the address is not for a counter present value or if SV is not BCD when BCD mode is set.

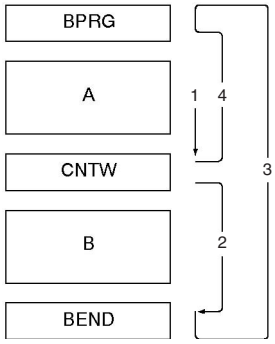
Examples

When CIO 000000 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 000100.



Address	Instruction	Operand
000200	LD	000000
000201	BPRG	0
. .	A	. .
000210	CNTW	0005
		#7000
		000100
. .	B	. .
000220	BEND	---

Program execution will flow from 2 to 3 to 4 and back to 2 during the 7,000 counts before “B” is executed, as shown in the following diagram.



3-32-9 TEN-MS TIMER WAIT: TMHW(815) and TMHWX(817)

Purpose

Delays execution of the rest of the block program until the specified time has elapsed. Execution will be continued from the next instruction after TMHW(815)/TMHWX(817) when the timer times out.

Ladder Symbol

PV Refresh Method: BCD

TMHW(815)	N	**N**: Timer number
	SV	**SV**: Set value

PV Refresh Method: Binary

TMHWX(817)	N	**N**: Timer number
	SV	**SV**: Set value

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	Not allowed.

Note TMHW(815)/TMHWX(817) must be used in block programming regions even within subroutines.

Operands

N: Timer Number
BCD: 0 to 4095 (decimal)
Binary: 0 to 4095 (decimal)

S: Set Value
BCD: #0000 to #9999 (BCD)
Binary: &0 to &65535 (decimal)
#0000 to #FFFF (hex)

Operand Specifications

Area	N	SV
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W511
Holding Bit Area	---	H000 to H511
Auxiliary Bit Area	---	A000 to A447 A448 to A959
Timer Area	0000 to 4095	T0000 to T4095
Counter Area	---	C0000 to C4095
DM Area	---	D00000 to D32767
EM Area without bank	---	E00000 to E32767
EM Area with bank	---	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	---	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	---	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---	BCD: #0000 to 9999 (BCD) “&” cannot be used. Binary: &0 to &65535 (decimal) #0000 to #FFFF (hex)
Data Registers	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

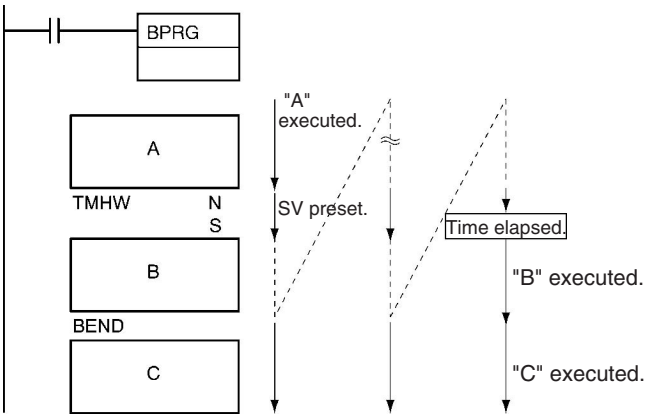
TMHW(815)/TMHWX(817) creates an ON-delay countdown timer (10-ms timer set in SV) between execution of the block program instruction preceding it and the instructions following. TMHW(815) can time from 0 to 99.99 s with a timer accuracy of 0 to 0.01 s. TMHWX(817) can time from 0 to 655.35 s with a timer accuracy of 0 to 0.01 s.

Note The timer accuracy for CS1D CPU Units is 10 ms + 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. Once the entire block program has been executed, the process will be repeated.

TMHW(815)/TMHWX(817) can be thought of as a WAIT instruction with a timer for the execution condition and it can thus be used for timed step progressions.



Flags

Name	Label	Operation
Error Flag	ER	ON if TMHW(815)/TMHWX(817) is not in a block program. ON if an indirect IR designation is used for N in BCD mode and the address is not for a timer present value. ON if in BCD mode and SV is not BCD. OFF in all other cases.

Precautions

The rest of the block program following TMHW(815)/TMHWX(817) will be executed if the Completion Flag for the timer is force set.

If the Completion Flag for the timer is force reset, the only TMHW(815)/TMHWX(817) will be executed in the block program until the force reset status is cleared.

The present value of timers programmed with timer numbers 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 timer numbers are also used by the other timer instructions. Operation will not be predictable if the same timer number is used for more than one timer instruction. Use each timer number only once. The only way that the same timer number can be used dependably is if only one of the timers is ever

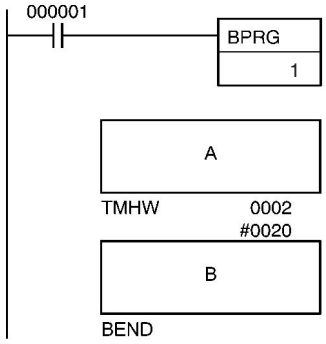
operating at the same time. An error will occur in the program check if the same timer number is used in more than one timer instruction.

An error will occur and the Error Flag will turn ON if an indirect IR designation is used for N in BCD mode and the address is not for a timer present value or if SV is not BCD.

The timer will not operate correctly if the cycle time is 100 ms or longer.

Examples

In the following example, “B” will be executed 20 seconds after “A” whenever CIO 000000 is ON.



Address	Instruction	Operand
000221	LD	000001
000222	BPRG	1
. .	A	. .
000250	TMHW	0002
		#0020
. .	B	. .
000281	BEND	---

3-32-10 Loop Control: LOOP(809)/LEND(810)/LEND(810) NOT

Purpose

Create a loop that is repeatedly executed until an execution condition turns ON or OFF or until an execution condition turns ON.

Ladder Symbol

LOOP(809)

LEND(810)

LEND(810) B **B**: Bit operand

LEND(810) NOT B

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Note LOOP(809), LEND(810), and LEND(810) NOT must be used in block programming regions even within subroutines and interrupt tasks.

Operand Specifications

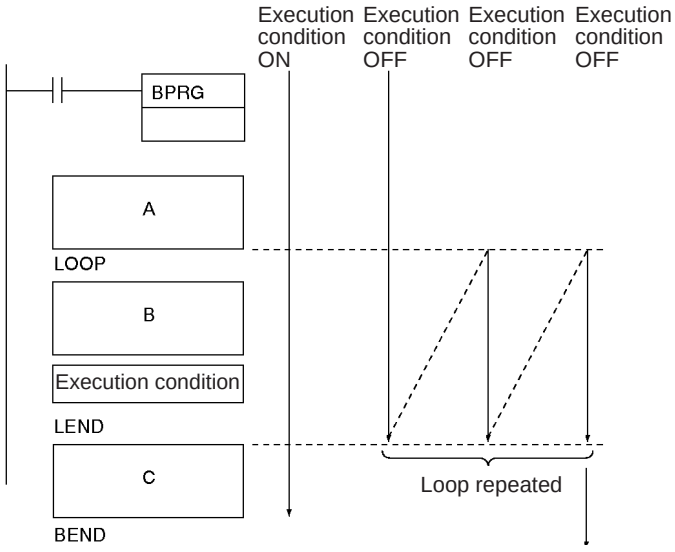
Area	B
CIO Area	CIO 000000 to CIO 614315
Work Area	W00000 to W51115
Holding Bit Area	H00000 to H51115
Auxiliary Bit Area	A00000 to A44715 A44800 to A95915
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
Task Flags	TK0000 to TK0031
Condition Flags	ER, CY, >, =, <, N, OF, UF, >=, <>, <=, ON,OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

LOOP(809) designates the beginning of the loop program. LEND(810) or LEND(810) NOT specifies the end of the loop. When LEND(810) or LEND(810) NOT is reached, program execution will loop back to the next previous LOOP(809) until the operand bit for LEND(810) or LEND(810) NOT turns ON or OFF (respectively) or until the execution condition for LEND(810) turns ON.

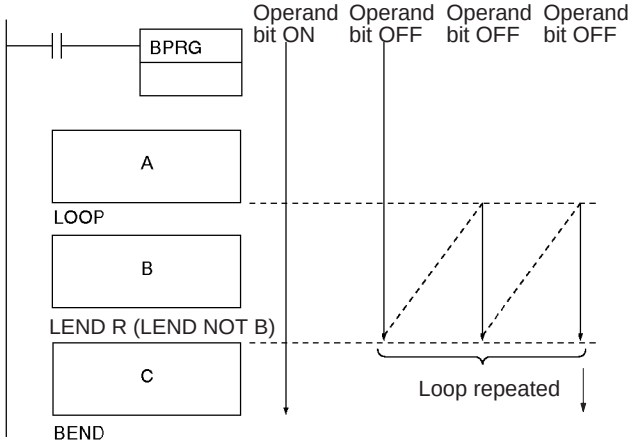
Using an Execution Condition for LEND(810)

LEND(810) can be programmed either with or without an operand bit. If an operand bit is not specified, an execution must be created before LEND(810) starting with LD. If the execution condition is OFF, execution of the loop is repeated starting with the next instruction after LOOP(809). If the execution condition is ON, the loop is ended and execution continues to the next instruction after LEND(810).



Using a Bit Operand for LEND(810) or LEND(810) NOT

Both LEND(810) and LEND(810) NOT can be programmed with an operand bit. If the operand bit is OFF for LEND(810) (or ON for LEND(810) NOT), execution of the loop is repeated starting with the next instruction after LOOP(809). If the operand bit is ON for LEND(810) (or OFF for LEND(810) NOT), the loop is ended and execution continues to the next instruction after LEND(810) or LEND(810) NOT.



Note The status of the operand bit would be reversed for LEND(810) NOT.

Note

1. Execution inside a loop does not refresh I/O data. If I/O data must be refreshed during the loop, use IORF(184).
2. The maximum cycle time can be exceeded if loops are repeated too long. Design the program so that the maximum cycle time is not exceeded.

Flags

Name	Label	Operation
Error Flag	ER	ON if a Loop Control Instruction is not in a block program. OFF in all other cases.

Precautions

Loops cannot be nested within loops.

Incorrect:

```
LOOP(809)
LOOP(809)
LEND(810)
LEND(810)
```

Do not reverse the order of LOOP and LEND.

Incorrect:

```
LEND(810)
   :
   :
LOOP(809)
```

Conditional block branching can be used within a loop, but the entire branch operation must be within the loop.

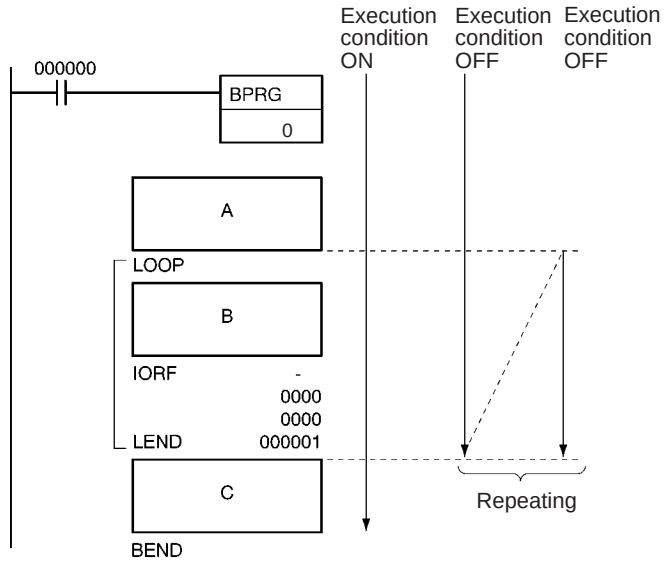
Correct:	**Incorrect:**
LOOP(809)	LOOP(809)
IF(802)	IF(802)
IF(802)	IF(802)
IEND(804)	IEND(804)
IEND(804)	LEND(810)
LEND(810)	IEND(804)

NOP processing will be performed if LOOP(809) is not executed.

An error will occur and the Error Flag will turn ON if a Loop Control Instruction is not in a block program.

Examples

When CIO 000000 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 000001 is ON, at which time C will be executed and the block program will end.



Address	Instruction	Operand
000220	LD	000000
000201	BPRG	0
. .	A	. .
000210	LOOP	---
. .	B	. .
000220	IORF	. .
		0000
		0000
000221	LEND	000001
. .	C	. .
000220	BEND	---

3-33 Text String Processing Instructions

This section describes instructions used to manipulate text strings.

Instruction	Mnemonic	Function code	Page
MOV STRING	MOV$	664	1221
CONCATENATE STRING	+$	656	1223
GET STRING LEFT	LEFT$	652	1226
GET STRING RIGHT	RGHT$	653	1228
GET STRING MIDDLE	MID$	654	1230
FIND IN STRING	FIND$	660	1233
STRING LENGTH	LEN$	650	1235
REPLACE IN STRING	RPLC$	661	1237
DELETE STRING	DEL$	658	1240
EXCHANGE STRING	XCHG$	665	1242
CLEAR STRING	CLR$	666	1245
INSERT INTO STRING	INS$	657	1246
String Comparison Instructions	=$, <>$, <$, <=$, >$, >=$	670 to 675	1250
WRITE TEXT FILE	TWRIT	704	1255

3-33-1 Text String Processing Overview

Data from the beginning until a NUL code (00 hex) is handled as text string data expressed in ASCII (except for 1-byte, special characters). It is stored from leftmost to rightmost bytes, and from rightmost to leftmost words.

When there is an odd number of characters, 00 hex (NUL code) is stored in the available space in the rightmost byte of the final word.

Example: Text string ABCDE

A	B	=	41	42
C	D		43	44
E	NUL		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

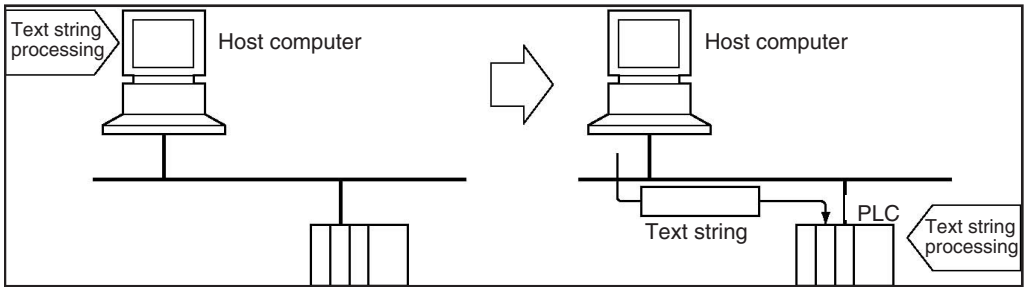
A	B	=	41	42
C	D		43	44
NUL	NUL		00	00

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$ D00000 D00100

D00000	41	42	→	D000100	41	42
D00001	43	44		D000101	43	44
D00002	45	NUL		D000102	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.

ASCII Characters

The ASCII characters that can be handled by text string processing instructions are shown in the following table.

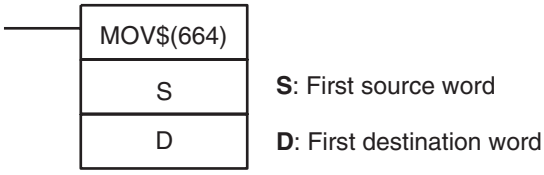
Four rightmost bits	Four leftmost bits 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p				ー	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			”	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			’	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A			*	:	J	Z	j	z			ェ	コ	ハ	レ		
B			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C			,	<	L	¥	l	|			ャ	シ	フ	ワ		
D			−	=	M	]	m	}			ュ	ス	ヘ	ン		
E			.	>	N	^	n	~			ョ	セ	ホ	゛		
F			/	?	O	_	o				ッ	ソ	マ	°		

3-33-2 MOV STRING: MOV$(664)

Purpose

Transfers a text string.

Ladder Symbol



Variations

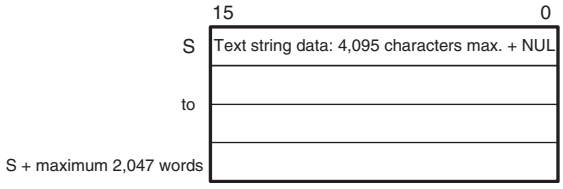
Variations		
Variations	**Executed Each Cycle for ON Condition**	MOV$(664)
	Executed Once for Upward Differentiation	@MOV$(664)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

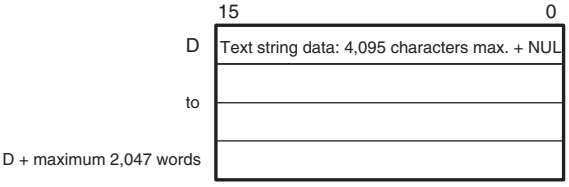
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: First Source Word



D: First Destination Word



Note

1. The data from S to S +the maximum 2,047 words and from D to D + the maximum 2,047 words must be in the same area.
2. The data from S to S + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

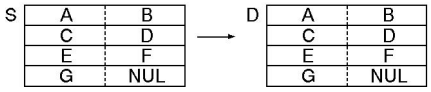
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A447 A448 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	

Area	S	D
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

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).



Note MOV$(664) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is transferred to D. OFF in all other cases.

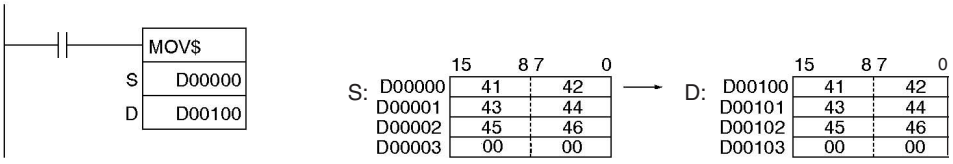
Precautions

If more than 4,095 characters are designated by S, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is transferred to D, the Equals Flag will turn ON.

Example

In this example, MOV$(664) is used to transfer the text string ABCDEF.

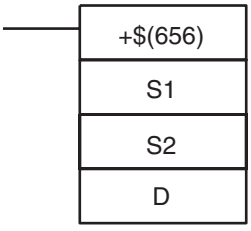


3-33-3 CONCATENATE STRING: +$(656)

Purpose

Links one text string to another text string.

Ladder Symbol



S1: Text string 1

S2: Text string 2

D: First destination word

Variations

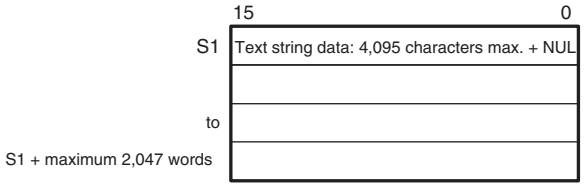
Variations		
Variations	**Executed Each Cycle for ON Condition**	+$(656)
	Executed Once for Upward Differentiation	@+$(656)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

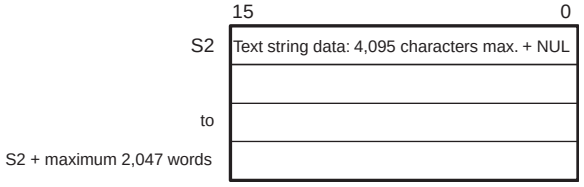
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

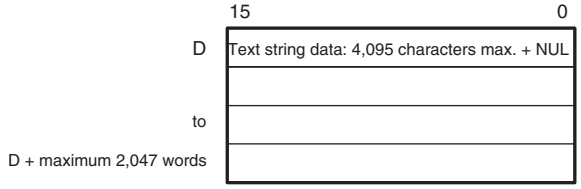
S1: Text String 1



S2: Text String 2



D: First Destination Word



Note

1. The data from S1 to S1 + the maximum 2,047 words, from S2 to S2 + the maximum 2,047 words, and from D to D + the maximum 2,047 words must be in the same area.
2. The data from S2 to S2 + the maximum 2,047 words and from D to D + the maximum 2,047 words cannot overlap.

Operand Specifications

Area	S1	S2	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A447 A448 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to 32767 (n = 0 to C)		

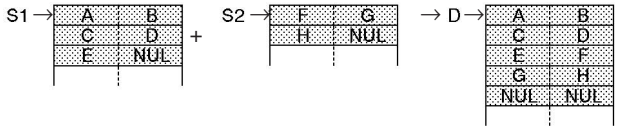
Area	S1	S2	D
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0V to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

+$(664) connects the text string data designated by S1 to the text string data designated by S2, and outputs the result to D as text string data (including the final NUL).

The maximum number of characters that can be designated by S1 and S2 is 4,095 (0FFF hex). If there is no NUL until 4,096 characters, an error will be generated and the Error Flag will turn ON. Moreover, the result of the linkage can be no more than 4,095 characters (0FFF hex). If the linkage results in more characters than that, only the first 4,095 characters (with NUL added as the 4,096th) will be output to D.

If there is a NUL for both S1 and S2, the two NUL characters (0000 hex) will be output to D.



Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 and S2. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is transferred to D. OFF in all other cases.

Precautions

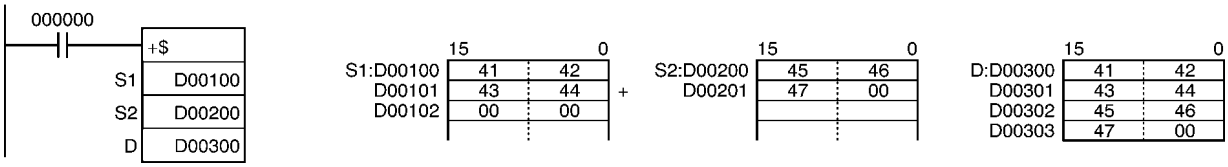
If more than 4,095 characters are designated by S1 and S2, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is transferred to D, the Equals Flag will turn ON.

Do not overlap the beginning word designated by D with the character data area for S2. If they overlap, the instruction cannot be executed properly.

Example

In this example, +$(656) is used to connect the text strings ABCD and EFG and output the result to D.

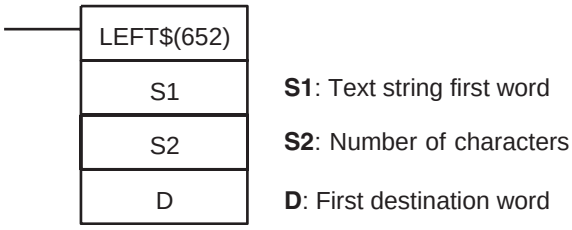


3-33-4 GET STRING LEFT: LEFT$(652)

Purpose

Fetches a designated number of characters from the left (beginning) of a text string.

Ladder Symbol



S1: Text string first word

S2: Number of characters

D: First destination word

Variations

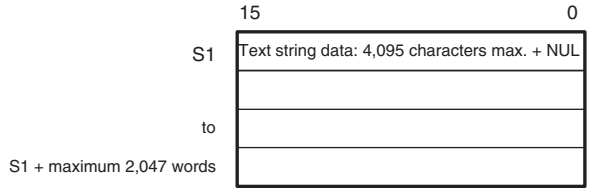
Variations		
	Executed Each Cycle for ON Condition	LEFT$(652)
	Executed Once for Upward Differentiation	@LEFT$(652)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

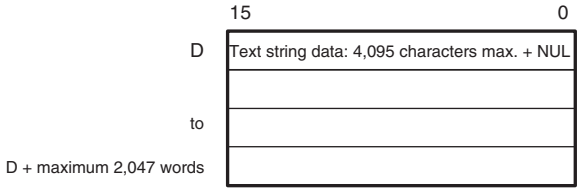
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String



S2: Number of Characters (0000 to 0FFF hex or &0 to &4095)



Note

1. The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words must be in the same area.
2. The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

Operand Specifications

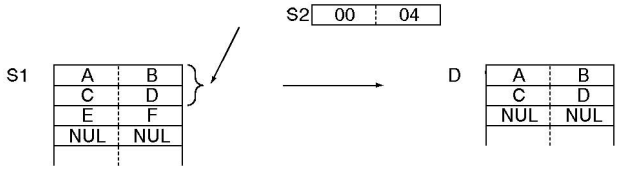
Area	S1	S2	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A447 A448 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0000 to #0FFF (binary) or &0 to &4095	---
Data Registers	---	DR0 to DR15	---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

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.



Note LEFT$(652) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1. ON if more than 4,095 characters (0FFF hex) are designated by S2. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

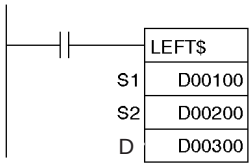
Precautions

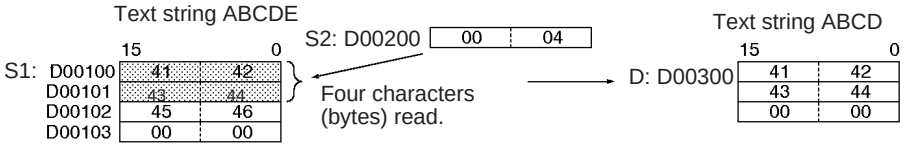
The maximum number of characters to be read that can be designated by S2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, LEFT$(652) is used to read four characters.



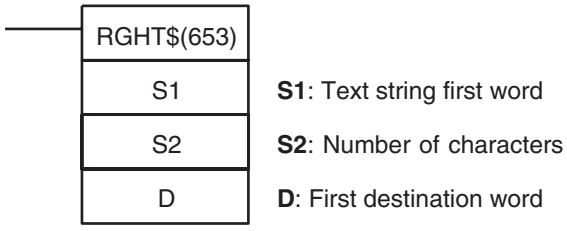


3-33-5 GET STRING RIGHT: RGHT$(653)

Purpose

Reads a designated number of characters from the right (end) of a text string.

Ladder Symbol



S1: Text string first word

S2: Number of characters

D: First destination word

Variations

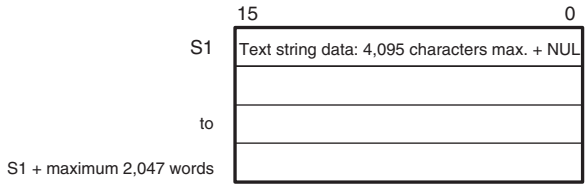
Variations		
	Executed Each Cycle for ON Condition	RGHT$(653)
	Executed Once for Upward Differentiation	@RGHT$(653)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

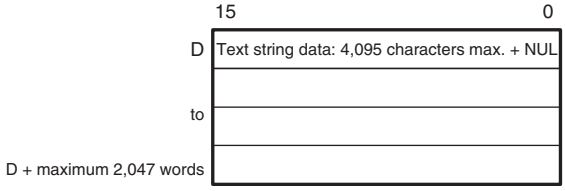
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String



S2: Number of Characters (0000 to 0FFF hex or &0 to &4095)



Note

1. The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words must be in the same area.
2. The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

Operand Specifications

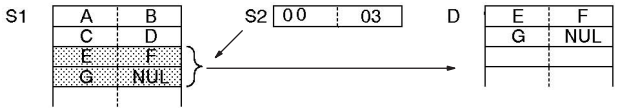
Area	S1	S2	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A447 A448 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0000 to #0FFF (binary) or &0 to &4095	---
Data Registers	---	DR0 to DR15	---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

RGHT$(653) reads the number of characters designated by S2, from the left (the beginning) of the first word of the text string designated by S1 until the NUL code (00 hex), and outputs the result to D (with NUL added at the end).

If the number of characters to be read exceeds the number of characters designated by S1, the entire S1 text string will be output.

If 0 (0000 hex) is designated as the number of characters to be read, the two NUL characters (0000 hex) will be output to D.



Note RGHT$(653) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1. ON if more than 4,095 characters (0FFF hex) are designated by S2. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

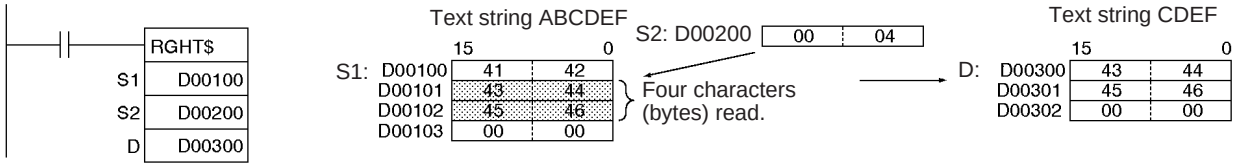
Precautions

The maximum number of characters to be read that can be designated by S2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, RGHT$(653) is used to read four characters.

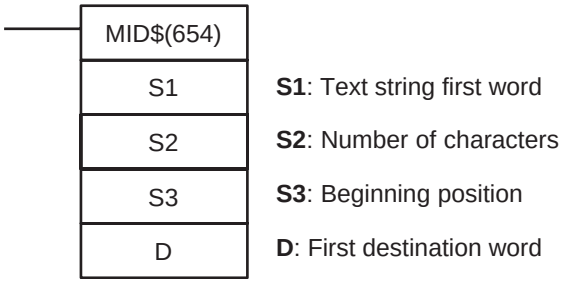


3-33-6 GET STRING MIDDLE: MID$(654)

Purpose

Reads a designated number of characters from any position in the middle of a text string.

Ladder Symbol



S1: Text string first word

S2: Number of characters

S3: Beginning position

D: First destination word

Variations

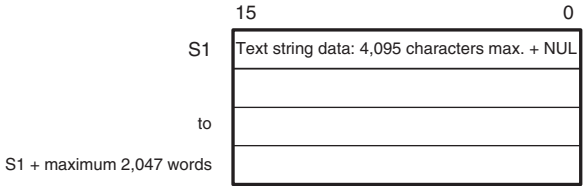
Variations		
	Executed Each Cycle for ON Condition	MID$(654)
	Executed Once for Upward Differentiation	@MID$(654)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

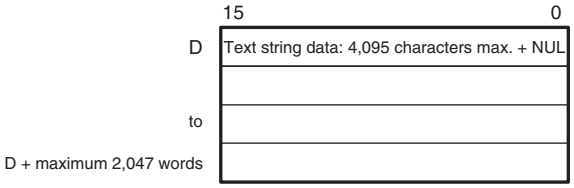
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String



S2: Number of Characters (0000 to 0FFF hex or &0 to &4095)
S3: Beginning Position (0001 to 0FFF hex or &1 to &4095)



Note

1. The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words must be in the same area.
2. The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

Operand Specifications

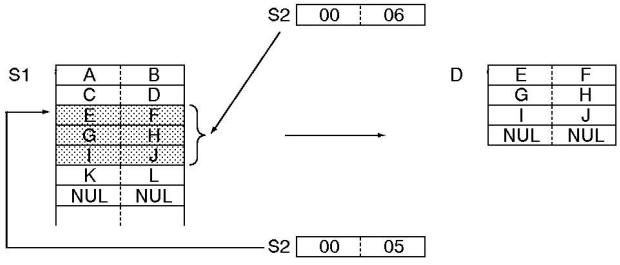
Area	S1	S2	S3	D
CIO Area	CIO 0000 to CIO 6143			
Work Area	W000 to W511			
Holding Bit Area	H000 to H511			
Auxiliary Bit Area	A000 to A447 A448 to A959			A448 to A959
Timer Area	T0000 to T4095			
Counter Area	C0000 to C4095			
DM Area	D00000 to 32767			
EM Area without bank	E00000 to E32767			
EM Area with bank	En_00000 to En_32767 (n = 0 to C)			
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)			

Area	S1	S2	S3	D
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)			
Constants	---	#0000 to #0FFF (binary) or &0 to &4095	#0001 to #0FFF (binary) or &1 to &4095	---
Data Registers	---	DR0 to DR15		
Index Registers	---			
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

Within the text string identified by the first word designated by S1 until the NUL code (00 hex), MID$(654) reads the number of characters designated by S2, from the beginning word designated by S3, and outputs the result to D as text string data (with NUL added at the end).

If the number of characters to be read extends beyond the end of the text string designated by S1, the string will be output up to the end.



Note MID$(654) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1. ON if more than 4,095 characters (0FFF hex) are designated by S2. ON if the S3 data is within the range of 1 to 4,095 (0001 to 0FFF hex). ON if S3 is greater than S1. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

Precautions

The range for the beginning position designated by S3 is the 1st to the 4,095th character (0001 to 0FFF hex). If the setting is outside of this range, an error will be generated and the Error Flag will turn ON.

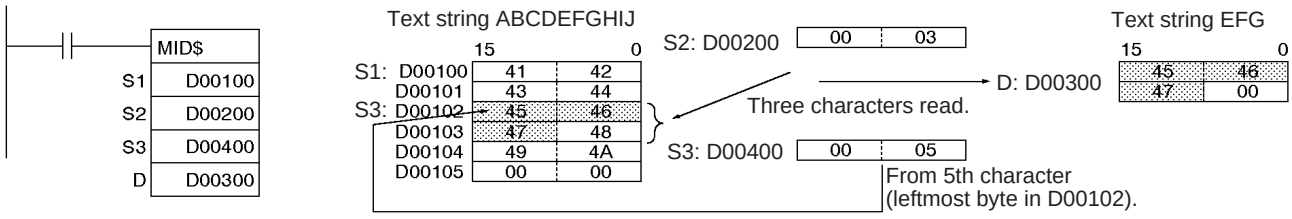
The maximum number of characters to be read that can be designated by S2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If 0 (0000 hex) is designated as the number of characters to be read, the two NUL characters (0000 hex) will be output to D.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, MID$(654) is used to read three characters.

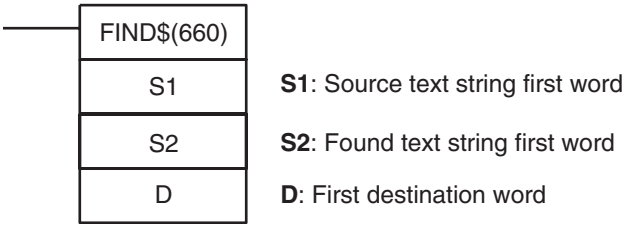


3-33-7 FIND IN STRING: FIND$(660)

Purpose

Finds a designated text string from within a text string.

Ladder Symbol



S1: Source text string first word

S2: Found text string first word

D: First destination word

Variations

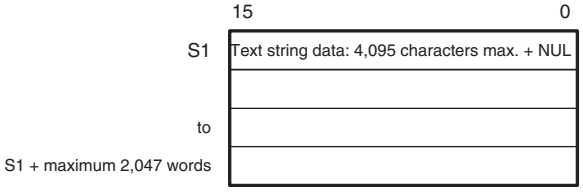
Variations		
	Executed Each Cycle for ON Condition	FIND$(660)
	Executed Once for Upward Differentiation	@FIND$(660)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

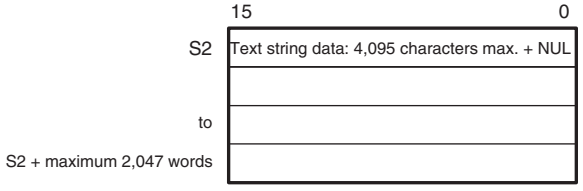
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Source Text String



S2: Found Text String

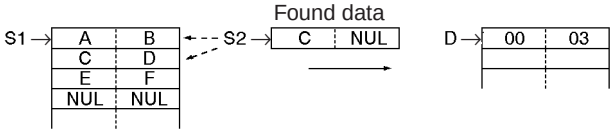


Note The data from S1 to S1 + the maximum 2,047 words and from S2 to S2 + the maximum 2,047 words must be in the same area.

Operand Specifications

Area	S1	S2	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A447 A448 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description FIND$(660) finds the text string designated by S2 from within the text string designated by S1, and outputs the result (a given number of characters from the beginning of S1) in binary data to D. If there is no matching text string, 0000 hex is output to D.



Note FIND$(660) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 or S2. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

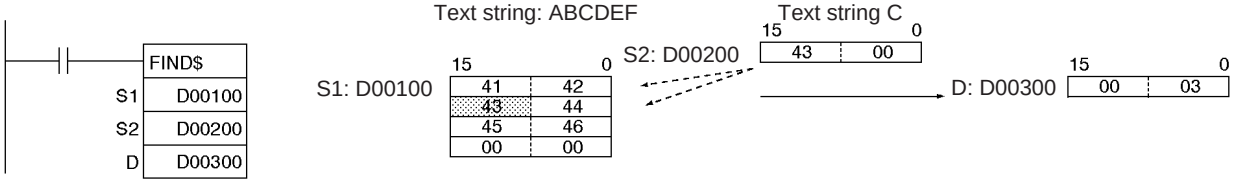
Precautions

The maximum number of characters to be read that can be designated by S1 or S2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, FIND$(660) is used to find one character from within a text string.

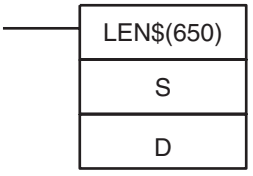


3-33-8 STRING LENGTH: LEN$(650)

Purpose

Calculates the length of a text string.

Ladder Symbol



S: Text string first word

D*:* First destination word

Variations

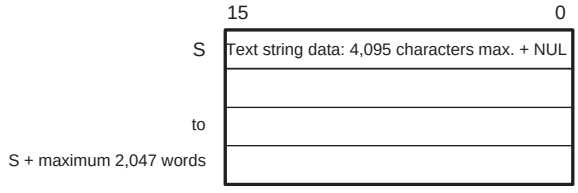
Variations		
	Executed Each Cycle for ON Condition	LEN$(650)
	Executed Once for Upward Differentiation	@LEN$(650)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: Text String

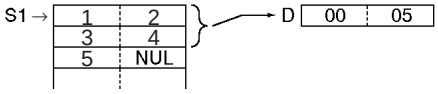


Note The data from S to S + the maximum 2,047 words must be in the same area.

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A447 A448 to A959	A448 to A959
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description 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.



Note LENS$(650) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if the calculated result comes to more than 4,095 characters. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if the calculated result is 0. OFF in all other cases.

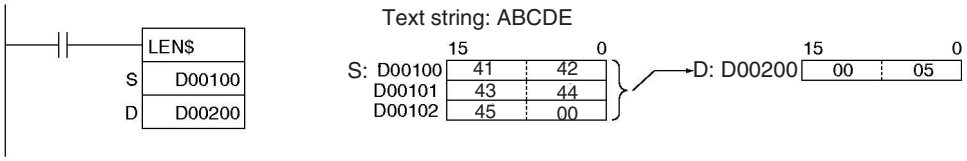
Precautions

The maximum number of characters is 4,095 (0FFF hex). If there are more than that (i.e., if there is no NUL before the 4,096th character), an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Example

In this example, LENS$(650) is used to calculate the number of characters and output the result.

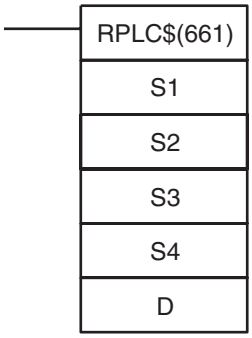


3-33-9 REPLACE IN STRING: RPLC$(661)

Purpose

Replaces a text string with a designated text string from a designated position.

Ladder Symbol



S1: Text string first word

S2: Replacement text string first word

S3: Number of characters

S4: Beginning position

D: First destination word

Variations

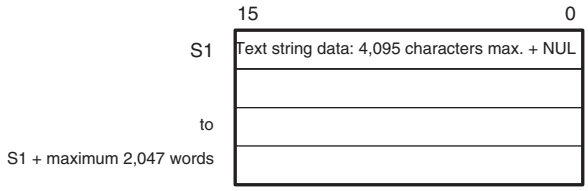
Variations		
	Executed Each Cycle for ON Condition	RPLC$(661)
	Executed Once for Upward Differentiation	@RPLC$(661)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

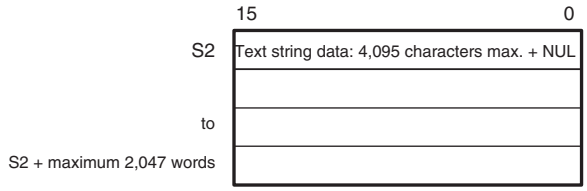
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

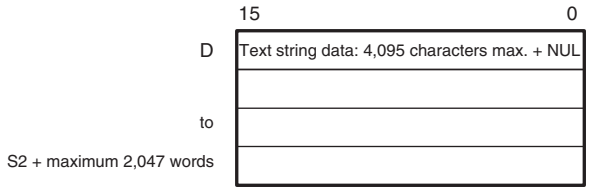
S1: Text String



S2: Replacement Text String



S3: Number of Characters (0000 to 0FFF hex or &0 to &4095)
S4: Beginning Position (0001 to 0FFF hex or &0 to &4095)



Note

1. The data from S1 to S1 + the maximum 2,047 words, from S2 to S2 + the maximum 2,047 words, and from D to D + the maximum 2,047 words must be in the same area.
2. The data from D to D + the maximum 2,047 words and from either S1 to S1 + the maximum 2,047 words or from S2 to S2 + the maximum 2,047 words can overlap.

Operand Specifications

Area	S1	S2	S3	S4	D
CIO Area	CIO 0000 to CIO 6143				
Work Area	W000 to W511				
Holding Bit Area	H000 to H511				
Auxiliary Bit Area	A000 to A447 A448 to A959				A448 to A959
Timer Area	T0000 to T4095				
Counter Area	C0000 to C4095				
DM Area	D00000 to D32767				
EM Area without bank	E00000 to E32767				
EM Area with bank	En_00000 to En_32767 (n = 0 to C)				
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)				
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)				

Area	S1	S2	S3	S4	D
Constants	---		#0000 to #0FFF (binary) or &0 to &4095	#0001 to #0FFF (binary) or &1 to &4095	---
Data Registers	---	DR0 to DR15			---
Index Registers	---				
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15				

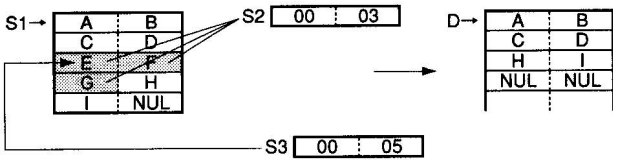
Description

RPLC$(661) replaces part of the text string designated by S1, from the beginning position designated by S4, with the text string designated by S2, and outputs the result to D as text string data (with NUL added at the end). The number of characters to be replaced is designated by S3.

The maximum number of characters in the result is 4,095 (0FFF hex). If the number is greater than that, only 4,095 characters will be output (with NUL added as the 4,096th).

From 0 to 4,095 characters (0000 to 0FFF hex) can be replaced. If the number is 0, then the text string designated by S1 will be output to D just as it is, with no change. If the S2 text string is NUL, then the operation will be the same as deleting the designated range of text in S1.

If the S1 text string from beginning to end is replaced by NUL, then two NUL characters (0000 hex) will be output to D.



Note RPLC$(661) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 or S2. ON if more than 4,095 characters (0FFF hex) are designated by S3. ON if the S4 data is within the range of 1 to 4,095 (0001 to 0FFF hex). ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

Precautions

The maximum number of characters for S1 or S2 is 4,095 (0FFF hex). If there are more than that (i.e., if there is no NUL before the 4,096th character), an error will be generated and the Error Flag will turn ON.

The range for the beginning position designated by S4 is the 1st to the 4,095th character (0001 to 0FFF hex). If the setting is outside of this range, an error will be generated and the Error Flag will turn ON.

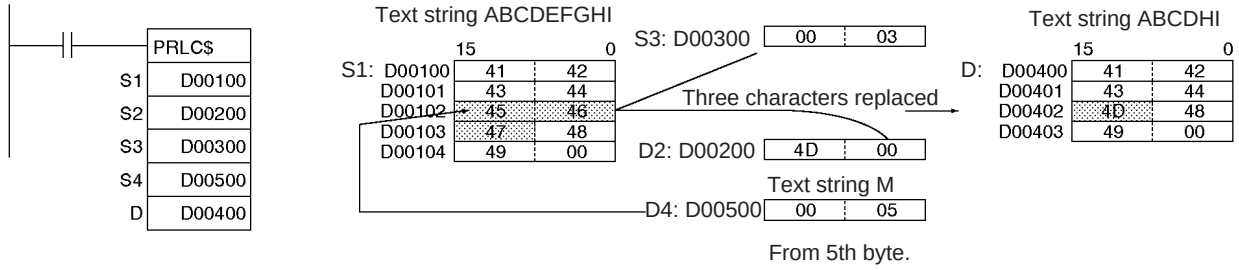
If the beginning position designated by S4 is beyond the text string designated by S1, an error will be generated and the Error Flag will turn ON.

If 0000 (hex) is output to D, the Equals Flag will turn ON.

Set the first destination word D so that it does not overlap with the areas set with the replacement text string first word S2. RPLC$(654) will not work correctly if these areas overlap.

Example

In this example, RPLC$(654) is used to read three characters.

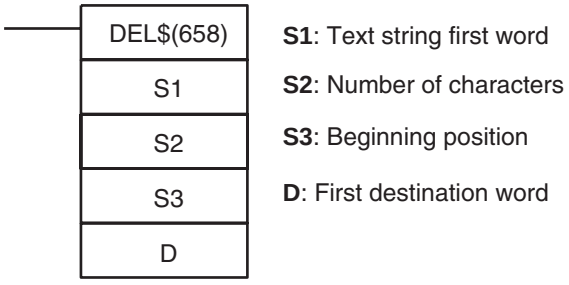


3-33-10 DELETE STRING: DEL$(658)

Purpose

Deletes a designated text string from the middle of a text string.

Ladder Symbol



S1: Text string first word

S2: Number of characters

S3: Beginning position

D: First destination word

Variations

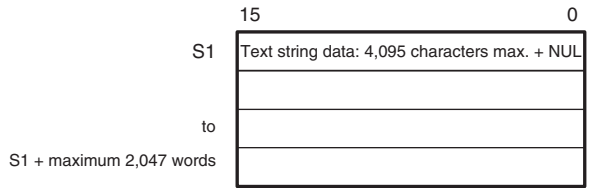
Variations		
	Executed Each Cycle for ON Condition	DEL$(658)
	Executed Once for Upward Differentiation	@DEL$(658)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

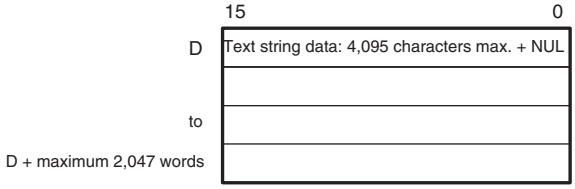
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String



S2: Number of Characters (0000 to 0FFF hex or &0 to &4095)
S3: Beginning Position (0001 to 0FFF hex or &1 to &4095)



Note

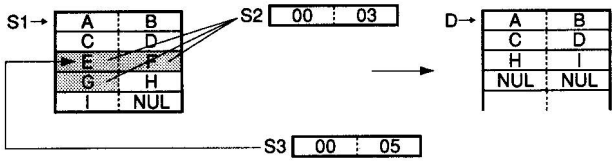
1. The data from S1 to S1 + the maximum 2,047 words, from S2 to S2 + the maximum 2,047 words, and from D to D + the maximum 2,047 words must be in the same area.
2. The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap.

Operand Specifications

Area	S1	S2	S3	D
CIO Area	CIO 0000 to CIO 6143			
Work Area	W000 to W511			
Holding Bit Area	H000 to H511			
Auxiliary Bit Area	A000 to A447 A448 to A959			A448 to A959
Timer Area	T0000 to T4095			
Counter Area	C0000 to C4095			
DM Area	D00000 to D32767			
EM Area without bank	E00000 to E32767			
EM Area with bank	En_00000 to En_32767 (n = 0 to C)			
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)			
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)			
Constants	---	#0000 to #0FFF (binary) or &0 to &4095	#0001 to #0FFF (binary) or &1 to &4095	---
Data Registers	---	DR0 to DR15		---
Index Registers	---			
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

Within the text string 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).



Note DEL$(658) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1. ON if more than 4,095 characters (0FFF hex) are designated by S2. ON if the S3 data is within the range of 1 to 4,095 (0001 to 0FFF hex). ON if S3 is greater than S1. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON when 0000 hex is output to D. OFF in all other cases.

Precautions

The maximum number of characters for S1 is 4,095 (0FFF hex). If there are more than that (i.e., if there is no NUL before the 4,096th character), an error will be generated and the Error Flag will turn ON.

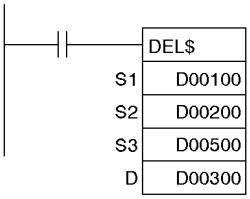
The range for the beginning position designated by S3 is the 1st to the 4,095th character (0001 to 0FFF hex). If the setting is outside of this range, an error will be generated and the Error Flag will turn ON.

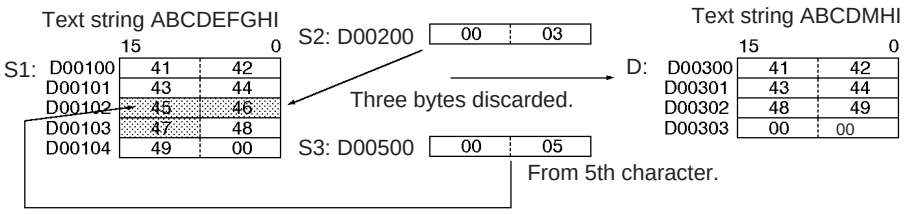
If the number of words specified for S1 exceeds the length of the text string, the Error Flag will turn ON.

If the number of characters to be deleted extends beyond the end of the S1 text string, all of the characters up to the end will be deleted. If all of the characters from the beginning of S1 to the end are designated to be deleted, then 000 hex will be output to D.

Example

In this example, DEL$(658) is used to read three characters.



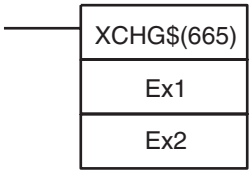


3-33-11 EXCHANGE STRING: XCHG$(665)

Purpose

Replaces a designated text string with another designated text string.

Ladder Symbol



Ex1: First exchange word 1

Ex2: First exchange word 2

Variations

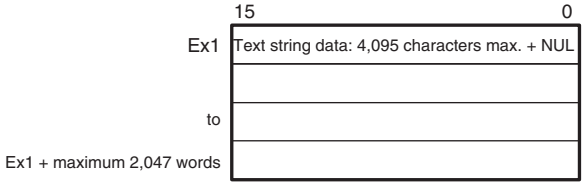
Variations		
	Executed Each Cycle for ON Condition	XCHG$(665)
	Executed Once for Upward Differentiation	@XCHG$(665)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

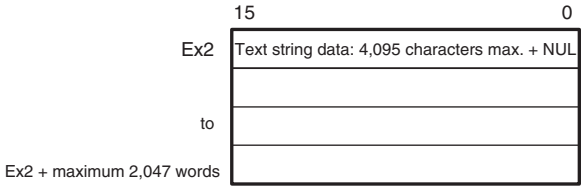
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

Ex1: First Exchange Word 1



Ex2: First Exchange Word 2



Note

1. The data from Ex1 to Ex1 + the maximum 2,047 words and from Ex2 to Ex2 + the maximum 2,047 words must be in the same area.
2. The data from Ex1 to Ex1 + the maximum 2,047 words and from Ex2 to Ex2 + the maximum 2,047 words cannot overlap.

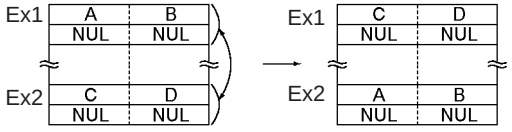
Operand Specifications

Area	Ex1	Ex2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	

Area	Ex1	Ex2
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

XCHG$(665) exchanges the text string designated by Ex1 with the text string designated by Ex2. If either Ex1 or Ex2 is NUL, then two NUL characters (0000 hex) will be output to the other one of them.



Note XCHG$(665) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by Ex1 or Ex2. ON the Ex1 and Ex2 data overlap. ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

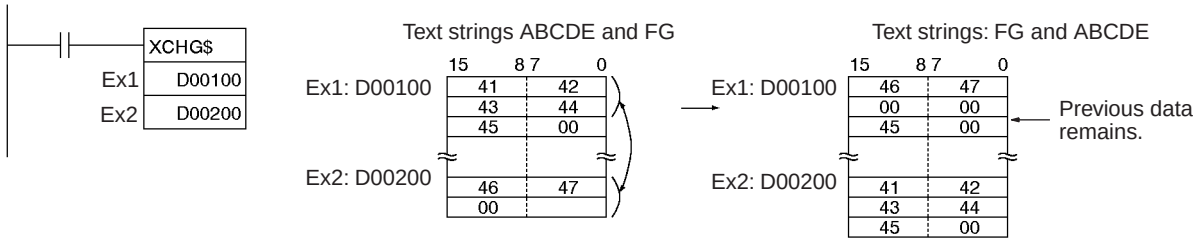
Precautions

The maximum number of characters that can be designated by Ex1 or Ex2 is 4,095 (0FFF hex). If more than that are designated, an error will be generated and the Error Flag will turn ON.

If the text string data designated by Ex1 and Ex2 overlaps, an error will be generated and the Error Flag will turn ON.

Example

In this example, XCHG$(665) is used to exchange two text strings.

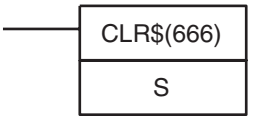


3-33-12 CLEAR STRING: CLR$(666)

Purpose

Clears an entire text string with NUL (00 hex).

Ladder Symbol



S: Text string first word

Variations

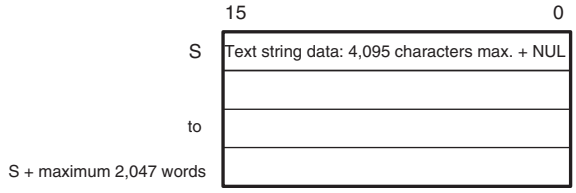
Variations	**Executed Each Cycle for ON Condition**	CLR$(666)
	Executed Once for Upward Differentiation	@CLR$(666)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: Text String First Word



Note The data from S to S + the maximum 2,047 words must be in the same area.

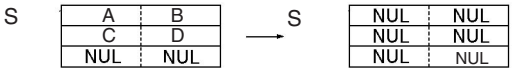
Operand Specifications

Area	S
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)

Area	S
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15

Description

CLR$(666) clears with NUL (00 hex) the entire text string from the first word designated by S until the NUL code (00 hex). The maximum number of characters that can be cleared is 4,096. If there is no NUL before the 4,096 character, only 4,096 characters will be cleared.



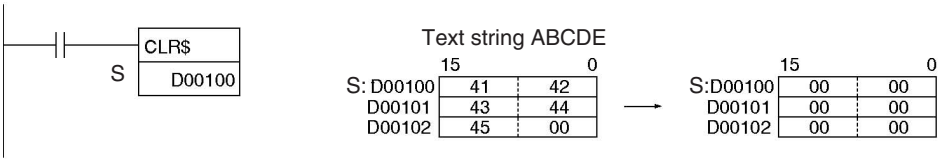
Note CLR$(666) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.

Example

In this example, CLR$(666) is used to clear text string ABCDE.

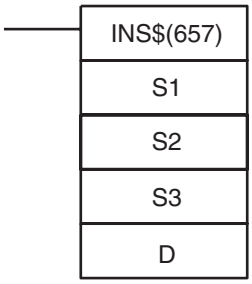


3-33-13 INSERT INTO STRING: INS$(657)

Purpose

Deletes a designated text string from the middle of a text string.

Ladder Symbol



S1: Base text string first word

S2: Inserted text string first word

S3: Beginning position

D: First destination word

Variations

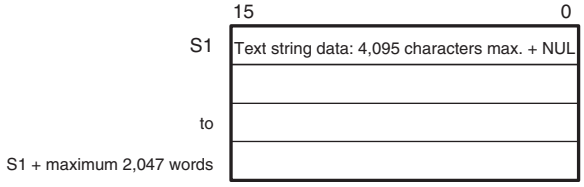
Variations		
	Executed Each Cycle for ON Condition	INS$(657)
	Executed Once for Upward Differentiation	@INS$(657)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

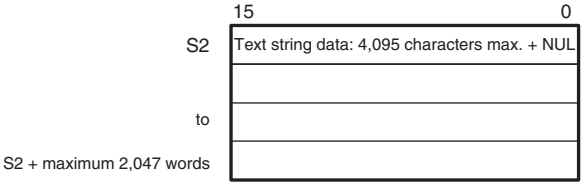
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

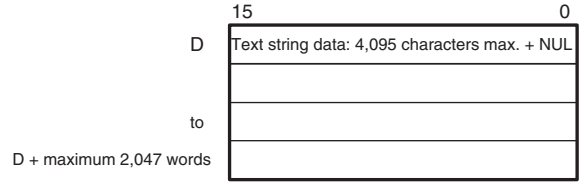
S1: Base Text String



S2: Inserted Text String



S3: Beginning Position (0000 to 0FFF hex or &0 to &4095)



Note

1. The data from S1 to S1 + the maximum 2,047 words, from S2 to S2 + the maximum 2,047 words, and from D to D + the maximum 2,047 words must be in the same area.
2. The data from S2 to S2 + the maximum 2,047 words and from D to D + the maximum 2,047 words cannot overlap. The data from S1 to S1 + the maximum 2,047 words and from D to D + the maximum 2,047 words can overlap. The data from S1 to S1 + the maximum 2,047 words and from S2 to S2 + the maximum 2,047 words can also overlap.

Operand Specifications

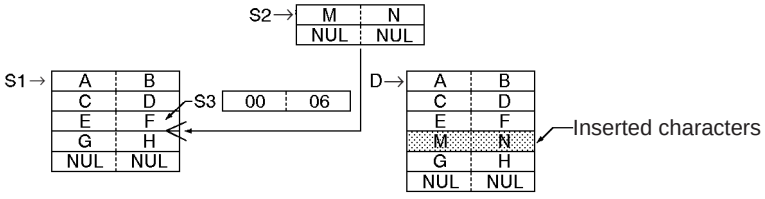
Area	S1	S2	S3	D
CIO Area	CIO 0000 to CIO 6143			
Work Area	W000 to W511			
Holding Bit Area	H000 to H511			
Auxiliary Bit Area	A000 to A447 A448 to A959			A448 to A959
Timer Area	T0000 to T4095			
Counter Area	C0000 to C4095			
DM Area	D00000 to D32767			
EM Area without bank	E00000 to E32767			
EM Area with bank	En_00000 to En_32767 (n = 0 to C)			
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)			
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)			
Constants	---		#0000 to #0FFF (binary) or &0 to &4095	---
Data Registers	---		DR0 to DR15	---
Index Registers	---			
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15			

Description

Within the text string designated by S1, INS$(657) inserts the text string designated by S2, after the beginning word designated by S3, and outputs the result to D as text string data (with NUL added at the end).

The maximum number of characters that can be inserted is 4,095 (0FFF hex). If there are more than that, only 4,095 characters will be output to D (with NUL added as the 4,096th character).

If either S1 or S2 is NUL, then the text string designated by the other one of them will be output to D just as it is. If S1 and S2 are both NUL, then two NUL characters (0000 hex) will be output to D.



Note INS$(657) can be processed in the background. Refer to the *SYSMAC CS/CJ/NSJ Series PLC Programming Manual* (W394) for details.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 or S2. ON if S3 exceeds 4,095 (0FFF hex). ON if the Communications Port Enabled Flag for the communications port number specified as the *Com Port number* for *Background Execution* is OFF when background processing is specified. OFF in all other cases.
Equals Flag	=	ON if 0000 (hex) is output to D. OFF in all other cases.

Precautions

The maximum number of characters for S1 and S2 is 4,095 (0FFF hex). If there are more than that (i.e., if there is no NUL before the 4,096th character), an error will be generated and the Error Flag will turn ON.

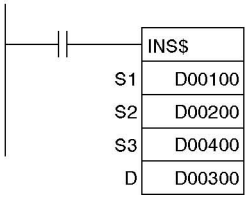
The range for the beginning position designated by S3 is 0 to 4,095. If the setting is outside of this range, an error will be generated and the Error Flag will turn ON.

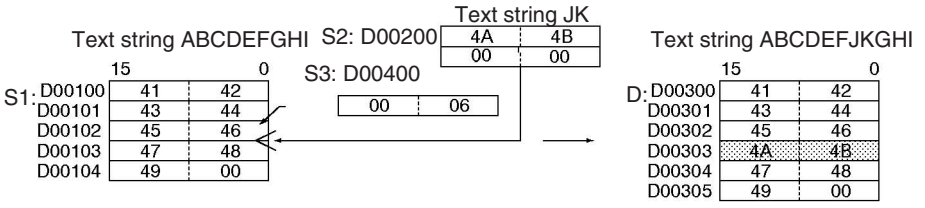
If 0000 (hex) is output to D, the Equals Flag will turn ON.

Do not overlap the destination words designated by D with the text string data designated by S2. If these overlap, the operation will not be executed properly.

Example

In this example, INS$(657) is used to insert two characters.





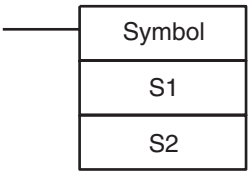
3-33-14 String Comparison Instructions (670 to 675)

Purpose

Sting comparison instructions (=$, <>$, <$, <=$, >$, >=$) compare two text strings from the beginning, in terms of value of the ASCII codes. If the result of the comparison is true, an ON execution condition is created for a LOAD, AND, or OR.

Ladder Symbol

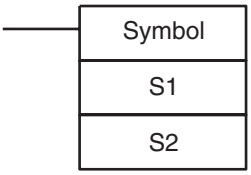
LD (Load)



S1: Text string 1

S2: Text string 2

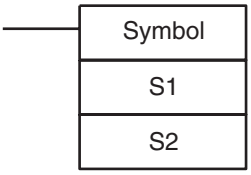
AND (Series Connection)



S1: Text string 1

S2: Text string 2

OR (Parallel Connection)



S1: Text string 1

S2: Text string 2

Variations

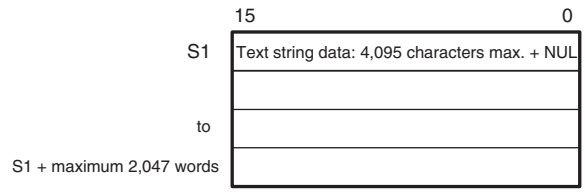
Variations	Creates ON Each Cycle Comparison is True	String comparison instructions
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

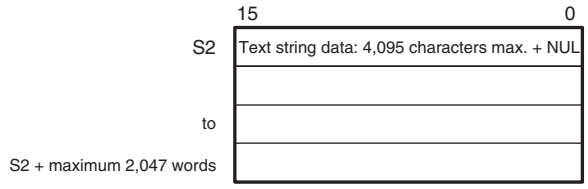
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S1: Text String 1



S2: Text String 2



Note

1. The data from S1 to S1 + the maximum 2,047 words and from S2 to S2 + the maximum 2,047 words be in the same area.
2. The data from S1 to S1 + the maximum 2,047 words and from S2 to S2 + the maximum 2,047 words cannot overlap.

Operand Specifications

Area	S1	S2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W511	
Holding Bit Area	H000 to H511	
Auxiliary Bit Area	A000 to A447 A448 to A959	
Timer Area	T0000 to T4095	
Counter Area	C0000 to C4095	
DM Area	D00000 to D32767	
EM Area without bank	E00000 to E32767	
EM Area with bank	En_00000 to En_32767 (n = 0 to C)	
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)	
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15	

Description

String comparison instructions compare the text strings designated by S1 and S2. If the result of the comparison is true, an ON execution condition is created in the ladder diagram. The maximum number of characters for either S1 or S2 is 4,095 (0FFF hex).

String comparison instructions are expressed using the 18 different mnemonics listed below. (LD, AND, and OR do not appear in the ladder diagram.)

LD=$, AND=$, OR=$
LD<>$, AND<>$, OR<>$
LD<$, AND<$, OR<$

LD<=$, AND<=$, OR<=$
LD>$, AND>$, OR>$
LD>=$, AND>=$, OR>=$

The following table provides details on these instructions.

Mnemonic (including function code)	Name	Function
LD=$(670)	LOAD STRING EQUALS	True when S1 text string equals S2 text string.
AND=$(670)	AND STRING EQUALS	
OR=$(670)	OR STRING EQUALS	
LD<>$(671)	LOAD STRING NOT EQUAL	True when S1 text string does not equal S2 text string.
AND<>$(671)	AND STRING NOT EQUAL	
OR<>$(671)	OR STRING NOT EQUAL	
LD<$(672)	LOAD STRING LESS THAN	True when S1 text string is less than S2 text string.
AND<$(672)	AND STRING LESS THAN	
OR<$(672)	OR STRING LESS THAN	
LD<=$(673)	LOAD STRING LESS THAN OR EQUALS	True when S1 text string is less than or equal to S2 text string.
AND<=$(673)	AND STRING LESS THAN OR EQUALS	
OR<=$(673)	OR STRING LESS THAN OR EQUALS	
LD>$(674)	LOAD STRING GREATER THAN	True when S1 text string is greater than S2 text string.
AND>$(674)	AND STRING GREATER THAN	
OR>$(674)	OR STRING GREATER THAN	
LD>=$(675)	LOAD STRING GREATER THAN OR EQUALS	True when S1 text string is greater than or equal to S2 text string.
AND>=$(675)	AND STRING GREATER THAN OR EQUALS	
OR>=$(675)	OR STRING GREATER THAN OR EQUALS	

Comparison Methods

The comparison methods are as follows:

The first character (byte) of each text string is compared with its counterpart from the other string as ASCII code. If the two ASCII codes are not equal, then that greater/lesser relationship becomes the greater/lesser relationship for the two text strings. If the two ASCII codes are equal, the next characters are compared. If these two ASCII codes are not equal, then, that greater/lesser relationship becomes the greater/lesser relationship for the two text strings.

In this manner, the two text strings are compared in order, character by character. If all of the characters, including the NUL, are equal, then the two text strings will have an equal relationship.

If the two text strings are of differing lengths, then the NUL (00 hex) will be added to the shorter of the two strings to fill in the difference, and the comparison will be made on that basis.

Comparison Examples

AD (414400 hex) and BC (424300 hex):
AD < BC, because at the beginning of the text strings 41 (hex) is less than 42 (hex).

ADC (41444300 hex) and B (4200 hex):
ADC < B, because at the beginning of the text strings 41 (hex) is less than 42 (hex).

ABC (41424300 hex) and ABD (41424400 hex):
ABC < ABD, because at the beginning of the text strings the 41s and 42s match, so the result is determined by 43 being less than 44.

ABC (41424300 hex) and AB (414200 hex):
ABC > AB, because at the beginning of the text strings the 41s and 42s match, so the result is determined by 43 being greater than 00.

AB (414200 hex) and AB (414200 hex):
AB = AB, because the 41s, the 42s, and the 00s all match.

Continue programming one instruction after another, treating LD, AND, and OR in the same way. LD and OR instructions can be connected directly to the bus bar, but AND instructions cannot.

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 4,095 characters are designated by S1 or S2. OFF in all other cases.
Greater Than Flag	>	ON if the comparison results in S1 greater than S2. OFF in all other cases.
Greater Than or Equals Flag	>=	ON if the comparison results in S1 greater than or equal to S2. OFF in all other cases.
Equals Flag	=	ON if the comparison results in S1 equal to S2. OFF in all other cases.
Not Equal Flag	<>	ON if the comparison results in S1 not equal to S2. OFF in all other cases.
Less Than Flag	<	ON if the comparison results in S1 less than S2. OFF in all other cases.
Less Than or Equals Flag	<=	ON if the comparison results in S1 less than or equal to S2. OFF in all other cases.

Note String comparison instructions are used to rearrange the order of text strings in order of ASCII. For example, the ASCII order from lower to higher is the order of the alphabet from A to Z, so text strings can be arranged in alphabetical order.

Precautions

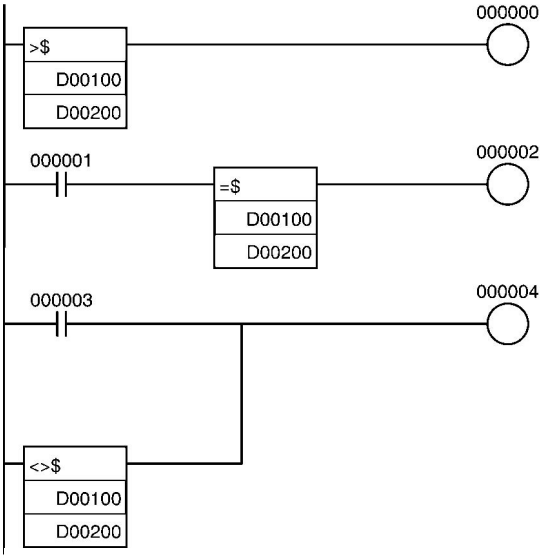
Please a right-hand instruction after these instructions. The String Comparison Instructions cannot appear on the right side of the ladder diagram.

These instructions cannot be used on the last rung of a logic block.

The maximum number of characters that can be compared is 4,095 (0FFF hex). If that number is exceeded (i.e., if there is no NUL before the 4,096th character), an error will occur and the Error Flag will turn ON. When this happens, an OFF execution condition will be output to the next instruction.

Example

In this example, string comparison instructions are used to compare data.



Address	Mnemonic	Operand
000000	LD > $	--- D00100 D00200
000001	OUT	000000
000002	LD	000001
000003	AND=$	--- D00100 D00200
000004	OUT	000002
000005	LD	000003
000006	OR <> $	--- D00100 D00200
000007	OUT	000004

> $	= $	<>$
D00100	D00100	D00100
D00200	D00200	D00200

Text string ABCD

D00100	41	42
D00101	44	43
D00102	00	00

Text string ABC

D00200	41	42
D00201	43	00

ON OFF ON

Text string ABC

D00100	41	42
D00101	43	00

Text string ABC

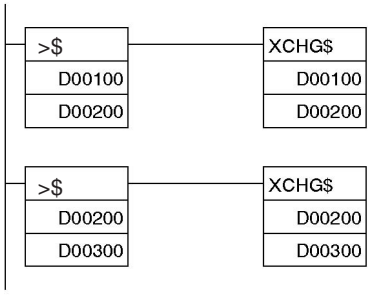
D00200	41	42
D00201	43	00

OFF ON OFF

In this example, three text strings are rearranged in alphabetical order. The original order is as follows:

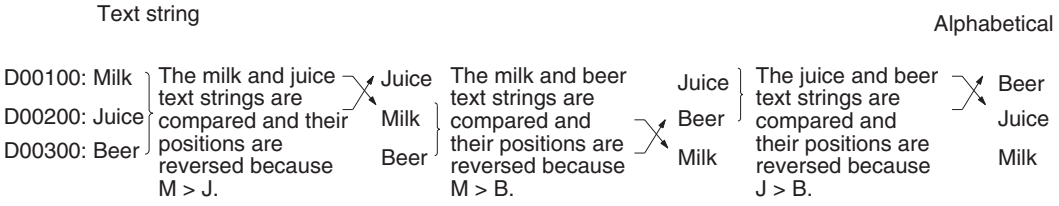
D00100: Milk
D00200: Juice
D00300: Beer

When rearranged alphabetically, the order changes as follows: beer, juice, milk.



Two text strings beginning with D00100 and D00200 are compared in ASCII order from lower to higher. If the text string beginning with D00100 is higher in ASCII order than the one beginning with D00200, then the position of the two text strings will be reversed.

Two text strings beginning with D00200 and D00300 are compared in ASCII order from lower to higher. If the text string beginning with D00200 is higher in ASCII order than the one beginning with D00300, then the position of the two text strings will be reversed.



In this way, three text strings can be rearranged in alphabetical order.

3-34 Task Control Instructions

This section describes instructions used to control tasks.

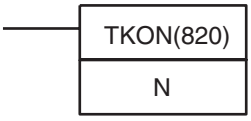
Instruction	Mnemonic	Function code	Page
TASK ON	TKON	820	1255
TASK OFF	TKOF	821	1258

3-34-1 TASK ON: TKON(820)

Purpose

Makes the specified task executable. Also, causes an interrupt task to operate as an extra cyclic task. (Extra cyclic tasks are supported by CS1-H, CJ1-H, and CJ1M CPU Units only.)

Ladder Symbol



N: Task number

Variations

Variations	**Executed Each Cycle for ON Condition**	TKON(820)
	Executed Once for Upward Differentiation	@TKON(820)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	Not allowed

Operands

N: Task number

The allowed range for N depends on the kind of task being specified.

- Cyclic tasks:
  N must be a constant between 0 and 31 decimal. (Values 0 to 31 specify cyclic tasks 0 to 31.)
- Extra cyclic tasks (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	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---

Area	N
Constants	00 to 31 or 8000 to 8255 (decimal)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

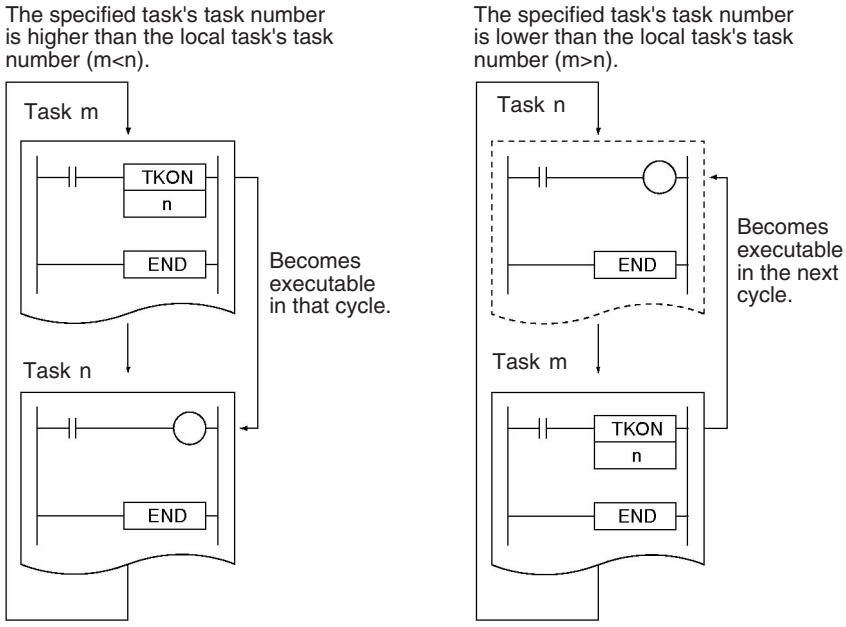
Description

TKON(820) puts the specified cyclic task or extra cyclic task in executable status. When N is 0 to 31 (specifying a cyclic task), the corresponding Task Flag (TK00 to TK31) will be turned ON at the same time.

This instruction can be executed only in a regular cyclic task or an extra cyclic task. An error will occur if an attempt is made to execute it in an interrupt task.

The cyclic task or extra cyclic task specified in TKON(820) will be also be executable in later cycles as long as it is not put in standby status by TKOF(821).

Any task can be made executable from any cyclic task, although the specified task will not be executed until the next cycle if its task number is lower than the task number of the local task. The task will be executed in the same cycle if its task number is higher than the local task's task number.



TKON(820) will be treated as NOP(000) if the specified task is already executable or the local task is specified.

A task in executable status can be put in standby status with TKOF(821), the CX-Programmer, or a FINS command.

The terms executable and executing are not interchangeable. Executable tasks are executed in order of their task numbers during cyclic program execution. An executable task will not be executed if it is put in standby status before program execution reaches its task number.

Note

1. The CX-Programmer's *General Properties Tab* for each task has a setting (the *Operation start* box) that specifies whether the cyclic task will be executable at startup. When the *Operation start* box has been checked, the corresponding cyclic task will be put in executable status automatically when the PLC begins operation. All other cyclic tasks will be in non-executable status.

(If the memory all clear operation is executed from the Programming Console, however, cyclic task 0 will automatically be made executable.)

2. If a task is in non-executable status, TKON(820) can executed to put that task into executable status. Likewise, a cyclic task in executable status can be put into non-executable status with the TKOF(821) instruction.
3. Cyclic tasks or extra cyclic tasks that were made executable will be put in executable status in that cycle in task-number order. Consequently, a task will not be executed if it is put into standby status before the cycle's processing reaches that task as each task is executed in task-number order.

Flags

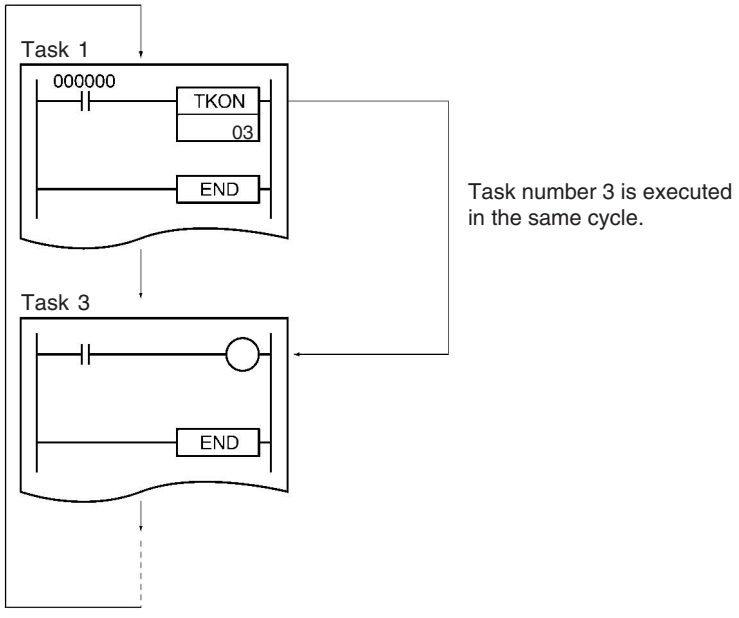
Name	Label	Operation
Error Flag	ER	ON if N is not a constant between 00 and 31 or between 8000 and 8255 (CS1-H, CJ1-H, and CJ1M CPU Units only). ON if the task specified with N does not exist. ON if TKON(820) is executed in an interrupt task. OFF in all other cases.

Name	Addresses	Operation
Task Flags	TK00 to TK31	These flags are turned ON when the corresponding cyclic task is executable and they are OFF when the corresponding cyclic task is not executable or in standby status. TK00 to TK31 correspond to cyclic task numbers 00 to 31.

Examples

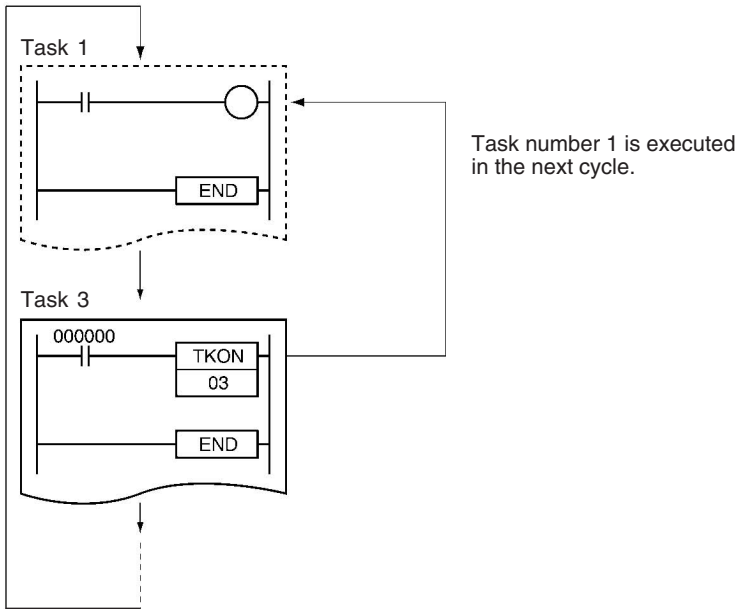
Specifying a Later Task

When CIO 000000 is ON in the following example, task number 3 is made executable in task number 1. Task number 3 will be executed in the same cycle when program execution reaches task number 3.



Specifying an Earlier Task

When CIO 000000 is ON in the following example, task number 1 is made executable in task number 3. Task number 1 will be executed in the next cycle when program execution reaches task number 1.

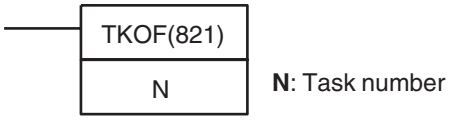


3-34-2 TASK OFF: TKOF(821)

Purpose

Puts the specified cyclic task or extra cyclic task into standby status, i.e., disables execution of the task. (Extra cyclic tasks are supported by CS1-H, CJ1-H, and CJ1M CPU Units only.)

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	TKOF(821)
	Executed Once for Upward Differentiation	@TKOF(821)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	Not allowed

Operands

N: Task number

The allowed range for N depends on the kind of task being specified.

- Cyclic tasks:
  N must be a constant between 0 and 31 decimal. (Values 0 to 31 specify cyclic tasks 0 to 31.)
- Extra cyclic tasks (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	N
CIO Area	---
Work Area	---
Holding Bit Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
EM Area without bank	---
EM Area with bank	---
Indirect DM/EM addresses in binary	---
Indirect DM/EM addresses in BCD	---
Constants	00 to 31 or 8000 to 8255 (decimal)
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	---

Description

TKOF(821) puts the specified cyclic task or extra cyclic into standby status and turns OFF the corresponding Task Flag (TK00 to TK31).

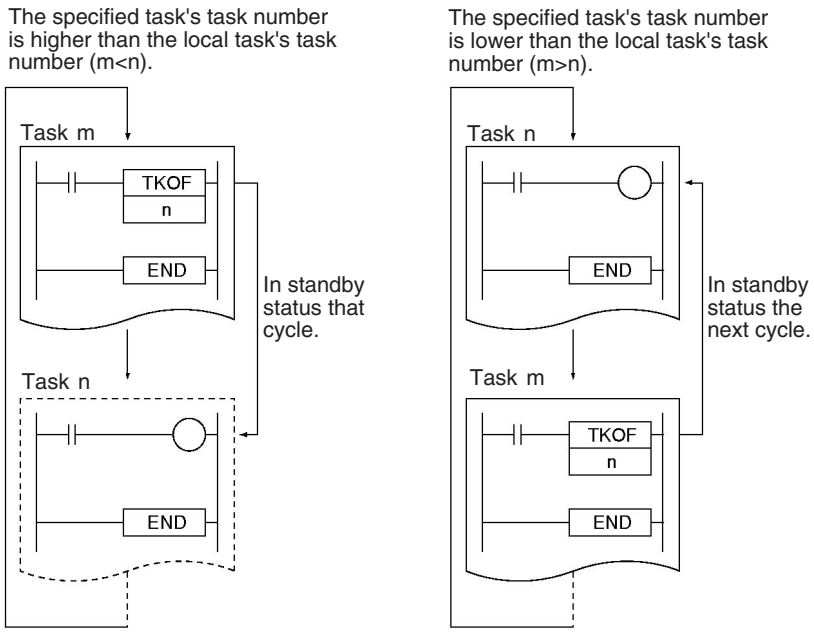
The task specified in TKOF(821) will be also be in standby status in later cycles as long as it is not put into executable status by TKON(820), a Peripheral Device running CX-Programmer, or a FINS command.

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.

Note

1. The CX-Programmer's *General Properties Tab* for each task has a setting (the *Operation start* box) that specifies whether the cyclic task will be executable at startup. When the *Operation start* box has been checked, the corresponding cyclic task will be put in executable status automatically when the PLC begins operation. All other cyclic tasks will be in non-executable status.
   (If the memory all clear operation is executed from the Programming Console, however, cyclic task 0 will automatically be made executable.)
2. If a task is in non-executable status, TKON(820) can executed to put that task into executable status. Likewise, a cyclic task in executable status can be put into non-executable status with the TKOF(821) instruction.
3. Cyclic tasks or extra cyclic tasks that are in executable status can be put into standby status by the TKOF(821) instruction.



A regular task that has been set to be executed at startup will be put in executable status automatically when the PLC begins operation. All other regular tasks will be in non-executable status.

A task in executable status can be put in standby status with TKOF(821), a Peripheral Device running CX-Programmer, or a FINS command.

The terms executable and executing are not interchangeable. Executable tasks are executed in order of their task numbers during cyclic program execution. An executable task will not be executed if it is put in standby status before program execution reaches its task number.

Unlike TKON(820), this instruction can be placed in interrupt tasks as well as in cyclic tasks.

Flags

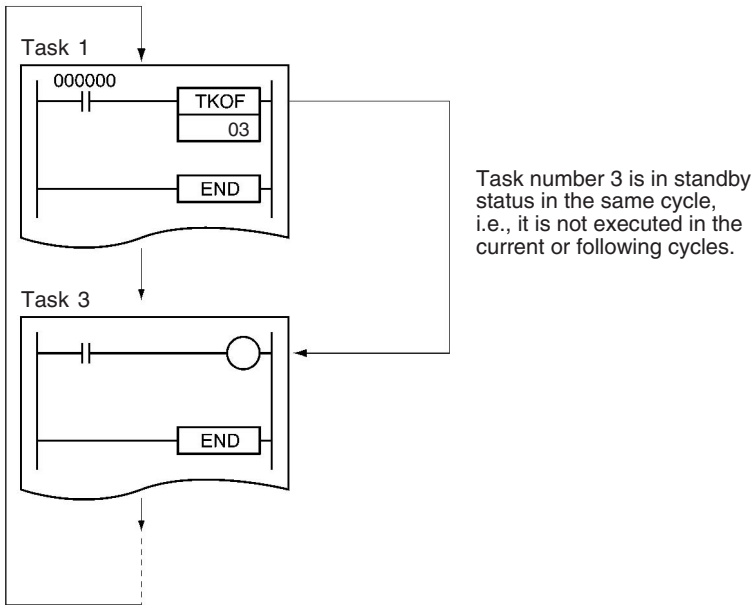
Name	Label	Operation
Error Flag	ER	ON if N is not a constant between 00 and 31 or between 8000 and 8255 (CS1-H, CJ1-H, and CJ1M CPU Units only). ON if the task specified with N does not exist. ON if TKOF(821) is executed in an interrupt task. OFF in all other cases.

Name	Addresses	Operation
Task Flags	TK00 to TK31	These flags are turned ON when the corresponding cyclic task is executable and they are OFF when the corresponding cyclic task is not executable or in standby status. TK00 to TK31 correspond to cyclic task numbers 00 to 31.

Examples

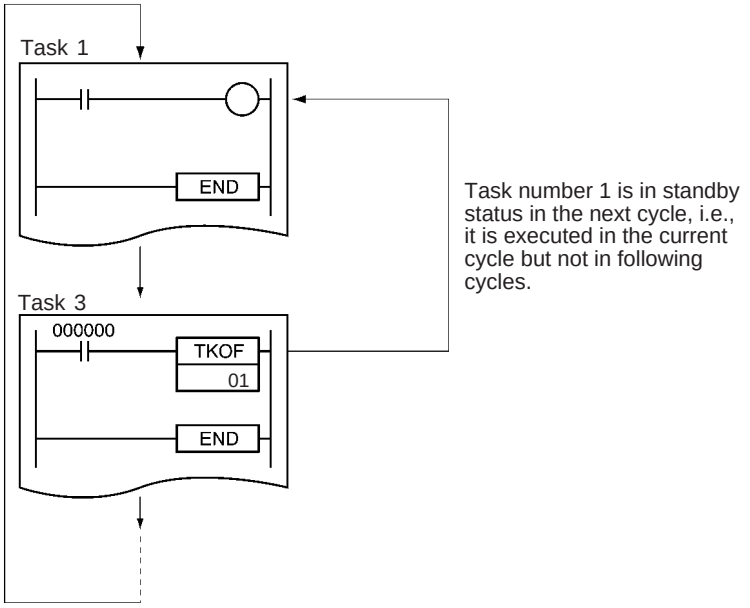
Specifying a Later Task

When CIO 000000 is ON in the following example, task number 3 is put into standby status in task number 1. Task number 3 will be not be executed in the that cycle when program execution reaches task number 3.



Specifying an Earlier Task

When CIO 000000 is ON in the following example, task number 1 is put into standby status in task number 3. Task number 1 will be not be executed in the next cycle when program execution reaches task number 1.



3-35 Model Conversion Instructions (Unit Ver. 3.0 or Later)

This section describes instructions used when changing PLC models.

Instruction	Mnemonic	Function code	Page
BLOCK TRANSFER	XFERC	565	1263
SINGLE WORD DISTRIBUTE	DISTC	566	1266

Instruction	Mnemonic	Function code	Page
DATA COLLECT	COLLC	567	1269
MOVE BIT	MOVBC	568	1273
BIT COUNTER	BCNTC	621	1275

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 or 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 (Unit Ver. 3.0 or later)	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.
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)	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.

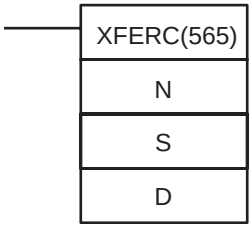
- The Error Flag will turn ON if the data for the above operands is not BCD.
- For DISTC(566), the operation of the Conditions Flags was added for the stack push operation.
- For COLLC(567), the operation of the Conditions Flags was added for the stack read operation.

3-35-1 BLOCK TRANSFER: XFERC(565)

Purpose

Transfers the specified number of consecutive words.

Ladder Symbol



N: Number of words

S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	XFERC(565)
	Executed Once for Upward Differentiation	@XFERC(565)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

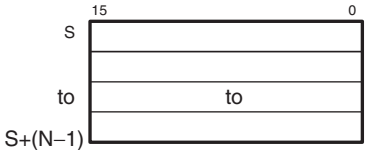
Operands

N: Number of Words

Specifies the number of words to be transferred. The possible range for N is 0000 to 9999 BCD.

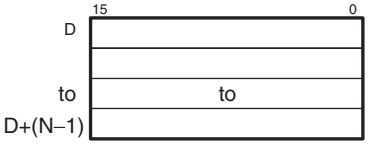
S: First Source Word

Specifies the first source word.



D: First Destination Word

Specifies the first destination word.



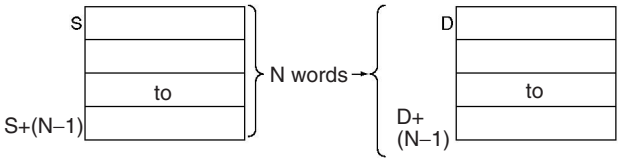
Operand Specifications

Area	N	S	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #9999 (BCD)	---	---
Data Registers	DR0 to DR15	---	

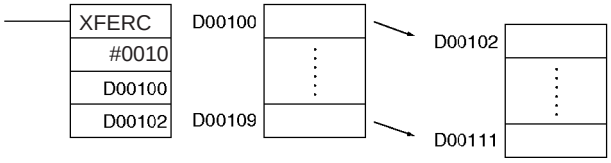
Area	N	S	D
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

XFERC(565) copies N words beginning with S (S to S+(N–1)) to the N words beginning with D (D to D+(N–1)).



It is possible for the source words and destination words to overlap, so XFERC(565) can perform word-shift operations.



Flags

Name	Label	Operation
Error Flag	ER	ON if the data in N (the number of words) is not BCD.

Note In C-series PLCs, the BLOCK TRANSFER (XFER) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. XFERC(565) will not cause the Error Flag to go ON in these cases.

Precautions

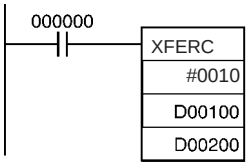
Be sure that the source words (S to S+N–1) and destination words (D to D+N–1) do not exceed the end of the data area.

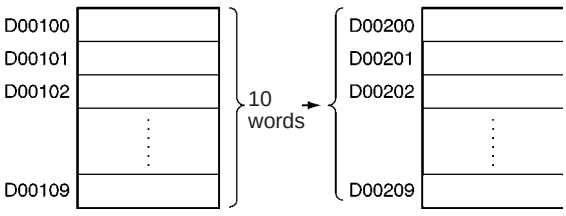
Some time will be required to complete XFERC(565) when a large number of words is being transferred. In this case, the XFERC(565) transfer might not be completed if a power interruption occurs during execution of the instruction.

The content of N must be BCD. If N is not BCD, an error will occur and the Error Flag will be turned ON.

Example

When CIO 000000 is ON in the following example, the 10 words D00100 through D00109 are copied to D00200 through D00209.



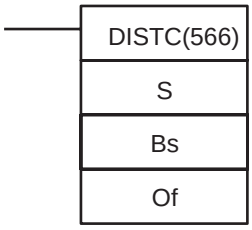


3-35-2 SINGLE WORD DISTRIBUTE: DISTC(566)

Purpose

Transfers the source word to a destination word calculated by adding an offset value to the base address.

Ladder Symbol



S: Source word

Bs: Destination base address

Of: Offset

Variations

Variations	**Executed Each Cycle for ON Condition**	DISTC(566)
	Executed Once for Upward Differentiation	@DISTC(566)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

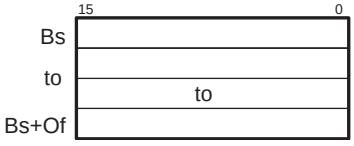
Bs: Destination Base Address

Specifies the destination base address. The offset is added to this address to calculate the destination word.

Of: Offset

- Data Distribution Operation (0000 to 7999 BCD)

  This value is added to the base address to calculate the destination word. The offset can be any value from 0000 to 7999 in BCD, but Bs and Bs+Of must be in the same data area.



- Stack Push Operation (9000 to 9999 BCD)

  When the leftmost digit of Of is 9, the rightmost 3 digits of Of specify the number of words in the stack. The offset can be any value from 9000 to 9999 BCD.

Operand Specifications

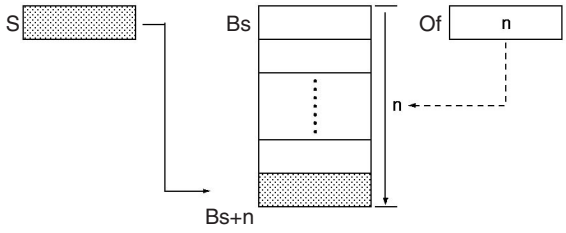
Area	S	Bs	Of
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959	A448 to A959	A000 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		

Area	S	Bs	Of
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	---	#0000 to #7999 for distribution #9000 to #9999 for stack operation
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

Data Distribution Operation

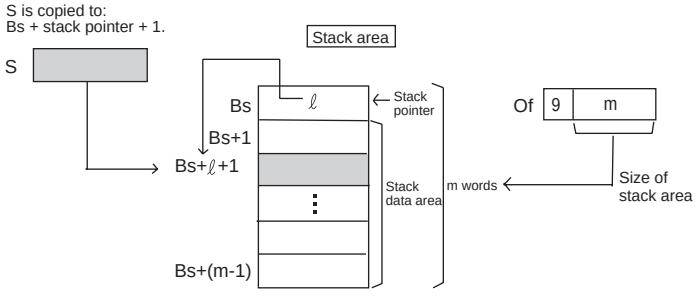
DISTC(566) copies S to the destination word calculated by adding Of to Bs. The same DISTC(566) instruction can be used to distribute the source word to various words in the data area by changing the value of Of.



Stack Push Operation

When the leftmost digit (bits 12 to 15) of Of is 9 BCD, DISTC(566) operates a stack from Bs to Bs+Of-9000. The destination base address (Bs) contains the stack pointer and the rest of the words in the stack contain the stack data.

DISTC(566) copies S to the destination word calculated by adding the stack pointer (content of Bs) + 1 to address Bs. The same DISTC(566) instruction can be used to distribute the source word to various words in the data area by changing the value of Of.



Each time that the content of S is copied to a word in the stack data area, the stack pointer in Bs is automatically incremented by +1.

Note Use COLLC(567) to read stack data from the stack area.

Flags

Name	Label	Operation
Error Flag	ER	ON if Stack Push Operation is specified, but the stack pointer data in Bs is not BCD. ON if Stack Push Operation is specified and the stack pointer indicates a word that exceeds the stack data area.
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.

Note In C-series PLCs, the SINGLE WORD DISTRIBUTE (DIST) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. DISTC(566) will not cause the Error Flag to go ON in these cases.

Precautions

Once DISTC(566) has been executed with Stack Push Operation to allocate a stack area, always specify the same length stack area in subsequent DISTC(566) instructions. Operation will be unreliable if a different stack area size is specified in later DISTC(566) instructions.

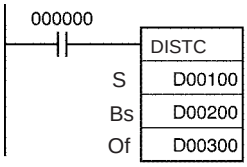
Be sure that the offset or stack size specified by Of does not exceed the end of the data area when added to Bs.

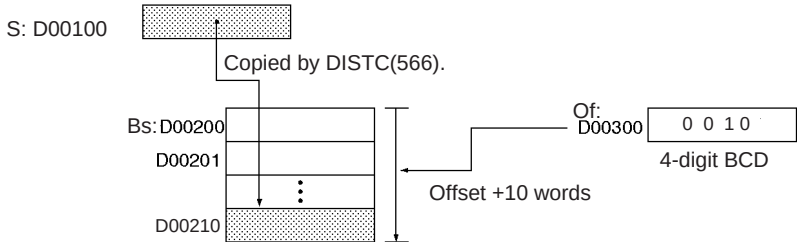
Examples

Data Distribution Operation

The leftmost byte of D00300 is 0, so DISTC(566) performs the Data Distribution Operation.

When CIO 000000 is ON in the following example, the contents of D00100 will be copied to D00210 (D00200 + 10) if the content of D00300 is 0010 BCD. The content of D00100 can be copied to other words by changing the offset in D00300.

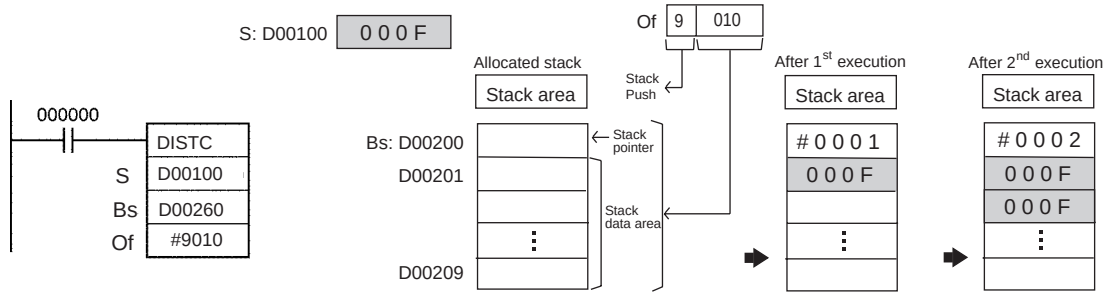




Stack Push Operation

The leftmost byte of Of is 9, so DISTC(566) performs the Stack Push Operation.

When CIO 000000 is ON in the following example, DISTC(566) allocates a 10 word stack area (since the rightmost 3 digits of Of are #010) between D00200 and D00209. At the same time, the contents of D00100 will be copied to the word calculated by adding D00200 + stack pointer +1. Finally, the stack pointer is incremented by +1.

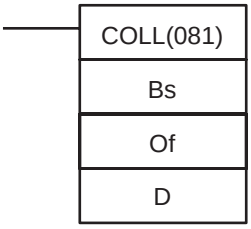


3-35-3 DATA COLLECT: COLLC(567)

Purpose

Transfers the source word (calculated by adding an offset value to the base address) to the destination word.

Ladder Symbol



Bs: Source base address

Of: Offset

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	COLLC(567)
	Executed Once for Upward Differentiation	@COLLC(567)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

Bs: Source Base Address

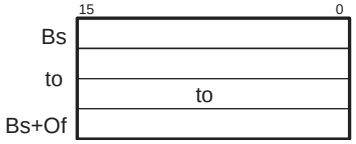
Specifies the source base address. The offset is added to this address to calculate the source word.

Of: Offset

The value of Of determines the operation of COLLC(567).

- Data Collect Operation (Of = 0000 to 7999 BCD)

  The Of value is added to the base address to calculate the source word. The offset can be any value from 0000 to 7999 BCD, but Bs and Bs+Of must be in the same data area.



- LIFO Stack Read Operation (Of = 8000 to 8999 BCD)
  If the leftmost digit of Of is 8, COLLC(567) will operate as a LIFO stack instruction. The stack begins at Bs with a length specified in the rightmost 3 digits of Of.

- FIFO Stack Read Operation (Of = 9000 to 9999 BCD)
  If the leftmost digit of Of is 9, COLLC(567) will operate as a FIFO stack instruction. The stack begins at Bs with a length specified in the rightmost 3 digits of Of.

Operand Specifications

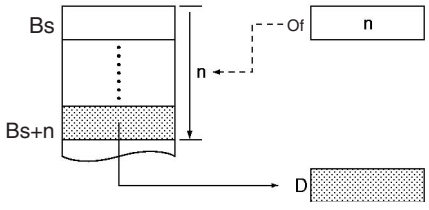
Area	Bs	Of	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---	#0000 to #7999 for Data Collection #8000 to #8999 for LIFO Stack Read #9000 to #9999 for FIFO Stack Read	---
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

Depending on the value of Of, COLLC(567) will operate as a data collection instruction, FIFO stack instruction, or LIFO stack instruction.

Data Collection Operation (Of = 0000 to 7999 BCD)

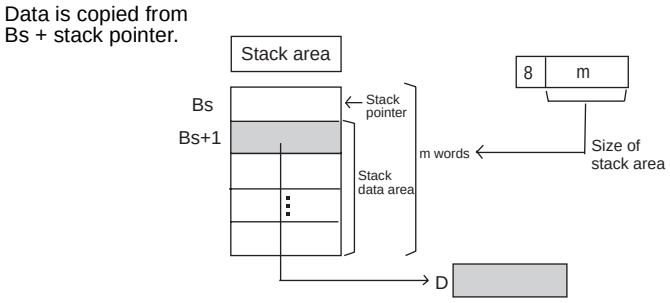
COLLC(567) copies the source word (calculated by adding Of to Bs) to the destination word. The same COLLC(567) instruction can be used to collect data from various source words in the data area by changing the value of Of.



LIFO Stack Read Operation (Of = 8000 to 8999 BCD)

If the leftmost digit of Of is 8, COLLC(567) will operate as a LIFO stack instruction (LIFO stands for Last-In-First-Out). In this case, the rightmost 3 digits of Of specify the size of the stack.

COLLC(567) copies the data most recently recorded in the stack to D. The source word is Bs + the stack pointer (content of Bs). After the data is copied, the stack pointer is decremented by 1.

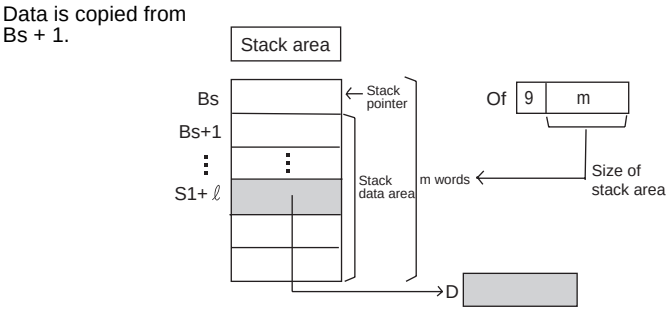


Note Use DISTC(566) to write stack data to the stack area.

FIFO Stack Read Operation (Of = 9000 to 9999 BCD)

If the leftmost digit of Of is 9, COLLC(567) will operate as a FIFO stack instruction (FIFO stands for First-In-First-Out). In this case, the rightmost 3 digits of Of specify the size of the stack.

COLLC(567) copies the data from the oldest word recorded in the stack to D. The source word is Bs + 1. After the data is copied, the stack pointer is decremented by 1.



Note Use DISTC(566) to write stack data to the stack area.

Flags

Name	Label	Operation
Error Flag	ER	ON if the offset data in Of is not BCD. ON if LIFO or FIFO Stack Operation is specified, but the stack pointer data in Bs is not BCD. ON if LIFO or FIFO Stack Operation is specified and the stack pointer indicates a word that exceeds the stack data area. OFF in all other cases.
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.

Note In C-series PLCs, the DATA COLLECT (COLL) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. COLLC(567) will not cause the Error Flag to go ON in these cases.

Precautions

Once DISTC(566) has been executed with Stack Push Operation to allocate a stack area, always specify that same length stack area in the COLLC(567) instructions. Operation will be unreliable if a different stack area size is specified in the COLLC(567) instructions.

Be sure that the offset or stack size specified by Of does not exceed the end of the data area when added to Bs.

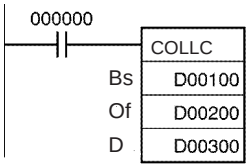
The offset data in Of must be BCD.

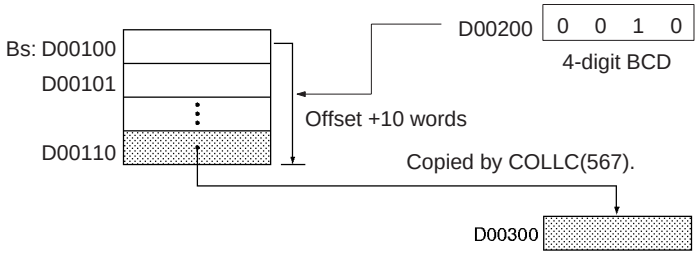
Examples

Data Collection Operation

The leftmost byte of D00200 is 0, so COLLC(567) performs the Data Collection Operation.

When CIO 000000 is ON in the following example, the contents of D00110 (D00100 + 10) will be copied to D00300 if the content of D00200 is 10 (0010 BCD). The contents of other words can be copied to D00300 by changing the offset in D00200.

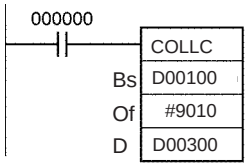


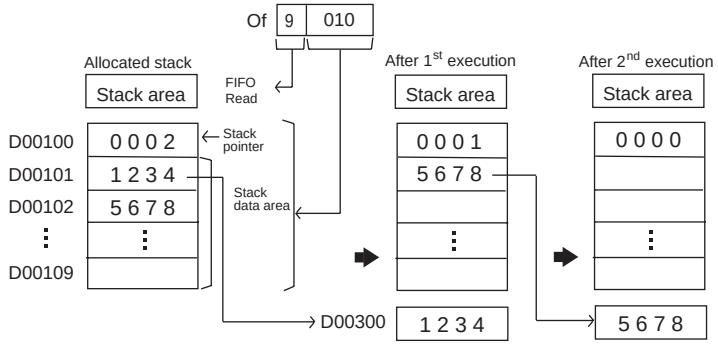


FIFO Stack Operation

The leftmost byte of Of is 9, so COLLC(567) performs the FIFO Stack Operation.

When CIO 000000 is ON in the following example, COLLC(567) allocates a 10 word stack area (since the rightmost 3 digits of Of are #010) between D00100 and D00109. At the same time, the contents of D00101 (Bs +1) are copied to D00300. Finally, the stack pointer is decremented by 1.

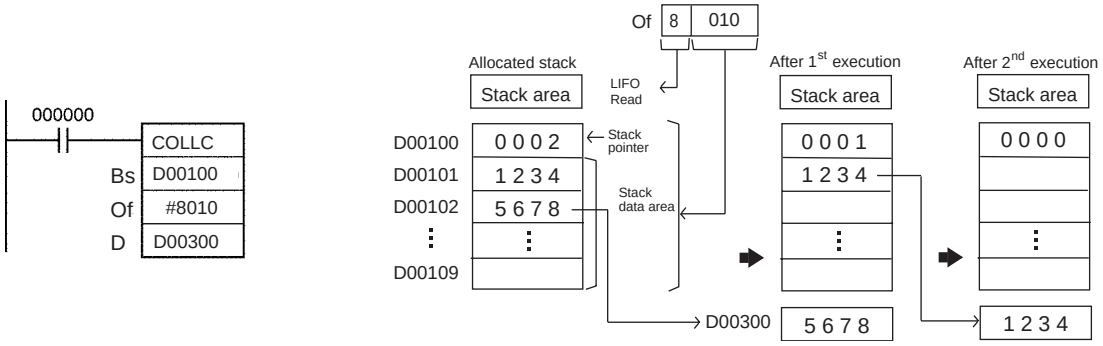




LIFO Stack Operation

The leftmost byte of Of is 8, so COLLC(567) performs the LIFO Stack Operation.

When CIO 000000 is ON in the following example, COLLC(567) allocates a 10 word stack area (since the rightmost 3 digits of Of are #010) between D00100 and D00109. At the same time, the contents of the source word (D00100 + stack pointer) are copied to D00300. Finally, the stack pointer is decremented by 1.

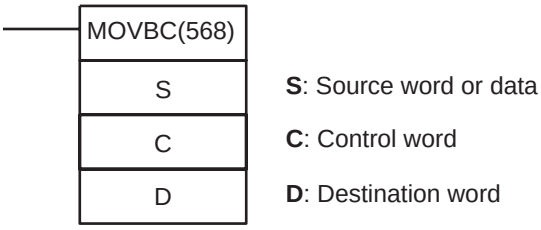


3-35-4 MOVE BIT: MOVBC(568)

Purpose

Transfers the specified bit.

Ladder Symbol



S: Source word or data

C: Control word

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVBC(568)
	Executed Once for Upward Differentiation	@MOVBC(568)
	Executed Once for Downward Differentiation	Not supported
Immediate Refreshing Specification		Not supported

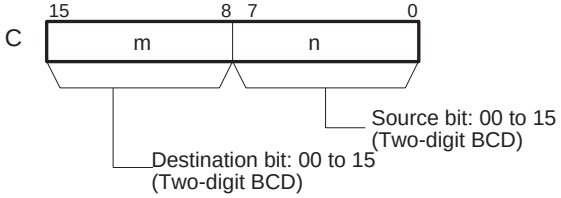
Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word

The rightmost two digits of C indicate which bit of S is the source bit and the leftmost two digits of C indicate which bit of D is the destination bit.



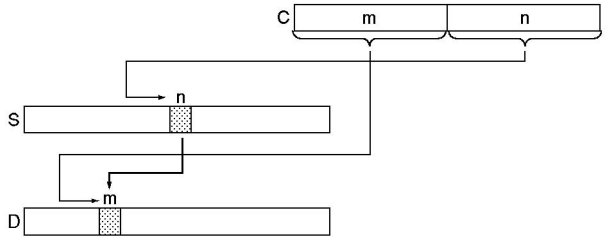
Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		

Area	S	C	D
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0000 to #FFFF (binary)	Specified values only	---
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to, –(– –) IR15		

Description

MOVBC(568) copies the specified bit (n) from S to the specified bit (m) in D. The other bits in the destination word are left unchanged.



Note The same word can be specified for both S and D to copy a bit within a word.

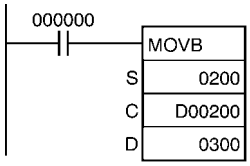
Flags

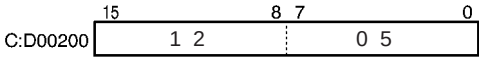
Name	Label	Operation
Error Flag	ER	ON if the rightmost and leftmost two digits of C are not BCD or outside of the specified range of 00 to 15. OFF in all other cases.

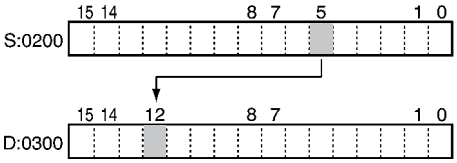
Note In C-series PLCs, the MOVE BIT (MOVB) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. MOVBC(568) will not cause the Error Flag to go ON in these cases.

Examples

When CIO 000000 is ON in the following example, the 5th bit of the source word (CIO 0200) is copied to the 12th bit of the destination word (CIO 0300) in accordance with the control word's value of 1205.





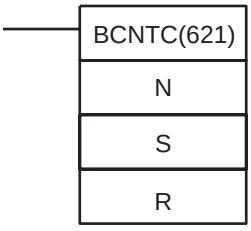


3-35-5 BIT COUNTER: BCNTC(621)

Purpose

Counts the total number of ON bits in the specified word(s).

Ladder Symbol



N: Number of words

S: First source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BCNTC(621)
	Executed Once for Upward Differentiation	@BCNTC(621)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

N: Number of words

The number of words must be 0001 to 9999 (BCD).

S: First source word

S and S+(N–1) must be in the same data area.

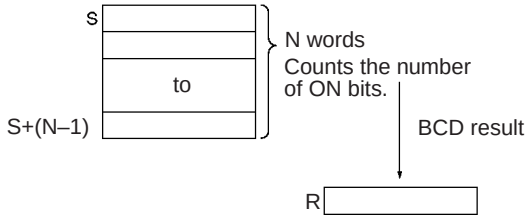
Operand Specifications

Area	N	S	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		

Area	N	S	R
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	#0001 to #9999 (BCD)	---	
Data Registers	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

BCNTC(621) counts the total number of bits that are ON in all words between S and S+(N–1) and places the BCD result in R.



Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the range 0001 to 9999 BCD. ON if result exceeds 9999 BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

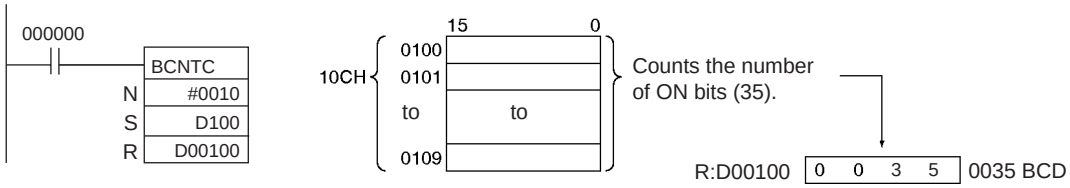
Note In C-series PLCs, the BIT COUNTER (BITC) instruction will cause the Error Flag to go ON if the content of an indirectly addressed DM word (*DM) is not BCD, or the DM area boundary is exceeded. BCNTC(621) will not cause the Error Flag to go ON in these cases.

Precautions

An error will occur if N is not BCD between 0001 and 9999, or the result exceeds 9,999.

Example

When CIO 000000 is ON in the following example, BCNTC(621) counts the total number of ON bits in the 10 words from CIO 0100 through CIO 0109 and writes the result to D00100.

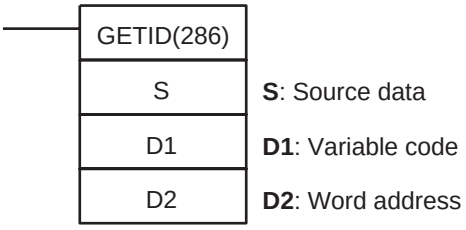


3-35-6 GET VARIABLE ID: GETID(286)

Purpose

Outputs the FINS command variable type (data area) code and word address for the specified variable or address. This instruction is generally used to get the assigned address of a variable in a function block.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	GETID(286)
	Executed Once for Upward Differentiation	@GETID(286)
	Executed Once for Downward Differentiation	Not supported.
Immediate Refreshing Specification		Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: Source data
Specifies the variable or address for which the variable type and word address will be retrieved.

D1: Variable code
Contains the FINS variable type code (data area code) of the source data.

D2: Word address
Contains the word address of the source data in 4-digit hexadecimal.

Operand Specifications

Area	S	D1	D2
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W511		
Holding Bit Area	H000 to H511		
Auxiliary Bit Area	A000 to A959		
Timer Area	T0000 to T4095		
Counter Area	C0000 to C4095		
DM Area	D00000 to D32767		
EM Area without bank	E00000 to E32767		
EM Area with bank	En_00000 to En_32767 (n = 0 to C)		

Area	S	D1	D2
Indirect DM/EM addresses in binary	@ D00000 to @ D32767 @ E00000 to @ E32767 @ En_00000 to @ En_32767 (n = 0 to C)		
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)		
Constants	---		
Data Registers	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to, –(– –)IR15		

Description

GETID(286) retrieves the data area address of the specified source variable or address, outputs the data area code to D1 in 4-digit hexadecimal, and outputs the word address number to D2 in 4-digit hexadecimal.

The following table shows the variable type (data area) codes and corresponding address ranges for the PLC's data areas.

Data area		Data size	Data area code (Output to D1.)	Address (Output to D2.)
CIO Area	CIO	Word	00B0 hex	0000 to 17FF hex (0000 to 6143)
Work Area	W		00B1 hex	0000 to 01FF hex (000 to 511)
Holding Bit Area	H		00B2 hex	0000 to 01FF hex (000 to 511)
DM Area			0082 hex	0000 to 7FFF hex (00000 to 32767)
EM Area (Specific bank)	En_ (n = 0 to C)		00A0 to 00AC hex	0000 to 7FFF hex (00000 to 32767)

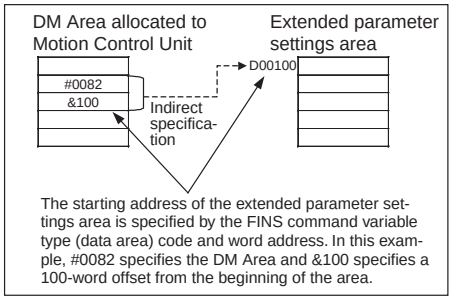
Variables in function blocks are automatically allocated addresses by CX-Programmer Ver. 5.0 and later systems, unless the AT specification is used. For example, if it is necessary to indirectly specify the extended parameter settings of a Special Unit such as a Motion Control Unit and a variable is used at the beginning of the extended parameter settings area, that variable's address must be set. In this case, GETID(286) can be used to retrieve the variable's data area address.

Flags

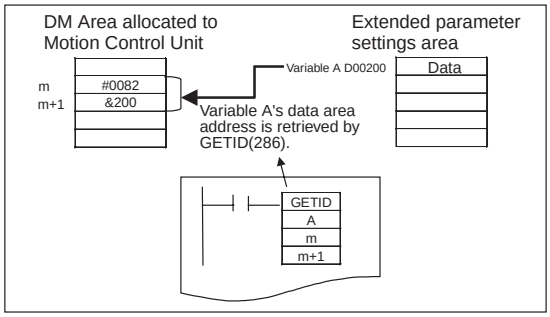
Name	Label	Operation
Error Flag	ER	ON if S is not within the allowed range.

Example

Normal Operation



Using Function Blocks



SECTION 4
Instruction Execution Times and Number of Steps

This section provides instruction execution times and the number of steps for each CS/CJ-series instruction.

4-1 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 PLCs.

The total execution time of instructions within one whole user program is the process time for program execution when calculating the cycle time (See note.).

Note User programs are allocated tasks that can be executed within cyclic tasks and interrupt tasks that satisfy interrupt conditions.

Execution times for most instructions 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 top line for each instruction in the following table shows the minimum time required to process the instruction and the necessary execution conditions, and the bottom line shows the maximum time and execution conditions required to process the instruction.

The execution time can also vary when the execution condition is OFF.

The following table also lists the length of each instruction in the *Length (steps)* column. The number of steps required in the user program area for each of the 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 *4-1 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	CPU4□H	CPU6□	CPU4□
↑ 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	CPU4□H	CPU6□	CPU4□
Approx. 0.1	Approx. 0.2	Approx. 0.1 to 0.3	Approx. 0.2 to 0.4

4-1-1 Sequence Input Instructions

Instruction	Mnemonic	Code	Length (steps)	ON execution time (μs)				Conditions
				CPU6□H	CPU4□H	CPU6□	CPU4□	
LOAD	LD	---	1	0.02	0.04	0.04	0.08	---
	!LD	---	2	+21.14	+21.16	+21.16	+21.16	Increase for CS Series
				+45.1	+45.1	+45.1	+45.1	Increase for C200H
LOAD NOT	LD NOT	---	1	0.02	0.04	0.04	008	---
	!LD NOT	---	2	+21.14	+21.16	+21.16	+21.16	Increase for CS Series
				+45.1	+45.1	+45.1	+45.1	Increase for C200H
AND	AND	---	1	0.02	0.04	0.04	0.08	---
	!AND	---	2	+21.14	+21.16	+21.16	+21.16	Increase for CS Series
				+45.1	+45.1	+45.1	+45.1	Increase for C200H
AND NOT	AND NOT	---	1	0.02	0.04	0.04	0.08	---
	!AND NOT	---	2	+21.14	+21.16	+21.16	+21.16	Increase for CS Series
				+45.1	+45.1	+45.1	+45.1	Increase for C200H
OR	OR	---	1	0.02	0.04	0.04	0.08	---
	!OR	---	2	+21.14	+21.16	+21.16	+21.16	Increase for CS Series
				+45.1	+45.1	+45.1	+45.1	Increase for C200H
OR NOT	OR NOT	---	1	0.02	0.04	0.04	0.08	---
	!OR NOT	---	2	+21.14	+21.16	+21.16	+21.16	Increase for CS Series
				+45.1	+45.1	+45.1	+45.1	Increase for 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	520	1	0.02	0.04	0.04	0.08	---
CONDITION ON	UP	521	3	0.3	0.42	0.46	0.54	---
CONDITION OFF	DOWN	522	4	0.3	0.42	0.46	0.54	---
LOAD BIT TEST	LD TST	350	4	0.14	0.24	0.25	0.37	---
LOAD BIT TEST NOT	LD TSTN	351	4	0.14	0.24	0.25	0.37	---
AND BIT TEST NOT	AND TSTN	351	4	0.14	0.24	0.25	0.37	---
OR BIT TEST	OR TST	350	4	0.14	0.24	0.25	0.37	---
OR BIT TEST NOT	OR TSTN	351	4	0.14	0.24	0.25	0.37	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-2 Sequence Output Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
OUTPUT	OUT	---	1	0.02	0.04	0.17	0.21	---
	!OUT	---	2	+21.37	+21.37	+21.37	+21.37	Increase for CS Series
				+49.3	+49.3	+49.3	+49.3	Increase for C200H
OUTPUT NOT	OUT NOT	---	1	0.02	0.04	0.17	0.21	---
	!OUT NOT	---	2	+21.37	+21.37	+21.37	+21.37	Increase for CS Series
				+49.3	+49.3	+49.3	+49.3	Increase for C200H
KEEP	KEEP	011	1	0.06	0.08	0.25	0.29	---
DIFFERENTIATE UP	DIFU	013	2	0.24	0.40	0.46	0.54	---
DIFFERENTIATE DOWN	DIFD	014	2	0.24	0.40	0.46	0.54	---
SET	SET	---	1	0.02	0.06	0.17	0.21	---
	!SET	---	2	+21.37	+21.37	+21.37	+21.37	Increase for CS Series
				+49.3	+49.3	+49.3	+49.3	Increase for C200H
RESET	RSET	---	1	0.02	0.06	0.17	0.21	Word specified
	!RSET	---	2	+21.37	+21.37	+21.37	+21.37	Increase for CS Series
				+49.3	+49.3	+49.3	+49.3	Increase for C200H
MULTIPLE BIT SET	SETA	530	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	531	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	532	2	0.24	0.34	---	---	---
	!SETB		3	+21.44	+21.54	---	---	---
SINGLE BIT RESET	RSTB	534	2	0.24	0.34	---	---	---
	!RSTB		3	+21.44	+21.54	---	---	---
SINGLE BIT OUTPUT	OUTB	534	2	0.22	0.32	---	---	---
	!OUTB		3	+21.42	+21.52	---	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-3 Sequence Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
END	END	001	1	5.5	6.0	4.0	4.0	---
NO OPERATION	NOP	000	1	0.02	0.04	0.08	0.12	---
INTERLOCK	IL	002	1	0.06	0.06	0.12	0.12	---
INTERLOCK CLEAR	ILC	003	1	0.06	0.06	0.12	0.12	---
MULTI-INTERLOCK DIFFERENTIATION HOLD (See note 2.)	MILH	517	3	6.1	6.5	---	---	During interlock
				7.5	7.9	---	---	Not during interlock and interlock not set
				8.9	9.7	---	---	Not during interlock and interlock set
MULTI-INTERLOCK DIFFERENTIATION RELEASE (See note 2.)	MILR	518	3	6.1	6.5	---	---	During interlock
				7.5	7.9	---	---	Not during interlock and interlock not set
				8.9	9.7	---	---	Not during interlock and interlock set
MULTI-INTERLOCK CLEAR (See note 2.)	MILC	519	2	5.0	5.6	---	---	Interlock not cleared
				5.7	6.2	---	---	Interlock cleared
JUMP	JMP	004	2	0.38	0.48	8.1	8.1	---
JUMP END	JME	005	2	---	---	---	---	---
CONDITIONAL JUMP	CJP	510	2	0.38	0.48	7.4	7.4	When JMP condition is satisfied
CONDITIONAL JUMP NOT	CJPN	511	2	0.38	0.48	8.5	8.5	When JMP condition is satisfied
MULTIPLE JUMP	JMP0	515	1	0.06	0.06	0.12	0.12	---
MULTIPLE JUMP END	JME0	516	1	0.06	0.06	0.12	0.12	---
FOR LOOP	FOR	512	2	0.52	0.54	0.12	0.21	Designating a constant
BREAK LOOP	BREAK	514	1	0.06	0.06	0.12	0.12	---
NEXT LOOP	NEXT	513	1	0.18	0.16	0.17	0.17	When loop is continued
				0.22	0.40	0.12	0.12	When loop is ended

Note 1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.

2. Supported only by CPU Units Ver. 2.0 or later.

4-1-4 Timer and Counter Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
HUNDRED-MS TIMER	TIM	---	3	0.56	0.88	0.37	0.42	---
	TIMX	550	3	0.56	0.88	---	---	---
TEN-MS TIMER	TIMH	015	3	0.88	1.14	0.37	0.42	---
	TIMHX	551	3	0.88	1.14	---	---	---
ONE-MS TIMER	TMHH	540	3	0.86	1.12	0.37	0.42	---
	TMHHX	552	3	0.86	1.12	---	---	---
ACCUMULA-TIVE TIMER	TTIM	087	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 interlock-ing
	TTIMX	555	3	16.1	17.0	---	---	---
				10.9	11.4	---	---	When resetting
				8.5	8.7	---	---	When interlock-ing
LONG TIMER	TIML	542	4	7.6	10.0	12.8	12.8	---
				6.2	6.5	7.8	7.8	When interlock-ing
	TIMLX	553	4	7.6	10.0	---	---	---
				6.2	6.5	---	---	When interlock-ing
MULTI-OUT-PUT TIMER	MTIM	543	4	20.9	23.3	26.0	26.0	---
				5.6	5.8	7.8	7.8	When resetting
	MTIMX	554	4	20.9	23.3	---	---	---
				5.6	5.8	---	---	When resetting
COUNTER	CNT	---	3	0.56	0.88	0.37	0.42	---
	CNTX	546	3	0.56	0.88	---	---	---
REVERSIBLE COUNTER	CNTR	012	3	16.9	19.0	20.9	20.9	---
	CNTRX	548	3	16.9	19.0	---	---	---
RESET TIMER/ COUNTER	CNR	545	3	9.9	10.6	13.9	13.9	When resetting 1 word
				4.16 ms	4.16 ms	5.42 ms	5.42 ms	When resetting 1,000 words
	CNRX	547	3	9.9	10.6	---	---	When resetting 1 word
				4.16 ms	4.16 ms	---	---	When resetting 1,000 words

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-5 Comparison Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
Input Comparison Instructions (unsigned)	LD, AND, OR +=	300	4	0.10	0.16	0.21	0.37	---
	LD, AND, OR + <>	305						
	LD, AND, OR + <	310						
	LD, AND, OR + <=	315						
	LD, AND, OR + >	320						
	LD, AND, OR + >=	325						
Input Comparison Instructions (double, unsigned)	LD, AND, OR +=+L	301	4	0.10	0.16	0.29	0.54	---
	LD, AND, OR +<>+L	306						
	LD, AND, OR +<+L	311						
	LD, AND, OR +<=+L	316						
	LD, AND, OR +>+L	321						
	LD, AND, OR +>=+L	326						
Input Comparison Instructions (signed)	LD, AND, OR +=+S	302	4	0.10	0.16	6.50	6.50	---
	LD, AND, OR +<>+S	307						
	LD, AND, OR +<+S	312						
	LD, AND, OR +<=	317						
	LD, AND, OR +>+S	322						
	LD, AND, OR +>=+S	327						
Input Comparison Instructions (double, signed)	LD, AND, OR +=+SL	303	4	0.10	0.16	6.50	6.50	---
	LD, AND, OR +<>+SL	308						
	LD, AND, OR +<+SL	313						
	LD, AND, OR +<=+SL	318						
	LD, AND, OR +>+SL	323						
	LD, AND, OR +>=+SL	328						

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
Time Comparison Instructions (See note 2.)	LD, AND, OR +DT	341	4	25.1	36.4	---	---	ON and OFF execution times are the same as given at the left.
	LD, AND, OR +<>DT	342	4	25.2	36.4	---	---	
	LD, AND, OR +<DT	343	4	25.2	36.4	---	---	
	LD, AND, OR +<=DT	344	4	25.2	36.4	---	---	
	LD, AND, OR +>DT	345	4	25.1	36.4	---	---	
	LD, AND, OR +>=DT	346	4	25.2	36.4	---	---	
COMPARE	CMP	020	3	0.04	0.04	0.17	0.29	---
	!CMP	020	7	+42.1	+42.1	+42.4	+42.4	Increase for CS Series
				+90.4	+90.4	+90.5	+90.5	Increase for C200H
DOUBLE COMPARE	CMPL	060	3	0.08	0.08	0.25	0.46	---
SIGNED BINARY COMPARE	CPS	114	3	0.08	0.08	6.50	6.50	---
	!CPS	114	7	+35.9	+35.9	+42.4	+42.4	Increase for CS Series
				+84.1	+84.1	+90.5	+90.5	Increase for C200H
DOUBLE SIGNED BINARY COMPARE	CPSL	115	3	0.08	0.08	6.50	6.50	---
TABLE COMPARE	TCMP	085	4	14.0	15.2	21.9	21.9	---
MULTIPLE COMPARE	MCMP	019	4	20.5	22.8	31.2	31.2	---
UNSIGNED BLOCK COMPARE	BCMP	068	4	21.5	23.7	32.6	32.6	---
AREA RANGE COMPARE	ZCP	088	3	5.3	5.4	---	---	---
DOUBLE AREA RANGE COMPARE	ZCPL	116	3	5.5	6.7	---	---	---

Note 1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.

2. Supported only by CPU Units Ver. 2.0 or later.

4-1-6 Data Movement Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
MOVE	MOV	021	3	0.18	0.20	0.25	0.29	---
	!MOV	021	7	+21.38	+21.40	+42.36	+42.36	Increase for CS Series
				+90.52	+90.52	+90.52	+90.52	Increase for C200H
DOUBLE MOVE	MOVL	498	3	0.32	0.34	0.42	0.50	---
MOVE NOT	MVN	022	3	0.18	0.20	0.25	0.29	---
DOUBLE MOVE NOT	MVNL	499	3	0.32	0.34	0.42	0.50	---
MOVE BIT	MOVB	082	4	0.24	0.34	7.5	7.5	---
MOVE DIGIT	MOVD	083	4	0.24	0.34	7.3	7.3	---
MULTIPLE BIT TRANSFER	XFRB	062	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	070	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	071	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	073	3	0.40	0.56	0.5	0.7	---
DOUBLE DATA EXCHANGE	XCGL	562	3	0.76	1.04	0.9	1.3	---
SINGLE WORD DISTRIBUTE	DIST	080	4	5.1	5.4	7.0	7.0	---
DATA COLLECT	COLL	081	4	5.1	5.3	7.1	7.1	---
MOVE TO REGISTER	MOVR	560	3	0.08	0.08	0.42	0.50	---
MOVE TIMER/ COUNTER PV TO REGISTER	MOVRW	561	3	0.42	0.50	0.42	0.50	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-7 Data Shift Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
SHIFT REGISTER	SFT	010	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	084	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	017	4	6.2	6.4	7.7	7.7	Shifting 1 word
				1.22 ms	1.22 ms	2.01 ms	2.01 ms	Shifting 1,000 words
WORD SHIFT	WSFT	016	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	025	2	0.22	0.32	0.29	0.37	---
DOUBLE SHIFT LEFT	ASLL	570	2	0.40	0.56	0.50	0.67	---
ARITHMETIC SHIFT RIGHT	ASR	026	2	0.22	0.32	0.29	0.37	---
DOUBLE SHIFT RIGHT	ASRL	571	2	0.40	0.56	0.50	0.67	---
ROTATE LEFT	ROL	027	2	0.22	0.32	0.29	0.37	---
DOUBLE ROTATE LEFT	ROLL	572	2	0.40	0.56	0.50	0.67	---
ROTATE LEFT WITHOUT CARRY	RLNC	574	2	0.22	0.32	0.29	0.37	---
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	576	2	0.40	0.56	0.50	0.67	---
ROTATE RIGHT	ROR	028	2	0.22	0.32	0.29	0.37	---
DOUBLE ROTATE RIGHT	RORL	573	2	0.40	0.56	0.50	0.67	---
ROTATE RIGHT WITHOUT CARRY	RRNC	575	2	0.22	0.32	0.29	0.37	---
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	577	2	0.40	0.56	0.50	0.67	---
ONE DIGIT SHIFT LEFT	SLD	074	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	075	3	6.9	7.1	8.7	8.7	Shifting 1 word
				760.5	895.5	1.07 ms	1.07 ms	Shifting 1,000 words
SHIFT N-BIT DATA LEFT	NSFL	578	4	7.5	8.3	10.5	10.5	Shifting 1 bit
				40.3	45.4	55.5	55.5	Shifting 1,000 bits

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
SHIFT N-BIT DATA RIGHT	NSFR	579	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	580	3	0.22	0.32	0.29	0.37	---
DOUBLE SHIFT N-BITS LEFT	NSLL	582	3	0.40	0.56	0.50	0.67	---
SHIFT N-BITS RIGHT	NASR	581	3	0.22	0.32	0.29	0.37	---
DOUBLE SHIFT N-BITS RIGHT	NSRL	583	3	0.40	0.56	0.50	0.67	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-8 Increment/Decrement Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
INCREMENT BINARY	++	590	2	0.22	0.32	0.29	0.37	---
DOUBLE INCREMENT BINARY	++L	591	2	0.40	0.56	0.50	0.67	---
DECREMENT BINARY	– –	592	2	0.22	0.32	0.29	0.37	---
DOUBLE DECREMENT BINARY	– –L	593	2	0.40	0.56	0.50	0.67	---
INCREMENT BCD	++B	594	2	6.4	4.5	7.4	7.4	---
DOUBLE INCREMENT BCD	++BL	595	2	5.6	4.9	6.1	6.1	---
DECREMENT BCD	– –B	596	2	6.3	4.6	7.2	7.2	---
DOUBLE DECREMENT BCD	– –BL	597	2	5.3	4.7	7.1	7.1	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-9 Symbol Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
SIGNED BINARY ADD WITHOUT CARRY	+	400	4	0.18	0.20	0.25	0.37	---
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	401	4	0.32	0.34	0.42	0.54	---
SIGNED BINARY ADD WITH CARRY	+C	402	4	0.18	0.20	0.25	0.37	---
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	403	4	0.32	0.34	0.42	0.54	---
BCD ADD WITHOUT CARRY	+B	404	4	8.2	8.4	14.0	14.0	---
DOUBLE BCD ADD WITHOUT CARRY	+BL	405	4	13.3	14.5	19.0	19.0	---
BCD ADD WITH CARRY	+BC	406	4	8.9	9.1	14.5	14.5	---
DOUBLE BCD ADD WITH CARRY	+BCL	407	4	13.8	15.0	19.6	19.6	---
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	410	4	0.18	0.20	0.25	0.37	---
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	411	4	0.32	0.34	0.42	0.54	---
SIGNED BINARY SUBTRACT WITH CARRY	–C	412	4	0.18	0.20	0.25	0.37	---
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	413	4	0.32	0.34	0.42	0.54	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
BCD SUBTRACT WITHOUT CARRY	–B	414	4	8.0	8.2	13.1	13.1	---
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	415	4	12.8	14.0	18.2	18.2	---
BCD SUBTRACT WITH CARRY	–BC	416	4	8.5	8.6	13.8	13.8	---
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	417	4	13.4	14.7	18.8	18.8	---
SIGNED BINARY MULTIPLY	*	420	4	0.38	0.40	0.50	0.58	---
DOUBLE SIGNED BINARY MULTIPLY	*L	421	4	7.23	8.45	11.19	11.19	---
UNSIGNED BINARY MULTIPLY	*U	422	4	0.38	0.40	0.50	0.58	---
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	423	4	7.1	8.3	10.63	10.63	---
BCD MULTIPLY	*B	424	4	9.0	9.2	12.8	12.8	---
DOUBLE BCD MULTIPLY	*BL	425	4	23.0	24.2	35.2	35.2	---
SIGNED BINARY DIVIDE	/	430	4	0.40	0.42	0.75	0.83	---
DOUBLE SIGNED BINARY DIVIDE	/L	431	4	7.2	8.4	9.8	9.8	---
UNSIGNED BINARY DIVIDE	/U	432	4	0.40	0.42	0.75	0.83	---
DOUBLE UNSIGNED BINARY DIVIDE	/UL	433	4	6.9	8.1	9.1	9.1	---
BCD DIVIDE	/B	434	4	8.6	8.8	15.9	15.9	---
DOUBLE BCD DIVIDE	/BL	435	4	17.7	18.9	26.2	26.2	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-10 Conversion Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
BCD TO BINARY	BIN	023	3	0.22	0.24	0.25	0.29	---
DOUBLE BCD TO DOUBLE BINARY	BINL	058	3	6.5	6.8	9.1	9.1	---
BINARY TO BCD	BCD	024	3	0.24	0.26	8.3	8.3	---
DOUBLE BINARY TO DOUBLE BCD	BCDL	059	3	6.7	7.0	9.2	9.2	---
2'S COMPLEMENT	NEG	160	3	0.18	0.20	0.25	0.29	---
DOUBLE 2'S COMPLEMENT	NEGL	161	3	0.32	0.34	0.42	0.5	---
16-BIT TO 32-BIT SIGNED BINARY	SIGN	600	3	0.32	0.34	0.42	0.50	---
DATA DECODER	MLPX	076	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 2 digits (8 to 256)
DATA ENCODER	DMPX	077	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	086	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	162	4	7.1	7.4	10.1	10.1	Converting 1 digit
COLUMN TO LINE	LINE	063	4	19.0	23.1	29.1	29.1	---
LINE TO COLUMN	COLM	064	4	23.2	27.5	37.3	37.3	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
SIGNED BCD TO BINARY	BINS	470	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	472	4	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	471	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	473	4	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 (See note 2.)	GRY	474	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	601	3	13.79	20.24	---	---	
EIGHT-DIGIT NUMBER TO ASCII	STR8	602	3	18.82	27.44	---	---	
SIXTEEN-DIGIT NUMBER TO ASCII	STR16	603	3	30.54	44.41	---	---	

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
ASCII TO FOUR-DIGIT NUMBER	NUM4	604	3	18.46	27.27	---	---	
ASCII TO EIGHT-DIGIT NUMBER	NUM8	605	3	18.46	27.27	---	---	
ASCII TO SIXTEEN-DIGIT NUMBER	NUM16	606	3	52.31	78.25	---	---	

Note 1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.

2. Supported only by CPU Units Ver. 2.0 or later.

4-1-11 Logic Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
LOGICAL AND	ANDW	034	4	0.18	0.20	0.25	0.37	---
DOUBLE LOGICAL AND	ANDL	610	4	0.32	0.34	0.42	0.54	---
LOGICAL OR	ORW	035	4	0.22	0.32	0.25	0.37	---
DOUBLE LOGICAL OR	ORWL	611	4	0.32	0.34	0.42	0.54	---
EXCLUSIVE OR	XORW	036	4	0.22	0.32	0.25	0.37	---
DOUBLE EXCLUSIVE OR	XORL	612	4	0.32	0.34	0.42	0.54	---
EXCLUSIVE NOR	XNRW	037	4	0.22	0.32	0.25	0.37	---
DOUBLE EXCLUSIVE NOR	XNRL	613	4	0.32	0.34	0.42	0.54	---
COMPLEMENT	COM	029	2	0.22	0.32	0.29	0.37	---
DOUBLE COMPLEMENT	COML	614	2	0.40	0.56	0.50	0.67	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-12 Special Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
BINARY ROOT	ROTB	620	3	49.6	50.0	530.7	530.7	---
BCD SQUARE ROOT	ROOT	072	3	13.7	13.9	514.5	514.5	---
ARITHMETIC PROCESS	APR	069	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
FLOATING POINT DIVIDE	FDIV	079	4	116.6	176.6	176.6	176.6	---
BIT COUNTER	BCNT	067	4	0.3	0.38	22.1	22.1	Counting 1 word

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-13 Floating-point Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
FLOATING TO 16-BIT	FIX	450	3	10.6	10.8	14.5	14.5	---
FLOATING TO 32-BIT	FIXL	451	3	10.8	11.0	14.6	14.6	---
16-BIT TO FLOATING	FLT	452	3	8.3	8.5	11.1	11.1	---
32-BIT TO FLOATING	FLTL	453	3	8.3	8.5	10.8	10.8	---
FLOATING-POINT ADD	+F	454	4	8.0	9.2	10.2	10.2	---
FLOATING-POINT SUB-TRACT	–F	455	4	8.0	9.2	10.3	10.3	---
FLOATING-POINT DIVIDE	/F	457	4	8.7	9.9	12.0	12.0	---
FLOATING-POINT MULTI-PLY	*F	456	4	8.0	9.2	10.5	10.5	---
DEGREES TO RADIANS	RAD	458	3	10.1	10.2	14.9	14.9	---
RADIANS TO DEGREES	DEG	459	3	9.9	10.1	14.8	14.8	---
SINE	SIN	460	3	42.0	42.2	61.1	61.1	---
COSINE	COS	461	3	31.5	31.8	44.1	44.1	---
TANGENT	TAN	462	3	16.3	16.6	22.6	22.6	---
ARC SINE	ASIN	463	3	17.6	17.9	24.1	24.1	---
ARC COSINE	ACOS	464	3	20.4	20.7	28.0	28.0	---
ARC TAN-GENT	ATAN	465	3	16.1	16.4	16.4	16.4	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
SQUARE ROOT	SQRT	466	3	19.0	19.3	28.1	28.1	---
EXPONENT	EXP	467	3	65.9	66.2	96.7	96.7	---
LOGARITHM	LOG	468	3	12.8	13.1	17.4	17.4	---
EXPONENTIAL POWER	PWR	840	4	125.4	126.0	181.7	181.7	---
Floating Symbol Comparison	LD, AND, OR +=F	329	3	6.6	8.3	---	---	---
	LD, AND, OR +<>F	330						
	LD, AND, OR +<F	331						
	LD, AND, OR +<=F	332						
	LD, AND, OR +>F	333						
	LD, AND, OR +>=F	334						
FLOATING-POINT TO ASCII	FSTR	448	4	48.5	48.9	---	---	---
ASCII TO FLOATING-POINT	FVAL	449	3	21.1	21.3	---	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-14 Double-precision Floating-point Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
DOUBLE SYMBOL COMPARISON	LD, AND, OR +=D	335	3	8.5	10.3	---	---	---
	LD, AND, OR +<>D	336						
	LD, AND, OR +<D	337						
	LD, AND, OR +<=D	338						
	LD, AND, OR +>D	339						
	LD, AND, OR +>=D	340						
DOUBLE FLOATING TO 16-BIT BINARY	FIXD	841	3	11.7	12.1	---	---	---
DOUBLE FLOATING TO 32-BIT BINARY	FIXLD	842	3	11.6	12.1	---	---	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
16-BIT BINARY TO DOUBLE FLOATING	DBL	843	3	9.9	10.0	---	---	---
32-BIT BINARY TO DOUBLE FLOATING	DBLL	844	3	9.8	10.0	---	---	---
DOUBLE FLOATING-POINT ADD	+D	845	4	11.2	11.9	---	---	---
DOUBLE FLOATING-POINT SUBTRACT	–D	846	4	11.2	11.9	---	---	---
DOUBLE FLOATING-POINT MULTIPLY	*D	847	4	12.0	12.7	---	---	---
DOUBLE FLOATING-POINT DIVIDE	/D	848	4	23.5	24.2	---	---	---
DOUBLE DEGREES TO RADIANS	RADD	849	3	27.4	27.8	---	---	---
DOUBLE RADIANS TO DEGREES	DEGD	850	3	11.2	11.9	---	---	---
DOUBLE SINE	SIND	851	3	45.4	45.8	---	---	---
DOUBLE COSINE	COSD	852	3	43.0	43.4	---	---	---
DOUBLE TANGENT	TAND	853	3	20.1	20.5	---	---	---
DOUBLE ARC SINE	ASIND	854	3	21.5	21.9	---	---	---
DOUBLE ARC COSINE	ACOSD	855	3	24.7	25.1	---	---	---
DOUBLE ARC TANGENT	ATAND	856	3	19.3	19.7	---	---	---
DOUBLE SQUARE ROOT	SQRTD	857	3	47.4	47.9	---	---	---
DOUBLE EXPONENT	EXPD	858	3	121.0	121.4	---	---	---
DOUBLE LOGARITHM	LOGD	859	3	16.0	16.4	---	---	---
DOUBLE EXPONENTIAL POWER	PWRD	860	4	223.9	224.2	---	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-15 Table Data Processing Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
SET STACK	SSET	630	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	632	3	6.5	8.6	9.1	9.1	---
FIRST IN FIRST OUT	FIFO	633	3	6.9	8.9	10.6	10.6	Designating 5 words in stack area
				352.6	434.3	1.13 ms	1.13 ms	Designating 1,000 words in stack area
LAST IN FIRST OUT	LIFO	634	3	7.0	9.0	9.9	9.9	---
DIMENSION RECORD TABLE	DIM	631	5	15.2	21.6	142.1	142.1	---
SET RECORD LOCATION	SETR	635	4	5.4	5.9	7.0	7.0	---
GET RECORD NUMBER	GETR	636	4	7.8	8.4	11.0	11.0	---
DATA SEARCH	SRCH	181	4	15.5	19.5	19.5	19.5	Searching for 1 word
				2.42 ms	3.34 ms	3.34 ms	3.34 ms	Searching for 1,000 words
SWAP BYTES	SWAP	637	3	12.2	13.6	13.6	13.6	Swapping 1 word
				1.94 ms	2.82 ms	2.82 ms	2.82 ms	Swapping 1,000 words
FIND MAXIMUM	MAX	182	4	19.2	24.9	24.9	24.9	Searching for 1 word
				2.39 ms	3.36 ms	3.36 ms	3.36 ms	Searching for 1,000 words
FIND MINIMUM	MIN	183	4	19.2	25.3	25.3	25.3	Searching for 1 word
				2.39 ms	3.33 ms	3.33 ms	3.33 ms	Searching for 1,000 words
SUM	SUM	184	4	28.2	38.5	38.5	38.3	Adding 1 word
				1.42 ms	1.95 ms	1.95 ms	1.95 ms	Adding 1,000 words
FRAME CHECKSUM	FCS	180	4	20.0	28.3	28.3	28.3	For 1-word table length
				1.65 ms	2.48 ms	2.48 ms	2.48 ms	For 1,000-word table length
STACK SIZE READ	SNUM	638	3	6.0	6.3	---	---	---
STACK DATA READ	SREAD	639	4	8.0	8.4	---	---	---
STACK DATA OVERWRITE	SWRIT	640	4	7.2	7.6	---	---	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
STACK DATA INSERT	SINS	641	4	7.8	9.9	---	---	---
				354.0	434.8	---	---	For 1,000-word table
STACK DATA DELETE	SDEL	642	4	8.6	10.6	---	---	---
				354.0	436.0	---	---	For 1,000-word table

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-16 Data Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
PID CONTROL	PID	190	4	436.2	678.2	678.2	678.2	Initial execution
				332.3	474.9	474.9	474.9	Sampling
				97.3	141.3	141.3	141.3	Not sampling
LIMIT CONTROL	LMT	680	4	16.1	22.1	22.1	22.1	---
DEAD BAND CONTROL	BAND	681	4	17.0	22.5	22.5	22.5	---
DEAD ZONE CONTROL	ZONE	682	4	15.4	20.5	20.5	20.5	---
TIME-PROPORTIONAL OUTPUT (See note 2.)	TPO	685	4	10.4	14.8	---	---	OFF execution time
				54.5	82.0	---	---	ON execution time with duty designation or displayed output limit
				61.0	91.9	---	---	ON execution time with manipulated variable designation and output limit enabled
SCALING	SCL	194	4	37.1	53.0	56.8	56.8	---
SCALING 2	SCL2	486	4	28.5	40.2	50.7	50.7	---
SCALING 3	SCL3	487	4	33.4	47.0	57.7	57.7	---
AVERAGE	AVG	195	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
PID CONTROL WITH AUTOTUNING	PIDAT	191	4	446.3	712.5	---	---	Initial execution
				339.4	533.9	---	---	Sampling
				100.7	147.1	---	---	Not sampling
				189.2	281.6	---	---	Initial execution of autotuning
				535.2	709.8	---	---	Autotuning when sampling

Note 1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.

2. Supported only by CPU Units Ver. 2.0 or later.

4-1-17 Subroutine Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
SUBROUTINE CALL	SBS	091	2	1.26	1.96	17.0	17.0	---
SUBROUTINE ENTRY	SBN	092	2	---	---	---	---	---
SUBROUTINE RETURN	RET	093	1	0.86	1.60	20.60	20.60	---
MACRO	MCRO	099	4	23.3	23.3	23.3	23.3	---
GLOBAL SUBROUTINE CALL	GSBN	751	2	---	---	---	---	---
GLOBAL SUBROUTINE ENTRY	GRET	752	1	1.26	1.96	---	---	---
GLOBAL SUBROUTINE RETURN	GSBS	750	2	0.86	1.60	---	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-18 Interrupt Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
SET INTERRUPT MASK	MSKS	690	3	25.6	38.4	39.5	39.5	---
READ INTERRUPT MASK	MSKR	692	3	11.9	11.9	11.9	11.9	---
CLEAR INTERRUPT	CLI	691	3	27.4	41.3	41.3	41.3	---
DISABLE INTERRUPTS	DI	693	1	15.0	16.8	16.8	16.8	---
ENABLE INTERRUPTS	EI	694	1	19.5	21.8	21.8	21.8	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-19 Step Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
STEP DEFINE	STEP	008	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	009	2	6.6	7.3	10.0	10.0	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-20 Basic I/O Unit Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
I/O REFRESH	IORF	097	3	58.5	63.2	81.7	81.7	1-word refresh (IN) for C200H Basic I/O Units
				62.6	67.0	86.7	86.7	1-word refresh (OUT) for C200H Basic I/O Units
				15.5	16.4	23.5	23.5	1-word refresh (IN) for CS-series Basic I/O Units
				17.20	18.40	25.6	25.6	1-word refresh (OUT) for CS-series Basic I/O Units
				303.3	343.9	357.1	357.1	10-word refresh (IN) for C200H Basic I/O Units
				348.2	376.6	407.5	407.5	10-word refresh (OUT) for C200H Basic I/O Units
				319.9	320.7	377.5	377.6	60-word refresh (IN) for CS-series Basic I/O Units
				358.00	354.40	460.1	460.1	60-word refresh (OUT) for CS-series Basic I/O Units
CPU BUS I/O REFRESH	DLNK	226	4	287.8	315.5	---	---	Allocated 1 word
7-SEGMENT DECODER	SDEC	078	4	6.5	6.9	14.1	14.1	---
DIGITAL SWITCH INPUT (See note 2.)	DSW	210	6	50.7	73.5	---	---	4 digits, data input value: 0
				51.5	73.4	---	---	4 digits, data input value: F
				51.3	73.5	---	---	8 digits, data input value: 0
				50.7	73.4	---	---	8 digits, data input value: F
TEN KEY INPUT (See note 2.)	TKY	211	4	9.7	13.2	---	---	Data input value: 0
				10.7	14.8	---	---	Data input value: F
HEXADECIMAL KEY INPUT (See note 2.)	HKY	212	5	50.3	70.9	---	---	Data input value: 0
				50.1	71.2	---	---	Data input value: F
MATRIX INPUT (See note 2.)	MTR	213	5	47.8	68.1	---	---	Data input value: 0
				48.0	68.0	---	---	Data input value: F
7-SEGMENT DISPLAY OUTPUT (See note 2.)	7SEG	214	5	58.1	83.3	---	---	4 digits
				63.3	90.3	---	---	8 digits

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
INTELLIGENT I/O READ	IORD	222	4	Read/write times depend on the Special I/O Unit for which the instruction is being executed.				---
INTELLIGENT I/O WRITE	IOWR	223	4					---

Note 1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.

2. Supported only by CPU Units Ver. 2.0 or later.

4-1-21 Serial Communications Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
PROTOCOL MACRO	PMCR	260	5	100.1	142.1	276.8	276.8	Sending 0 words, receiving 0 words
				134.2	189.6	305.9	305.9	Sending 1 word, receiving 1 word
TRANSMIT	TXD	236	4	68.5	98.8	98.8	98.8	Sending 1 byte
				734.3	1.10 ms	1.10 ms	1.10 ms	Sending 256 bytes
RECEIVE	RXD	235	4	89.6	131.1	131.1	131.1	Storing 1 byte
				724.2	1.11 ms	1.11 ms	1.11 ms	Storing 256 bytes
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	256	4	131.5	202.4	---	---	Sending 1 byte
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	255	4	131	200.8	---	---	Storing 1 byte
CHANGE SERIAL PORT SETUP	STUP	237	3	341.2	400.0	440.4	440.4	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-22 Network Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
NETWORK SEND	SEND	090	4	84.4	123.9	123.9	123.9	---
NETWORK RECEIVE	RECV	098	4	85.4	124.7	124.7	124.7	---
DELIVER COMMAND	CMND	490	4	106.8	136.8	136.8	136.8	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
EXPLICIT MESSAGE SEND (See note 2.)	EXPLT	720	4	127.6	190.0	---	---	---
EXPLICIT GET ATTRIBUTE (See note 2.)	EGATR	721	4	123.9	185.0	---	---	---
EXPLICIT SET ATTRIBUTE (See note 2.)	ESATR	722	3	110.0	164.4	---	---	---
EXPLICIT WORD READ (See note 2.)	ECHRD	723	4	106.8	158.9	---	---	---
EXPLICIT WORD WRITE (See note 2.)	ECHWR	724	4	106.0	158.3	---	---	---

Note 1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.

2. Supported only by CPU Units Ver. 2.0 or later.

4-1-23 File Memory Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
READ DATA FILE	FREAD	700	5	391.4	632.4	684.1	684.1	2-character directory + file name in binary
				836.1	1.33 ms	1.35 ms	1.35 ms	73-character directory + file name in binary
WRITE DATA FILE	FWRIT	701	5	387.8	627.0	684.7	684.7	2-character directory + file name in binary
				833.3	1.32 ms	1.36 ms	1.36 ms	73-character directory + file name in binary
WRITE TEXT FILE	TWRIT	704	5	390.1	619.1	---	---	

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-24 Display Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
DISPLAY MESSAGE	MSG	046	3	10.1	14.2	14.3	14.3	Displaying message
				8.4	11.3	11.3	11.3	Deleting displayed message

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-25 Clock Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
CALENDAR ADD	CADD	730	4	38.3	201.9	209.5	209.5	---
CALENDAR SUBTRACT	CSUB	731	4	38.6	170.4	184.1	184.1	---
HOURS TO SECONDS	SEC	065	3	21.4	29.3	35.8	35.8	---
SECONDS TO HOURS	HMS	066	3	22.2	30.9	42.1	42.1	---
CLOCK ADJUSTMENT	DATE	735	2	60.5	87.4	95.9	95.9	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-26 Debugging Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
Trace Memory Sampling	TRSM	045	1	80.4	120.0	120.0	120.0	Sampling 1 bit and 0 words
				848.1	1.06 ms	1.06 ms	1.06 ms	Sampling 31 bits and 6 words

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-27 Failure Diagnosis Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
FAILURE ALARM	FAL	006	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	007	3	---	---	---	---	---
FAILURE POINT DETECTION	FPD	269	4	140.9	202.3	202.3	202.3	When executed
				163.4	217.6	217.6	217.6	First time
				185.2	268.9	268.9	268.9	When executed
				207.5	283.6	283.6	283.6	First time

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-28 Other Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
SET CARRY	STC	040	1	0.06	0.06	0.12	0.12	---
CLEAR CARRY	CLC	041	1	0.06	0.06	0.12	0.12	---
SELECT EM BANK	EMBC	281	2	14.0	15.1	15.1	15.1	---
EXTEND MAXIMUM CYCLE TIME	WDT	094	2	15.0	19.7	19.7	19.7	---
SAVE CONDITION FLAGS	CCS	282	1	8.6	12.5	---	---	---
LOAD CONDITION FLAGS	CCL	283	1	9.8	13.9	---	---	---
CONVERT ADDRESS FROM CV	FRMCV	284	3	13.6	19.9	---	---	---
CONVERT ADDRESS TO CV	TOCV	285	3	11.9	17.2	---	---	---
DISABLE PERIPHERAL SERVICING	IOSP	287	---	13.9	19.8	---	---	---
ENABLE PERIPHERAL SERVICING	IORS	288	---	63.6	92.3	---	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-29 Block Programming Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
BLOCK PROGRAM BEGIN	BPRG	096	2	12.1	13.0	13.0	13.0	---
BLOCK PROGRAM END	BEND	801	1	9.6	12.3	13.1	13.1	---
BLOCK PROGRAM PAUSE	BPPS	811	2	10.6	12.3	14.9	14.9	---
BLOCK PROGRAM RESTART	BPRS	812	2	5.1	5.6	8.3	8.3	---
CONDITIONAL BLOCK EXIT	(Execution condition) EXIT	806	1	10.0	11.3	12.9	12.9	EXIT condition satisfied
				4.0	4.9	7.3	7.3	EXIT condition not satisfied

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
CONDITIONAL BLOCK EXIT	EXIT (bit address)	806	2	6.8	13.5	16.3	16.3	EXIT condition satisfied
				4.7	7.2	10.7	10.7	EXIT condition not satisfied
CONDITIONAL BLOCK EXIT (NOT)	EXIT NOT (bit address)	806	2	12.4	14.0	16.8	16.8	EXIT condition satisfied
				7.1	7.6	11.2	11.2	EXIT condition not satisfied
Branching	IF (execution condition)	802	1	4.6	4.8	7.2	7.2	IF true
				6.7	7.3	10.9	10.9	IF false
Branching	IF (relay number)	802	2	6.8	7.2	10.4	10.4	IF true
				9.0	9.6	14.2	14.2	IF false
Branching (NOT)	IF NOT (relay number)	802	2	7.1	7.6	10.9	10.9	IF true
				9.2	10.1	14.7	14.7	IF false
Branching	ELSE	803	1	6.2	6.7	9.9	9.9	IF true
				6.8	7.7	11.2	11.2	IF false
Branching	IEND	804	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)	805	1	12.6	13.7	16.7	16.7	WAIT condition satisfied
				3.9	4.1	6.3	6.3	WAIT condition not satisfied
ONE CYCLE AND WAIT	WAIT (relay number)	805	2	12.0	13.4	16.5	16.5	WAIT condition satisfied
				6.1	6.5	9.6	9.6	WAIT condition not satisfied
ONE CYCLE AND WAIT (NOT)	WAIT NOT (relay number)	805	2	12.2	13.8	17.0	17.0	WAIT condition satisfied
				6.4	6.9	10.1	10.1	WAIT condition not satisfied
COUNTER WAIT	CNTW	814	4	17.9	22.6	27.4	27.4	Default setting
				19.1	23.9	28.7	28.7	Normal execution
	CNTWX	818	4	17.9	22.6	---	---	Default setting
				19.1	23.9	---	---	Normal execution
TEN-MS TIMER WAIT	TMHW	815	3	25.8	27.9	34.1	34.1	Default setting
				20.6	22.7	28.9	28.9	Normal execution
	TMHWX	817	3	25.8	27.9	---	---	Default setting
				20.6	22.7	---	---	Normal execution
				9.3	10.8	---	---	LEND condition not satisfied
Loop Control	LOOP	809	1	7.9	9.1	12.3	12.3	---
Loop Control	LEND (execution condition)	810	1	7.7	8.4	10.9	10.9	LEND condition satisfied
				6.8	8.0	9.8	9.8	LEND condition not satisfied

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
Loop Control	LEND (relay number)	810	2	9.9	10.7	14.4	14.4	LEND condition satisfied
				8.9	10.3	13.0	13.0	LEND condition not satisfied
Loop Control	LEND NOT (relay number)	810	2	10.2	11.2	14.8	14.8	LEND condition satisfied
				9.3	10.8	13.5	13.5	LEND condition not satisfied
HUNDRED-MS TIMER WAIT	TIMW	813	3	22.3	25.2	33.1	33.1	Default setting
				24.9	27.8	35.7	35.7	Normal execution
	TIMWX	816	3	22.3	25.2	---	---	Default setting
				24.9	27.8	---	---	Normal execution

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-30 Text String Processing Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
MOV STRING	MOV$	664	3	45.6	66.0	84.3	84.3	Transferring 1 character
CONCATENATE STRING	+$	656	4	86.5	126.0	167.8	167.8	1 character + 1 character
GET STRING LEFT	LEFT$	652	4	53.0	77.4	94.3	94.3	Retrieving 1 character from 2 characters
GET STRING RIGHT	RGHT$	653	4	52.2	76.3	94.2	94.2	Retrieving 1 character from 2 characters
GET STRING MIDDLE	MID$	654	5	56.5	84.6	230.2	230.2	Retrieving 1 character from 3 characters
FIND IN STRING	FIND$	660	4	51.4	77.5	94.1	94.1	Searching for 1 character from 2 characters
STRING LENGTH	LEN$	650	3	19.8	28.9	33.4	33.4	Detecting 1 character
REPLACE IN STRING	RPLC$	661	6	175.1	258.7	479.5	479.5	Replacing the first of 2 characters with 1 character
DELETE STRING	DEL$	658	5	63.4	94.2	244.6	244.6	Deleting the leading character of 2 characters
EXCHANGE STRING	XCHG$	665	3	60.6	87.2	99.0	99.0	Exchanging 1 character with 1 character
CLEAR STRING	CLR$	666	2	23.8	36.0	37.8	37.8	Clearing 1 character

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
INSERT INTO STRING	INS$	657	5	136.5	200.6	428.9	428.9	Inserting 1 character after the first of 2 characters
String Comparison Instructions	LD, AND, OR +=$	670	4	48.5	69.8	86.2	86.2	Comparing 1 character with 1 character
	LD, AND, OR +<>$	671						
	LD, AND, OR +<$	672						
	LD, AND, OR +>$	674						
	LD, AND, OR +>=$	675						

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-1-31 Task Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
TASK ON	TKON	820	2	19.5	26.3	26.3	26.3	---
TASK OFF	TKOF	821	2	13.3	19.0	26.3	26.3	---

4-1-32 Model Conversion Instructions (CPU Unit Ver. 3.0 or later only)

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
BLOCK TRANSFER	XFERC	565	4	6.4	6.5	---	---	Transferring 1 word
				481.6	791.6	---	---	Transferring 1,000 words
SINGLE WORD DISTRIBUTE	DISTC	566	4	3.4	3.5	---	---	Data distribute
				5.9	7.3	---	---	Stack operation
DATA COLLECT	COLLC	567	4	3.5	3.85	---	---	Data distribute
				8	9.1	---	---	Stack operation
				8.3	9.6	---	---	Stack operation 1 word FIFO Read
				2,052.3	2,097.5	---	---	Stack operation 1,000 word FIFO Read
MOVE BIT	MOVBC	568	4	4.5	4.88	---	---	---
BIT COUNTER	BCNTC	621	4	4.9	5	---	---	Counting 1 word
				1,252.4	1284.4	---	---	Counting 1,000 words

4-1-33 Special Function Block Instructions (CPU Unit Ver. 3.0 or Later Only)

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)				Conditions
				CPU-6□H	CPU-4□H	CPU-6□	CPU-4□	
GET VARIABLE ID	GETID	286	4	14	22.2	---	---	---

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 CV-series PLC would be 4 words per instruction and that of the CS-series PLC would be 7 (4 + 3) steps.

4-2 CJ-series Instruction Execution Times and Number of Steps

The following table lists the execution times for all instructions that are available for CJ PLCs.

The total execution time of instructions within one whole user program is the process time for program execution when calculating the cycle time (See note.).

Note User programs are allocated tasks that can be executed within cyclic tasks and interrupt tasks that satisfy interrupt conditions.

Execution times for most instructions differ depending on the CPU Unit used (CJ1H-CPU6□H-R, CJ1H-CPU6□H, CJ1H-CPU4□H, CJ1M-CPU□□ andCJ1G-CPU4□) and the conditions when the instruction is executed. The top line for each instruction in the following table shows the minimum time required to process the instruction and the necessary execution conditions, and the bottom line shows the maximum time and execution conditions required to process the instruction.

The execution time can also vary when the execution condition is OFF.

The following table also lists the length of each instruction in the *Length (steps)* column. The number of steps required in the user program area for each of the CJ-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 CJ-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 CJ-series instructions, and inaccuracies will occur if the capacity of a user program for another PLC is converted for a CJ-series PLC based on the assumption that 1 word is 1 step. Refer to the information at the end of *4-1 CS-series Instruction Execution Times and Number of Steps* for guidelines on converting program capacities from previous OMRON PLCs.
2. Most instructions are supported in differentiated form (indicated with ↑, ↓, @, and %). Specifying differentiation will increase the execution times by the following amounts.

Symbol	CJ1-H			CJ1M	CJ1
	CPU6□H-R	CPU6□H	CPU4□H	CPU□□	CPU4□
↑ or ↓	+0.24 μs	+0.24 μs	+0.32 μs	+0.5 μs	+0.45 μs
@ or %	+0.24 μs	+0.24 μs	+0.32 μs	+0.5 μs	+0.33 μs

3. Use the following times as guidelines when instructions are not executed.

CJ1-H			CJ1M	CJ1
CPU6□H-R	CPU6□H	CPU4□H	CPU□□	CPU4□
Approx. 0.1 μs	Approx. 0.1 μs	Approx. 0.2 μs	Approx. 0.2 to 0.5 μs	Approx. 0.2 to 0.4 μs

4-2-1 Sequence Input Instructions

Instruction	Mnemonic	Code	Length (steps)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
LOAD	LD	---	1	0.016	0.02	0.04	0.08	0.10	0.10	---
	!LD	---	2	+21.14	+21.14	+21.16	+21.16	+24.10	+28.07	Increase for immediate refresh
LOAD NOT	LD NOT	---	1	0.016	0.02	0.04	008	0.10	0.10	---
	!LD NOT	---	2	+21.14	+21.14	+21.16	+21.16	+24.10	+28.07	Increase for immediate refresh
AND	AND	---	1	0.016	0.02	0.04	0.08	0.10	0.10	---
	!AND	---	2	+21.14	+21.14	+21.16	+21.16	+24.10	+28.07	Increase for immediate refresh

Instruction	Mnemonic	Code	Length (steps)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
AND NOT	AND NOT	---	1	0.016	0.02	0.04	0.08	0.10	0.10	---
	!AND NOT	---	2	+21.14	+21.14	+21.16	+21.16	+24.10	+28.07	Increase for immediate refresh
OR	OR	---	1	0.016	0.02	0.04	0.08	0.10	0.10	---
	!OR	---	2	+21.14	+21.14	+21.16	+21.16	+24.10	+28.07	Increase for immediate refresh
OR NOT	OR NOT	---	1	0.016	0.02	0.04	0.08	0.10	0.10	---
	!OR NOT	---	2	+21.14	+21.14	+21.16	+21.16	+24.10	+28.07	Increase for immediate refresh
AND LOAD	AND LD	---	1	0.016	0.02	0.04	0.08	0.05	0.05	---
OR LOAD	OR LD	---	1	0.016	0.02	0.04	0.08	0.05	0.05	---
NOT	NOT	520	1	0.016	0.02	0.04	0.08	0.05	0.05	---
CONDITION ON	UP	521	3	0.24	0.3	0.42	0.54	0.50	0.50	---
CONDITION OFF	DOWN	522	4	0.24	0.3	0.42	0.54	0.50	0.50	---
LOAD BIT TEST	LD TST	350	4	0.11	0.14	0.24	0.37	0.35	0.35	---
LOAD BIT TEST NOT	LD TSTN	351	4	0.11	0.14	0.24	0.37	0.35	0.35	---
AND BIT TEST NOT	AND TSTN	351	4	0.11	0.14	0.24	0.37	0.35	0.35	---
OR BIT TEST	OR TST	350	4	0.11	0.14	0.24	0.37	0.35	0.35	---
OR BIT TEST NOT	OR TSTN	351	4	0.11	0.14	0.24	0.37	0.35	0.35	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table

4-2-2 Sequence Output Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
OUTPUT	OUT	---	1	0.016	0.02	0.04	0.21	0.35	0.35	---
	!OUT	---	2	+21.37	+21.37	+21.37	+21.37	+23.07	+28.60	Increase for immediate refresh
OUTPUT NOT	OUT NOT	---	1	0.016	0.02	0.04	0.21	0.35	0.35	---
	!OUT NOT	---	2	+21.37	+21.37	+21.37	+21.37	+23.07	+28.60	Increase for immediate refresh
KEEP	KEEP	11	1	0.048	0.06	0.08	0.29	0.40	0.40	---
DIFFERENTIATE UP	DIFU	13	2	0.21	0.24	0.40	0.54	0.50	0.50	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
DIFFERENTIATE DOWN	DIFD	14	2	0.21	0.24	0.40	0.54	0.50	0.50	---
SET	SET	---	1	0.016	0.02	0.06	0.21	0.30	0.30	---
	!SET	---	2	+21.37	+21.37	+21.37	+21.37	+23.17	+28.60	Increase for immediate refresh
RESET	RSET	---	1	0.016	0.02	0.06	0.21	0.30	0.30	Word specified
	!RSET	---	2	+21.37	+21.37	+21.37	+21.37	+23.17	+28.60	Increase for immediate refresh
MULTIPLE BIT SET	SETA	530	4	5.8	5.8	6.1	7.8	11.8	11.8	With 1-bit set
				25.7	25.7	27.2	38.8	64.1	64.1	With 1,000-bit set
MULTIPLE BIT RESET	RSTA	531	4	5.7	5.7	6.1	7.8	11.8	11.8	With 1-bit reset
				25.8	25.8	27.1	38.8	64.0	64.0	With 1,000-bit reset
SINGLE BIT SET	SETB	532	2	0.19	0.24	0.34	---	0.5	0.5	---
	!SETB		3	+21.44	+21.44	+21.54	---	+23.31	+23.31	---
SINGLE BIT RESET	RSTB	533	2	0.19	0.24	0.34	---	0.5	0.5	---
	!RSTB		3	+21.44	+21.44	+21.54	---	+23.31	+23.31	---
SINGLE BIT OUTPUT	OUTB	534	2	0.19	0.22	0.32	---	0.45	0.45	---
	!OUTB		3	+21.42	+21.42	+21.52	---	+23.22	+23.22	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-3 Sequence Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note 1.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
END	END	001	1	5.5	5.5	6.0	4.0	7.9	7.9	---
NO OPERATION	NOP	000	1	0.016	0.02	0.04	0.12	0.05	0.05	---
INTERLOCK	IL	002	1	0.048	0.06	0.06	0.12	0.15	0.15	---
INTERLOCK CLEAR	ILC	003	1	0.048	0.06	0.06	0.12	0.15	0.15	---
MULTI-INTERLOCK DIFFERENTIATION HOLD (See note 2.)	MILH	517	3	6.1	6.1	6.5	---	10.3	11.7	During interlock
				7.5	7.5	7.9	---	13.3	14.6	Not during interlock and interlock not set
				8.9	8.9	9.7	---	16.6	18.3	Not during interlock and interlock set
MULTI-INTERLOCK DIFFERENTIATION RELEASE (See note 2.)	MILR	518	3	6.1	6.1	6.5	---	10.3	11.7	During interlock
				7.5	7.5	7.9	---	13.3	14.6	Not during interlock and interlock not set
				8.9	8.9	9.7	---	16.6	18.3	Not during interlock and interlock set

Instruction	Mnemonic	Code	Length (steps) (See note 1.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
MULTI-INTERLOCK CLEAR (See note 2.)	MILC	519	2	5.0	5.0	5.6	---	8.3	12.5	Interlock not cleared
				5.7	5.7	6.2	---	9.6	14.2	Interlock cleared
JUMP	JMP	004	2	0.31	0.38	0.48	8.1	0.95	0.95	---
JUMP END	JME	005	2	---	---	---	---	---	---	---
CONDITIONAL JUMP	CJP	510	2	0.31	0.38	0.48	7.4	0.95	0.95	When JMP condition is satisfied
CONDITIONAL JUMP NOT	CJPN	511	2	0.31	0.38	0.48	8.5	0.95	0.95	When JMP condition is satisfied
MULTIPLE JUMP	JMP0	515	1	0.048	0.06	0.06	0.12	0.15	0.15	---
MULTIPLE JUMP END	JME0	516	1	0.048	0.06	0.06	0.12	0.15	0.15	---
FOR LOOP	FOR	512	2	0.18	0.21	0.21	0.21	1.00	1.00	Designating a constant
BREAK LOOP	BREAK	514	1	0.048	0.12	0.12	0.12	0.15	0.15	---
NEXT LOOP	NEXT	513	1	0.14	0.18	0.18	0.18	0.45	0.45	When loop is continued
				0.18	0.22	0.22	0.22	0.55	0.55	When loop is ended

Note
1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.
2. Supported only by CPU Units Ver. 2.0 or later.

4-2-4 Timer and Counter Instructions

Instruction	Mnemonic	Code	Length (steps) (See note 1.)	ON execution time (μs)						Conditions
				CPU6 □H-R	CPU6 □H	CPU4 □H	CPU4 □	CJ1M excluding CPU11/21	CJ1M CPU11/21	
HUNDRED-MS TIMER	TIM	---	3	0.45	0.56	0.88	0.42	1.30	1.30	---
	TIMX	550		0.45						
TEN-MS TIMER	TIMH	015	3	0.70	0.88	1.14	0.42	1.80	1.80	---
	TIMHX	551		0.46	0.56	0.88	0.42	1.30	1.30	
ONE-MS TIMER	TMHH	540	3	0.69	0.86	1.12	0.42	1.75	1.75	---
	TMHHX	552		0.46	0.56	0.88	0.42	1.30	1.30	
TENTH-MS TIMER (See note 2.)	TIMU	541	3	0.45	---	---	---	---	---	---
	TIMUX	556		0.45						
HUNDREDTH-MS TIMER (See note 2.)	TMUH	544	3	0.45	---	---	---	---	---	---
	TMUHX	557		0.45						

Instruction	Mnemonic	Code	Length (steps) (See note 1.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
ACCUMULATIVE TIMER	TTIM	087	3	16.1	16.1	17.0	21.4	27.4	30.9	---
				10.9	10.9	11.4	14.8	19.0	21.2	When resetting
				8.5	8.5	8.7	10.7	15.0	16.6	When interlocking
	TTIMX	555		16.1	16.1	17.0	---	27.4	---	---
				10.9	10.9	11.4	---	19.0	---	When resetting
				8.5	8.5	8.7	---	15.0	---	When interlocking
LONG TIMER	TIML	542	4	7.6	7.6	10.0	12.8	16.3	17.2	---
				6.2	6.2	6.5	7.8	13.8	15.3	When interlocking
	TIMLX	553		7.6	7.6	10.0	---	16.3	---	---
				6.2	6.2	6.5	---	13.8	---	When interlocking
MULTI-OUTPUT TIMER	MTIM	543	4	20.9	20.9	23.3	26.0	38.55	43.3	---
				5.6	5.6	5.8	7.8	12.9	13.73	When resetting
	MTIMX	554		20.9	20.9	23.3	---	38.55	---	---
				5.6	5.6	5.8	---	12.9	---	When resetting
COUNTER	CNT	---	3	0.51	0.56	0.88	0.42	1.30	1.30	---
	CNTX	546		0.51					---	
REVERSIBLE COUNTER	CNTR	012	3	16.9	16.9	19.0	20.9	31.8	27.2	---
	CNTRX	548							---	
RESET TIMER/ COUNTER	CNR	545	3	9.9	9.9	10.6	13.9	14.7	17.93	When resetting 1 word
				4.16 ms	4.16 ms	4.16 ms	5.42 ms	6.21 ms	6.30 ms	When resetting 1,000 words
	CNRX	547	3	9.9	9.9	10.6	13.9	14.7	17.93	When resetting 1 word
				4.16 ms	4.16 ms	4.16 ms	5.42 ms	6.21 ms	6.30 ms	When resetting 1,000 words

Note 1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.

2. CJ1-H-R CPU Units only.

4-2-5 Comparison Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
Input Comparison Instructions (unsigned)	LD, AND, OR += LD, AND, OR + <> LD, AND, OR + < LD, AND, OR +<= LD, AND, OR +> LD, AND, OR +>=	300 305 310 315 320 325	4	0.08	0.10	0.16	0.37	0.35	0.35	---
Input Comparison Instructions (double, unsigned)	LD, AND, OR +=+L LD, AND, OR +<>+L LD, AND, OR +<+L LD, AND, OR +<=+L LD, AND, OR +>+L LD, AND, OR +>=+L	301 306 311 316 321 326	4	0.08	0.10	0.16	0.54	0.35	0.35	--- --- --- --- --- ---
Input Comparison Instructions (signed)	LD, AND, OR +=+S LD, AND, OR +<>+S LD, AND, OR +<+S LD, AND, OR +<= LD, AND, OR +>+S LD, AND, OR +>=+S	302 307 312 317 322 327	4	0.08	0.10	0.16	6.50	0.35	0.35	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
Input Comparison Instructions (double, signed)	LD, AND, OR +=+SL	303	4	0.08	0.10	0.16	6.50	0.35	0.35	---
	LD, AND, OR +<>+SL	308								
	LD, AND, OR +<+SL	313								
	LD, AND, OR +<=+SL	318								
	LD, AND, OR +>+SL	323								
	LD, AND, OR +>=+SL	328								
Time Comparison Instructions (See note 2.)	LD, AND, OR +DT	341	4	25.1	25.1	36.4	---	18.8	39.6	---
	LD, AND, OR +<>DT	342	4	25.2	25.2	36.4	---	45.6	40.6	---
	LD, AND, OR +<DT	343	4	25.2	25.2	36.4	---	45.6	40.7	---
	LD, AND, OR +<=DT	344	4	25.2	25.2	36.4	---	18.8	39.6	---
	LD, AND, OR +>DT	345	4	25.1	25.1	36.4	---	45.6	41.1	---
	LD, AND, OR +>=DT	346	4	25.2	25.2	36.4	---	18.8	39.6	---
COMPARE	CMP	20	3	0.032	0.04	0.04	0.29	0.10	0.10	---
	!CMP	20	7	42.1	42.1	42.1	42.4	+45.2	45.2	Increase for immediate refresh
DOUBLE COMPARE	CMPL	60	3	0.064	0.08	0.08	0.46	0.50	0.50	---
SIGNED BINARY COMPARE	CPS	114	3	0.064	0.08	0.08	6.50	0.30	0.30	---
	!CPS	114	7	35.9	35.9	35.9	42.4	+45.2	45.2	Increase for immediate refresh
DOUBLE SIGNED BINARY COMPARE	CPSL	115	3	0.064	0.08	0.08	6.50	0.50	0.50	---
TABLE COMPARE	TCMP	85	4	14.0	14.0	15.2	21.9	29.77	32.13	---
MULTIPLE COMPARE	MCMP	19	4	20.5	20.5	22.8	31.2	45.80	48.67	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
UNSIGNED BLOCK COMPARE	BCMP	68	4	21.5	21.5	23.7	32.6	47.93	51.67	---
EXPANDED BLOCK COMPARE	BCMP2	502	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	88	3	5.3	5.3	5.4	---	11.53	12.43	---
DOUBLE AREA RANGE COMPARE	ZCPL	116	3	5.5	5.5	6.7	---	11.28	11.90	---

Note
1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.
2. Supported only by CPU Units Ver. 2.0 or later.

4-2-6 Data Movement Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
MOVE	MOV	21	3	0.14	0.18	0.20	0.29	0.30	0.30	---
	!MOV	21	7	21.38	21.38	21.40	42.36	+35.1	43.0	Increase for immediate refresh
DOUBLE MOVE	MOVL	498	3	0.26	0.32	0.34	0.50	0.60	0.60	---
MOVE NOT	MVN	22	3	0.14	0.18	0.20	0.29	0.35	0.35	---
DOUBLE MOVE NOT	MVNL	499	3	0.26	0.32	0.34	0.50	0.60	0.60	---
MOVE BIT	MOVB	82	4	0.19	0.24	0.34	7.5	0.50	0.50	---
MOVE DIGIT	MOVD	83	4	0.19	0.24	0.34	7.3	0.50	0.50	---
MULTIPLE BIT TRANSFER	XFRB	62	4	10.1	10.1	10.8	13.6	20.9	22.1	Transferring 1 bit
				186.4	186.4	189.8	269.2	253.3	329.7	Transferring 255 bits
BLOCK TRANSFER	XFER	70	4	0.29	0.36	0.44	11.2	0.8	0.8	Transferring 1 word
				240.1	300.1	380.1	633.5	650.2	650.2	Transferring 1,000 words
BLOCK SET	BSET	71	4	0.21	0.26	0.28	8.5	0.55	0.55	Setting 1 word
				142.2	200.1	220.1	278.3	400.2	400.2	Setting 1,000 words
DATA EXCHANGE	XCHG	73	3	0.32	0.40	0.56	0.7	0.80	0.80	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
DOUBLE DATA EXCHANGE	XCGL	562	3	0.61	0.76	1.04	1.3	1.5	1.5	---
SINGLE WORD DISTRIBUTE	DIST	80	4	5.1	5.1	5.4	7.0	6.6	12.47	---
DATA COLLECT	COLL	81	4	5.1	5.1	5.3	7.1	6.5	12.77	---
MOVE TO REGISTER	MOVR	560	3	0.064	0.08	0.08	0.50	0.60	0.60	---
MOVE TIMER/ COUNTER PV TO REGISTER	MOVRW	561	3	0.064	0.42	0.50	0.50	0.60	0.60	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-7 Data Shift Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
SHIFT REGISTER	SFT	10	3	7.4	7.4	10.4	10.4	11.9	15.3	Shifting 1 word
				187.3	433.2	488.0	763.1	1.39 ms	1.43 ms	Shifting 1,000 words
REVERSIBLE SHIFT REGISTER	SFTR	84	4	6.9	6.9	7.2	9.6	11.4	15.5	Shifting 1 word
				399.3	615.3	680.2	859.6	1.43 ms	1.55 ms	Shifting 1,000 words
ASYNCHRONOUS SHIFT REGISTER	ASFT	17	4	6.2	6.2	6.4	7.7	13.4	14.2	Shifting 1 word
				1.22 ms	1.22 ms	1.22 ms	2.01 ms	2.75 ms	2.99 ms	Shifting 1,000 words
WORD SHIFT	WSFT	16	4	4.5	4.5	4.7	7.8	9.6	12.3	Shifting 1 word
				171.5	171.5	171.7	781.7	928.0	933.3	Shifting 1,000 words
ARITHMETIC SHIFT LEFT	ASL	25	2	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE SHIFT LEFT	ASLL	570	2	0.32	0.40	0.56	0.67	0.80	0.80	---
ARITHMETIC SHIFT RIGHT	ASR	26	2	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE SHIFT RIGHT	ASRL	571	2	0.32	0.40	0.56	0.67	0.80	0.80	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
ROTATE LEFT	ROL	27	2	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE ROTATE LEFT	ROLL	572	2	0.32	0.40	0.56	0.67	0.80	0.80	---
ROTATE LEFT WITHOUT CARRY	RLNC	574	2	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	576	2	0.32	0.40	0.56	0.67	0.80	0.80	---
ROTATE RIGHT	ROR	28	2	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE ROTATE RIGHT	RORL	573	2	0.32	0.40	0.56	0.67	0.80	0.80	---
ROTATE RIGHT WITHOUT CARRY	RRNC	575	2	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	577	2	0.32	0.40	0.56	0.67	0.80	0.80	---
ONE DIGIT SHIFT LEFT	SLD	74	3	5.9	5.9	6.1	8.2	7.6	12.95	Shifting 1 word
				561.1	561.1	626.3	760.7	1.15 ms	1.27 ms	Shifting 1,000 words
ONE DIGIT SHIFT RIGHT	SRD	75	3	6.9	6.9	7.1	8.7	8.6	15.00	Shifting 1 word
				760.5	760.5	895.5	1.07 ms	1.72 ms	1.82 ms	Shifting 1,000 words
SHIFT N-BIT DATA LEFT	NSFL	578	4	7.5	7.5	8.3	10.5	14.8	16.0	Shifting 1 bit
				34.5	40.3	45.4	55.5	86.7	91.3	Shifting 1,000 bits
SHIFT N-BIT DATA RIGHT	NSFR	579	4	7.5	7.5	8.3	10.5	14.7	15.9	Shifting 1 bit
				48.2	50.5	55.3	69.3	114.1	119.6	Shifting 1,000 bits
SHIFT N-BITS LEFT	NASL	580	3	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE SHIFT N-BITS LEFT	NSLL	582	3	0.32	0.40	0.56	0.67	0.80	0.80	---
SHIFT N-BITS RIGHT	NASR	581	3	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE SHIFT N-BITS RIGHT	NSRL	583	3	0.32	0.40	0.56	0.67	0.80	0.80	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-8 Increment/Decrement Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
INCREMENT BINARY	++	590	2	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE INCREMENT BINARY	++L	591	2	0.18	0.40	0.56	0.67	0.80	0.80	---
DECREMENT BINARY	– –	592	2	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE DECREMENT BINARY	– –L	593	2	0.18	0.40	0.56	0.67	0.80	0.80	---
INCREMENT BCD	++B	594	2	5.7	6.4	4.5	7.4	12.3	14.7	---
DOUBLE INCREMENT BCD	++BL	595	2	5.6	5.6	4.9	6.1	9.24	10.8	---
DECREMENT BCD	– –B	596	2	5.7	6.3	4.6	7.2	11.9	14.9	---
DOUBLE DECREMENT BCD	– –BL	597	2	5.3	5.3	4.7	7.1	9.0	10.7	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-9 Symbol Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
SIGNED BINARY ADD WITHOUT CARRY	+	400	4	0.18	0.18	0.20	0.37	0.30	0.30	---
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	401	4	0.18	0.32	0.34	0.54	0.60	0.60	---
SIGNED BINARY ADD WITH CARRY	+C	402	4	0.18	0.18	0.20	0.37	0.40	0.40	---
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	403	4	0.18	0.32	0.34	0.54	0.60	0.60	---
BCD ADD WITHOUT CARRY	+B	404	4	7.6	8.2	8.4	14.0	18.9	21.5	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11 /21	CJ1M CPU11 /21	
DOUBLE BCD ADD WITHOUT CARRY	+BL	405	4	9.2	13.3	14.5	19.0	24.4	27.7	---
BCD ADD WITH CARRY	+BC	406	4	8.0	8.9	9.1	14.5	19.7	22.6	---
DOUBLE BCD ADD WITH CARRY	+BCL	407	4	9.6	13.8	15.0	19.6	25.2	28.8	---
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	410	4	0.18	0.18	0.20	0.37	0.3	0.3	---
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	411	4	0.18	0.32	0.34	0.54	0.60	0.60	---
SIGNED BINARY SUBTRACT WITH CARRY	–C	412	4	0.18	0.18	0.20	0.37	0.3	0.3	---
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	413	4	0.18	0.32	0.34	0.54	0.60	0.60	---
BCD SUBTRACT WITHOUT CARRY	–B	414	4	7.4	8.0	8.2	13.1	18.1	20.5	---
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	415	4	8.9	12.8	14.0	18.2	23.2	26.7	---
BCD SUBTRACT WITH CARRY	–BC	416	4	7.9	8.5	8.6	13.8	19.1	21.6	---
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	417	4	9.4	13.4	14.7	18.8	24.3	27.7	---
SIGNED BINARY MULTIPLY	*	420	4	0.26	0.38	0.40	0.58	0.65	0.65	---
DOUBLE SIGNED BINARY MULTIPLY	*L	421	4	5.93	7.23	8.45	11.19	13.17	15.0	---
UNSIGNED BINARY MULTIPLY	*U	422	4	0.26	0.38	0.40	0.58	0.75	0.75	---
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	423	4	5.9	7.1	8.3	10.63	13.30	15.2	---
BCD MULTIPLY	*B	424	4	8.3	9.0	9.2	12.8	17.5	19.7	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11 /21	CJ1M CPU11 /21	
DOUBLE BCD MULTIPLY	*BL	425	4	12.8	23.0	24.2	35.2	36.3	45.7	---
SIGNED BINARY DIVIDE	/	430	4	0.29	0.40	0.42	0.83	0.70	0.70	---
DOUBLE SIGNED BINARY DIVIDE	/L	431	4	7.2	7.2	8.4	9.8	13.7	15.5	---
UNSIGNED BINARY DIVIDE	/U	432	4	0.29	0.40	0.42	0.83	0.8	0.8	---
DOUBLE UNSIGNED BINARY DIVIDE	/UL	433	4	6.9	6.9	8.1	9.1	12.8	14.7	---
BCD DIVIDE	/B	434	4	8.6	8.6	8.8	15.9	19.3	22.8	---
DOUBLE BCD DIVIDE	/BL	435	4	13.1	17.7	18.9	26.2	27.1	34.7	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-10 Conversion Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11 /21	CJ1M CPU11 /21	
BCD TO BINARY	BIN	023	3	0.18	0.22	0.24	0.29	0.40	0.40	---
DOUBLE BCD TO DOUBLE BINARY	BINL	058	3	6.1	6.5	6.8	9.1	12.3	13.7	---
BINARY TO BCD	BCD	024	3	0.19	0.24	0.26	8.3	7.62	9.78	---
DOUBLE BINARY TO DOUBLE BCD	BCDL	059	3	6.7	6.7	7.0	9.2	10.6	12.8	---
2'S COMPLEMENT	NEG	160	3	0.14	0.18	0.20	0.29	0.35	0.35	---
DOUBLE 2'S COMPLEMENT	NEGL	161	3	0.26	0.32	0.34	0.5	0.60	0.60	---
16-BIT TO 32-BIT SIGNED BINARY	SIGN	600	3	0.26	0.32	0.34	0.50	0.60	0.60	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
DATA DECODER	MLPX	076	4	0.32	0.32	0.42	8.8	0.85	0.85	Decoding 1 digit (4 to 16)
				0.98	0.98	1.20	12.8	1.60	1.60	Decoding 4 digits (4 to 16)
				3.30	3.30	4.00	20.3	4.70	4.70	Decoding 1 digit (8 to 256)
				6.50	6.50	7.90	33.4	8.70	8.70	Decoding 2 digits (8 to 256)
DATA ENCODER	DMPX	077	4	7.5	7.5	7.9	10.4	9.4	13.9	Encoding 1 digit (16 to 4)
				49.6	49.6	50.2	59.1	57.3	71.73	Encoding 4 digits (16 to 4)
				18.2	18.2	18.6	23.6	56.8	82.7	Encoding 1 digit (256 to 8)
				55.1	55.1	57.4	92.5	100.0	150.7	Encoding 2 digits (256 to 8)
ASCII CONVERT	ASC	086	4	6.8	6.8	7.1	9.7	8.3	14.6	Converting 1 digit into ASCII
				9.0	11.2	11.7	15.1	19.1	21.8	Converting 4 digits into ASCII
ASCII TO HEX	HEX	162	4	7.1	7.1	7.4	10.1	12.1	15.6	Converting 1 digit
COLUMN TO LINE	LINE	063	4	16.6	19.0	23.1	29.1	37.0	40.3	---
LINE TO COLUMN	COLM	064	4	18.4	23.2	27.5	37.3	45.7	48.2	---
SIGNED BCD TO BINARY	BINS	470	4	6.8	8.0	8.3	12.1	16.2	17.0	Data format setting No. 0
				6.8	8.0	8.3	12.1	16.2	17.1	Data format setting No. 1
				7.1	8.3	8.6	12.7	16.5	17.7	Data format setting No. 2
				7.4	8.5	8.8	13.0	16.5	17.6	Data format setting No. 3
DOUBLE SIGNED BCD TO BINARY	BISL	472	4	6.9	9.2	9.6	13.6	18.4	19.6	Data format setting No. 0
				7.0	9.2	9.6	13.7	18.5	19.8	Data format setting No. 1
				7.3	9.5	9.9	14.2	18.6	20.1	Data format setting No. 2
				7.6	9.6	10.0	14.4	18.7	20.1	Data format setting No. 3
SIGNED BINARY TO BCD	BCDS	471	4	6.6	6.6	6.9	10.6	13.5	16.4	Data format setting No. 0
				6.7	6.7	7.0	10.8	13.8	16.7	Data format setting No. 1
				6.8	6.8	7.1	10.9	13.9	16.8	Data format setting No. 2
				7.1	7.2	7.5	11.5	14.0	17.1	Data format setting No. 3

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11 /21	CJ1M CPU11 /21	
DOUBLE SIGNED BINARY TO BCD	BDSL	473	4	7.6	8.1	8.4	11.6	11.4	12.5	Data format setting No. 0
				6.7	8.2	8.6	11.8	11.7	12.73	Data format setting No. 1
				6.7	8.3	8.7	12.0	11.8	12.8	Data format setting No. 2
				6.9	8.8	9.2	12.5	11.9	13.0	Data format setting No. 3
GRAY CODE CONVERSION (See note 2.)	GRY	474	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	601	3	13.79	13.79	20.24	---	22.16	19.88	---
EIGHT-DIGIT NUMBER TO ASCII	STR8	602	3	18.82	18.82	27.44	---	29.55	26.70	---
SIXTEEN-DIGIT NUMBER TO ASCII	STR16	603	3	30.54	30.54	44.41	---	48.16	44.10	---
ASCII TO FOUR-DIGIT NUMBER	NUM4	604	3	18.46	18.46	27.27	---	29.13	26.88	---
ASCII TO EIGHT-DIGIT NUMBER	NUM8	605	3	27.27	27.27	40.29	---	42.69	39.71	---
ASCII TO SIXTEEN-DIGIT NUMBER	NUM16	606	3	52.31	52.31	78.25	---	82.21	74.23	---

Note
1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.
2. Supported only by CPU Units Ver. 2.0 or later.

4-2-11 Logic Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
LOGICAL AND	ANDW	034	4	0.14	0.18	0.20	0.37	0.30	0.30	---
DOUBLE LOGICAL AND	ANDL	610	4	0.26	0.32	0.34	0.54	0.60	0.60	---
LOGICAL OR	ORW	035	4	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE LOGICAL OR	ORWL	611	4	0.26	0.32	0.34	0.54	0.60	0.60	---
EXCLUSIVE OR	XORW	036	4	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE EXCLUSIVE OR	XORL	612	4	0.26	0.32	0.34	0.54	0.60	0.60	---
EXCLUSIVE NOR	XNRW	037	4	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE EXCLUSIVE NOR	XNRL	613	4	0.26	0.32	0.34	0.54	0.60	0.60	---
COMPLEMENT	COM	029	2	0.18	0.22	0.32	0.37	0.45	0.45	---
DOUBLE COMPLEMENT	COML	614	2	0.32	0.40	0.56	0.67	0.80	0.80	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-12 Special Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
BINARY ROOT	ROTB	620	3	49.6	49.6	50.0	530.7	56.5	82.7	---
BCD SQUARE ROOT	ROOT	072	3	13.7	13.7	13.9	514.5	59.3	88.4	---
ARITHMETIC PROCESS	APR	069	4	6.7	6.7	6.9	32.3	14.0	15.0	Designating SIN and COS
				17.2	17.2	18.4	78.3	32.2	37.9	Designating line-segment approximation
FLOATING POINT DIVIDE	FDIV	079	4	116.6	116.6	176.6	176.6	246.0	154.7	---
BIT COUNTER	BCNT	067	4	0.24	0.3	0.38	22.1	0.65	0.65	Counting 1 word

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-13 Floating-point Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
FLOATING TO 16-BIT	FIX	450	3	0.13	10.6	10.8	14.5	16.2	19.5	---
FLOATING TO 32-BIT	FIXL	451	3	0.13	10.8	11.0	14.6	16.6	21.7	---
16-BIT TO FLOATING	FLT	452	3	0.13	8.3	8.5	11.1	12.2	14.6	---
32-BIT TO FLOATING	FLTL	453	3	0.13	8.3	8.5	10.8	14.0	15.8	---
FLOATING-POINT ADD	+F	454	4	0.24	8.0	9.2	10.2	13.3	15.7	---
FLOATING-POINT SUBTRACT	–F	455	4	0.24	8.0	9.2	10.3	13.3	15.8	---
FLOATING-POINT DIVIDE	/F	457	4	0.4	8.7	9.9	12.0	14.0	17.6	---
FLOATING-POINT MULTIPLY	*F	456	4	0.24	8.0	9.2	10.5	13.2	15.8	---
DEGREES TO RADIANS	RAD	458	3	8.1	10.1	10.2	14.9	15.9	20.6	---
RADIANS TO DEGREES	DEG	459	3	8.0	9.9	10.1	14.8	15.7	20.4	---
SINE	SIN	460	3	42.0	42.0	42.2	61.1	47.9	70.9	---
HIGH-SPEED SINE (See note 2.)	SINQ	475	8	0.59	---	---	---	---	---	---
COSINE	COS	461	3	31.5	31.5	31.8	44.1	41.8	51.0	---
HIGH-SPEED COSINE (See note 2.)	COSQ	476	8	0.59	---	---	---	---	---	---
TANGENT	TAN	462	3	16.3	16.3	16.6	22.6	20.8	27.6	---
HIGH-SPEED TANGENT (See note 2.)	TANQ	477	15	1.18	---	---	---	---	---	---
ARC SINE	ASIN	463	3	17.6	17.6	17.9	24.1	80.3	122.9	---
ARC COSINE	ACOS	464	3	20.4	20.4	20.7	28.0	25.3	33.5	---
ARC TANGENT	ATAN	465	3	16.1	16.1	16.4	16.4	45.9	68.9	---
SQUARE ROOT	SQRT	466	3	0.42	19.0	19.3	28.1	26.2	33.2	---
EXPONENT	EXP	467	3	65.9	65.9	66.2	96.7	68.8	108.2	---
LOGARITHM	LOG	468	3	12.8	12.8	13.1	17.4	69.4	103.7	---
EXPONENTIAL POWER	PWR	840	4	125.4	125.4	126.0	181.7	134.0	201.0	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11 /21	CJ1M CPU11 /21	
Floating Symbol Comparison	LD, AND, OR +=F	329	3	0.13	6.6	8.3	---	12.6	15.37	---
	LD, AND, OR +<>F	330								
	LD, AND, OR +<F	331								
	LD, AND, OR +<=F	332								
	LD, AND, OR +>F	333								
	LD, AND, OR +>=F	334								
FLOATING-POINT TO ASCII	FSTR	448	4	48.5	48.5	48.9	---	58.4	85.7	---
ASCII TO FLOATING-POINT	FVAL	449	3	21.1	21.1	21.3	---	31.1	43.773	---
MOVE FLOATING-POINT (SINGLE) (See note 2.)	MOVF	469	3	0.18	---	---	---	---	---	---

Note 1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.

2. CJ1-H-R CPU Units only.

4-2-14 Double-precision Floating-point Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
DOUBLE SYMBOL COMPARISON	LD, AND, OR +=D	335	3	8.5	8.5	10.3	---	16.2	19.9	---
	LD, AND, OR +<>D	336								
	LD, AND, OR +<D	337								
	LD, AND, OR +<=D	338								
	LD, AND, OR +>D	339								
	LD, AND, OR +>=D	340								
DOUBLE FLOATING TO 16-BIT BINARY	FIXD	841	3	11.0	11.7	12.1	---	16.1	21.6	---
DOUBLE FLOATING TO 32-BIT BINARY	FIXLD	842	3	10.2	11.6	12.1	---	16.4	21.7	---
16-BIT BINARY TO DOUBLE FLOATING	DBL	843	3	9.9	9.9	10.0	---	14.3	16.5	---
32-BIT BINARY TO DOUBLE FLOATING	DBLL	844	3	9.8	9.8	10.0	---	16.0	17.7	---
DOUBLE FLOATING-POINT ADD	+D	845	4	11.2	11.2	11.9	---	18.3	23.6	---
DOUBLE FLOATING-POINT SUBTRACT	–D	846	4	11.2	11.2	11.9	---	18.3	23.6	---
DOUBLE FLOATING-POINT MULTIPLY	*D	847	4	12.0	12.0	12.7	---	19.0	25.0	---
DOUBLE FLOATING-POINT DIVIDE	/D	848	4	23.5	23.5	24.2	---	30.5	44.3	---
DOUBLE DEGREES TO RADIANS	RADD	849	3	11.5	27.4	27.8	---	32.7	49.1	---
DOUBLE RADIANS TO DEGREES	DEGD	850	3	11.2	11.2	11.9	---	33.5	48.4	---
DOUBLE SINE	SIND	851	3	45.4	45.4	45.8	---	67.9	76.7	---
DOUBLE COSINE	COSD	852	3	43.0	43.0	43.4	---	70.9	72.3	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
DOUBLE TANGENT	TAND	853	3	19.8	20.1	20.5	---	97.9	157.0	---
DOUBLE ARC SINE	ASIND	854	3	21.5	21.5	21.9	---	32.3	37.3	---
DOUBLE ARC COSINE	ACOSD	855	3	24.7	24.7	25.1	---	29.9	42.5	---
DOUBLE ARC TANGENT	ATAND	856	3	19.3	19.3	19.7	---	24.0	34.4	---
DOUBLE SQUARE ROOT	SQRTD	857	3	47.4	47.4	47.9	---	52.9	81.9	---
DOUBLE EXPONENT	EXPD	858	3	121.0	121.0	121.4	---	126.3	201.3	---
DOUBLE LOGARITHM	LOGD	859	3	16.0	16.0	16.4	---	21.6	29.3	---
DOUBLE EXPONENTIAL POWER	PWRD	860	4	223.9	223.9	224.2	---	232.3	373.4	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-15 Table Data Processing Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
SET STACK	SSET	630	3	8.0	8.0	8.3	8.5	14.2	20.3	Designating 5 words in stack area
				231.6	231.6	251.8	276.8	426.5	435.3	Designating 1,000 words in stack area
PUSH ONTO STACK	PUSH	632	3	6.5	6.5	8.6	9.1	15.7	16.4	---
FIRST IN FIRST OUT	FIFO	633	3	6.9	6.9	8.9	10.6	15.8	16.8	Designating 5 words in stack area
				352.6	352.6	434.3	1.13 ms	728.0	732.0	Designating 1,000 words in stack area
LAST IN FIRST OUT	LIFO	634	3	7.0	7.0	9.0	9.9	16.6	17.2	---
DIMENSION RECORD TABLE	DIM	631	5	15.2	15.2	21.6	142.1	27.8	27.1	---
SET RECORD LOCATION	SETR	635	4	5.4	5.4	5.9	7.0	12.8	13.2	---
GET RECORD NUMBER	GETR	636	4	7.8	7.8	8.4	11.0	16.1	18.3	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
DATA SEARCH	SRCH	181	4	15.5	15.5	19.5	19.5	29.1	26.4	Searching for 1 word
				2.42 ms	2.42 ms	3.34 ms	3.34 ms	4.41 ms	3.60 ms	Searching for 1,000 words
SWAP BYTES	SWAP	637	3	12.2	12.2	13.6	13.6	21.0	18.4	Swapping 1 word
				1.94 ms	1.94 ms	2.82 ms	2.82 ms	3.65 ms	3.15 ms	Swapping 1,000 words
FIND MAXIMUM	MAX	182	4	19.2	19.2	24.9	24.9	35.3	32.0	Searching for 1 word
				2.39 ms	2.39 ms	3.36 ms	3.36 ms	4.39 ms	3.57 ms	Searching for 1,000 words
FIND MINIMUM	MIN	183	4	19.2	19.2	25.3	25.3	35.4	31.9	Searching for 1 word
				2.39 ms	2.39 ms	3.33 ms	3.33 ms	4.39 ms	3.58 ms	Searching for 1,000 words
SUM	SUM	184	4	28.2	28.2	38.5	38.3	49.5	44.1	Adding 1 word
				14.2 ms	1.42 ms	1.95 ms	1.95 ms	2.33 ms	2.11 ms	Adding 1,000 words
FRAME CHECKSUM	FCS	180	4	20.0	20.0	28.3	28.3	34.8	31.5	For 1-word table length
				1.65 ms	1.65 ms	2.48 ms	2.48 ms	3.11 ms	2.77 ms	For 1,000-word table length
STACK SIZE READ	SNUM	638	3	6.0	6.0	6.3	---	12.1	13.7	---
STACK DATA READ	SREAD	639	4	8.0	8.0	8.4	---	18.1	20.6	---
STACK DATA OVERWRITE	SWRIT	640	4	7.2	7.2	7.6	---	16.9	18.8	---
STACK DATA INSERT	SINS	641	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	642	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

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-16 Data Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
PID CONTROL	PID	190	4	436.2	436.2	678.2	678.2	612.0	552.6	Initial execution
				332.3	332.3	474.9	474.9	609.3	548.0	Sampling
				97.3	97.3	141.3	141.3	175.3	162.0	Not sampling
LIMIT CONTROL	LMT	680	4	16.1	16.1	22.1	22.1	27.1	26.1	---
DEAD BAND CONTROL	BAND	681	4	17.0	17.0	22.5	22.5	27.4	26.6	---
DEAD ZONE CONTROL	ZONE	682	4	15.4	15.4	20.5	20.5	28.0	26.4	---
TIME-PROPORTIONAL OUTPUT (See note 2.)	TPO	685	4	10.6	10.6	14.8	---	20.2	19.8	OFF execution time
				54.5	54.5	82.0	---	92.7	85.1	ON execution time with duty designation or displayed output limit
				61.0	61.0	91.9	---	102.5	95.3	ON execution time with manipulated variable designation and output limit enabled
SCALING	SCL	194	4	13.9	13.9	14.3	56.8	25.0	32.8	---
SCALING 2	SCL2	486	4	12.2	12.2	12.6	50.7	22.3	29.1	---
SCALING 3	SCL3	487	4	13.7	13.7	14.2	57.7	25.6	30.0	---
AVERAGE	AVG	195	4	36.3	36.3	52.6	53.1	62.9	59.1	Average of an operation
				291.0	291.0	419.9	419.9	545.3	492.7	Average of 64 operations
PID CONTROL WITH AUTOTUNING	PIDAT	191	4	446.3	446.3	712.5	---	765.3	700.0	Initial execution
				339.4	339.4	533.9	---	620.7	558.0	Sampling
				100.7	100.7	147.1	---	180.0	166.1	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

Note 1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.

2. Supported only by CPU Units Ver. 2.0 or later.

4-2-17 Subroutine Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
SUBROUTINE CALL	SBS	91	2	0.90	1.26	1.96	17.0	2.04	2.04	---
SUBROUTINE ENTRY	SBN	92	2	---	---	---	---	---	---	---
SUBROUTINE RETURN	RET	93	1	0.43	0.86	1.60	20.60	1.80	1.80	---
MACRO	MCRO	99	4	23.3	23.3	23.3	23.3	47.9	50.3	---
GLOBAL SUBROUTINE CALL	GSBN	751	2	---	---	---	---	---	---	---
GLOBAL SUBROUTINE ENTRY	GRET	752	1	0.90	1.26	1.96	---	2.04	2.04	---
GLOBAL SUBROUTINE RETURN	GSBS	750	2	0.43	0.86	1.60	---	1.80	1.80	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-18 Interrupt Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
SET INTERRUPT MASK	MSKS	690	3	25.6	25.6	38.4	39.5	44.7	42.9	---
READ INTERRUPT MASK	MSKR	692	3	11.9	11.9	11.9	11.9	16.9	15.9	---
CLEAR INTERRUPT	CLI	691	3	27.4	27.4	41.3	41.3	42.7	44.5	---
DISABLE INTERRUPTS	DI	693	1	15.0	15.0	16.8	16.8	30.3	28.5	---
ENABLE INTERRUPTS	EI	694	1	19.5	19.5	21.8	21.8	37.7	34.4	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-19 High-speed Counter and Pulse Output Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
MODE CONTROL	INI	880	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	881	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

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
COMPARISON TABLE LOAD	CTBL	882	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
COUNTER FREQUENCY CONVERT	PRV2	883	4	---	---	---	---	23.03	22.39	---
SPEED OUTPUT	SPED	885	4	---	---	---	---	56.00	89.3	Continuous mode
				---	---	---	---	62.47	94.9	Independent mode
SET PULSES	PULS	886	4	---	---	---	---	26.20	32.9	---
PULSE OUTPUT	PLS2	887	5	---	---	---	---	100.80	107.5	---
ACCELERATION CONTROL	ACC	888	4	---	---	---	---	90.80	114.8	Continuous mode
				---	---	---	---	80.00	122.1	Independent mode
ORIGIN SEARCH	ORG	889	3	---	---	---	---	106.13	116.0	Origin search
				---	---	---	---	52.00	102.1	Origin return
PULSE WITH VARIABLE DUTY FACTOR	PWM	891	4	---	---	---	---	25.80	33.0	---

Note Supported only by CPU Units Ver. 2.0 or later.

4-2-20 Step Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
STEP DEFINE	STEP	008	2	17.4	17.4	20.7	27.1	35.9	37.1	Step control bit ON
				11.8	11.8	13.7	24.4	13.8	18.3	Step control bit OFF
STEP START	SNXT	009	2	6.6	6.6	7.3	10.0	12.1	14.0	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-21 Basic I/O Unit Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
I/O REFRESH	IORF	097	3	15.5	15.5	16.4	23.5	26.7	30.4	1-word refresh (IN) for Basic I/O Units
				17.20	17.20	18.40	25.6	29.7	35.0	1-word refresh (OUT) for Basic I/O Units
				319.9	319.9	320.7	377.6	291.0	100.0	60-word refresh (IN) for Basic I/O Units
				358.00	358.00	354.40	460.1	325.0	134.7	60-word refresh (OUT) for Basic I/O Units
SPECIAL I/O UNIT I/O REFRESH (See note 4.)	FIORF	225	2	--- (See note 3.)	---	---	---	---	---	---
CPU BUS I/O REFRESH	DLNK	226	4	287.8	287.8	315.5	---	321.3	458.7	Allocated 1 word
7-SEGMENT DECODER	SDEC	078	4	6.5	6.5	6.9	14.1	8.1	15.7	---
DIGITAL SWITCH INPUT (See note 2.)	DSW	210	6	50.7	50.7	73.5	---	77.7	77.6	4 digits, data input value: 0
				51.5	51.5	73.4	---	77.9	77.6	4 digits, data input value: F
				51.3	51.3	73.5	---	83.2	80.0	8 digits, data input value: 0
				50.7	50.7	73.4	---	77.9	77.7	8 digits, data input value: F
TEN KEY INPUT (See note 2.)	TKY	211	4	9.7	9.7	13.2	---	18.7	18.6	Data input value: 0
				10.7	10.7	14.8	---	20.2	19.1	Data input value: F
HEXADECIMAL KEY INPUT (See note 2.)	HKY	212	5	50.3	50.3	70.9	---	77.3	78.1	Data input value: 0
				50.1	50.1	71.2	---	76.8	77.3	Data input value: F

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
MATRIX INPUT (See note 2.)	MTR	213	5	47.8	47.8	68.1	---	76.4	77.7	Data input value: 0
				48.0	48.0	68.0	---	77.7	76.9	Data input value: F
7-SEGMENT DISPLAY OUTPUT (See note 2.)	7SEG	214	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	222	4	--- (See note 3.)	--- (See note 3.)	--- (See note 3.)	--- (See note 3.)	225.3	217.7	First execution
								232.0	241.7	When busy
								223.0	215.3	At end
INTELLIGENT I/O WRITE	IOWR	223	4	--- (See note 3.)	--- (See note 3.)	--- (See note 3.)	--- (See note 3.)	245.3	219.7	First execution
								231.0	225.7	When busy
								244.0	218.7	At end

Note
1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.
2. Supported only by CPU Units Ver. 2.0 or later.
3. Execution times for the FIORF, IORD, and IORW instructions depends on the Special I/O Unit from which data is being read.
4. CJ1-H-R CPU Units only.

4-2-22 Serial Communications Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
PROTOCOL MACRO	PMCR	260	5	100.1	100.1	142.1	276.8	158.4	206.0	Sending 0 words, receiving 0 words
				134.2	134.2	189.6	305.9	210.0	256.7	Sending 1 word, receiving 1 word
TRANSMIT	TXD	236	4	68.5	68.5	98.8	98.8	109.3	102.9	Sending 1 byte
				734.3	734.3	1.10 ms	1.10 ms	1.23 ms	1.16 ms	Sending 256 bytes
RECEIVE	RXD	235	4	89.6	89.6	131.1	131.1	144.0	132.1	Storing 1 byte
				724.2	724.2	1.11 ms	1.11 ms	1.31 ms	1.22 ms	Storing 256 bytes
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	256	4	131.5	131.5	202.4	---	213.4	208.6	Sending 1 byte

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6 □H-R	CPU6 □H	CPU4 □H	CPU4 □	CJ1M excluding CPU11/21	CJ1M CPU11/21	
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	255	4	131	131	200.8	---	211.8	206.8	Storing 1 byte
CHANGE SERIAL PORT SETUP	STUP	237	3	341.2	341.2	400.0	440.4	504.7	524.7	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-23 Network Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6 □H-R	CPU6 □H	CPU4 □H	CPU4 □	CJ1M excluding CPU11/21	CJ1M CPU11/21	
NETWORK SEND	SEND	090	4	84.4	84.4	123.9	123.9	141.6	195.0	---
NETWORK RECEIVE	RECV	098	4	85.4	85.4	124.7	124.7	142.3	196.7	---
DELIVER COMMAND	CMND	490	4	106.8	106.8	136.8	136.8	167.7	226.7	---
EXPLICIT MESSAGE SEND (See note 2.)	EXPLT	720	4	127.6	127.6	190.0	---	217.0	238.0	---
EXPLICIT GET ATTRIBUTE (See note 2.)	EGATR	721	4	123.9	123.9	185.0	---	210.0	232.7	---
EXPLICIT SET ATTRIBUTE (See note 2.)	ESATR	722	3	110.0	110.0	164.4	---	188.3	210.3	---
EXPLICIT WORD READ (See note 2.)	ECHRD	723	4	106.8	106.8	158.9	---	176.3	220.3	---
EXPLICIT WORD WRITE (See note 2.)	ECHWR	724	4	106.0	106.0	158.3	---	175.7	205.3	---

Note
1. When a double-length operand is used, add 1 to the value shown in the length column in the following table.
2. Supported only by CPU Units Ver. 2.0 or later.

4-2-24 File Memory Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
READ DATA FILE	FREAD	700	5	391.4	391.4	632.4	684.1	657.3	641.3	2-character directory + file name in binary
				836.1	836.1	1.33 ms	1.35 ms	1.45 ms	1.16 ms	73-character directory + file name in binary
WRITE DATA FILE	FWRIT	701	5	387.8	387.8	627.0	684.7	650.7	637.3	2-character directory + file name in binary
				833.3	833.3	1.32 ms	1.36 ms	1.44 ms	1.16 ms	73-character directory + file name in binary
WRITE TEXT FILE	TWRIT	704	5	390.1	390.1	619.1	---	555.3	489.0	

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-25 Display Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
DISPLAY MESSAGE	MSG	046	3	10.1	10.1	14.2	14.3	16.8	17.3	Displaying message
				8.4	8.4	11.3	11.3	14.7	14.7	Deleting displayed message

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-26 Clock Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
CALENDAR ADD	CADD	730	4	34.0	38.3	201.9	209.5	217.0	194.0	---
CALENDAR SUBTRACT	CSUB	731	4	29.6	38.6	170.4	184.1	184.7	167.0	---
HOURS TO SECONDS	SEC	065	3	7.8	21.4	29.3	35.8	36.1	35.4	---
SECONDS TO HOURS	HMS	066	3	7.7	22.2	30.9	42.1	45.1	45.7	---
CLOCK ADJUSTMENT	DATE	735	2	216.0	216.0	251.5	120.0	118.7	128.3	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-27 Debugging Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
TRACE MEMORY SAMPLING	TRSM	045	1	80.4	80.4	120.0	120.0	207.0	218.3	Sampling 1 bit and 0 words
				848.1	848.1	1.06 ms	1.06 ms	1.16 ms	1.10 ms	Sampling 31 bits and 6 words

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-28 Failure Diagnosis Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
FAILURE ALARM	FAL	006	3	15.4	15.4	16.7	16.7	26.1	24.47	Recording errors
				179.8	179.8	244.8	244.8	294.0	264.0	Deleting errors (in order of priority)
				432.4	432.4	657.1	657.1	853.3	807.3	Deleting errors (all errors)
				161.5	161.5	219.4	219.4	265.7	233.0	Deleting errors (individually)
SEVERE FAILURE ALARM	FALS	007	3	---	---	---	---	---	---	---
FAILURE POINT DETECTION	FPD	269	4	140.9	140.9	202.3	202.3	220.7	250.0	When executed
				163.4	163.4	217.6	217.6	250.3	264.3	First time
				185.2	185.2	268.9	268.9	220.7	321.7	When executed
				207.5	207.5	283.6	283.6	320.7	336.0	First time

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-29 Other Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11 /21	CJ1M CPU11 /21	
SET CARRY	STC	040	1	0.048	0.06	0.06	0.12	0.15	0.15	---
CLEAR CARRY	CLC	041	1	0.048	0.06	0.06	0.12	0.15	0.15	---
SELECT EM BANK	EMBC	281	2	14.0	14.0	15.1	15.1	---	---	---
EXTEND MAXIMUM CYCLE TIME	WDT	094	2	15.0	15.0	19.7	19.7	23.6	22.0	---
SAVE CONDITION FLAGS	CCS	282	1	8.6	8.6	12.5	---	14.2	12.9	---
LOAD CONDITION FLAGS	CCL	283	1	9.8	9.8	13.9	---	16.3	15.7	---
CONVERT ADDRESS FROM CV	FRMCV	284	3	13.6	13.6	19.9	---	23.1	31.8	---
CONVERT ADDRESS TO CV	TOCV	285	3	11.9	11.9	17.2	---	22.5	31.4	---
DISABLE PERIPHERAL SERVICING	IOSP	287	---	13.9	13.9	19.8	---	21.5	21.5	---
ENABLE PERIPHERAL SERVICING	IORS	288	---	63.6	63.6	92.3	---	22.2	22.2	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-30 Block Programming Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□ H-R	CPU6□ H	CPU4□ H	CPU4□	CJ1M excluding CPU11 /21	CJ1M CPU11 /21	
BLOCK PROGRAM BEGIN	BPRG	096	2	12.1	12.1	13.0	13.0	27.5	30.4	---
BLOCK PROGRAM END	BEND	801	1	9.6	9.6	12.3	13.1	23.2	27.1	---
BLOCK PROGRAM PAUSE	BPPS	811	2	10.6	10.6	12.3	14.9	16.0	21.7	---
BLOCK PROGRAM RESTART	BPRS	812	2	5.1	5.1	5.6	8.3	9.0	10.2	---
CONDITIONAL BLOCK EXIT	(Execution condition) EXIT	806	1	10.0	10.0	11.3	12.9	23.8	26.0	EXIT condition satisfied
				4.0	4.0	4.9	7.3	7.2	8.4	EXIT condition not satisfied
CONDITIONAL BLOCK EXIT	EXIT (bit address)	806	2	6.8	6.8	13.5	16.3	28.4	30.6	EXIT condition satisfied
				4.7	4.7	7.2	10.7	11.4	13.1	EXIT condition not satisfied

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
CONDITIONAL BLOCK EXIT (NOT)	EXIT NOT (bit address)	806	2	12.4	12.4	14.0	16.8	28.4	31.2	EXIT condition satisfied
				7.1	7.1	7.6	11.2	11.8	13.5	EXIT condition not satisfied
Branching	IF (execution condition)	802	1	4.6	4.6	4.8	7.2	6.8	8.5	IF true
				6.7	6.7	7.3	10.9	12.2	13.9	IF false
Branching	IF (relay number)	802	2	6.8	6.8	7.2	10.4	11.0	12.7	IF true
				9.0	9.0	9.6	14.2	16.5	18.5	IF false
Branching (NOT)	IF NOT (relay number)	802	2	7.1	7.1	7.6	10.9	11.5	13.1	IF true
				9.2	9.2	10.1	14.7	16.8	18.9	IF false
Branching	ELSE	803	1	6.2	6.2	6.7	9.9	11.4	12.6	IF true
				6.8	6.8	7.7	11.2	13.4	15.0	IF false
Branching	IEND	804	1	6.9	6.9	7.7	11.0	13.5	15.4	IF true
				4.4	4.4	4.6	7.0	6.93	8.1	IF false
ONE CYCLE AND WAIT	WAIT (execution condition)	805	1	12.6	12.6	13.7	16.7	28.6	34.0	WAIT condition satisfied
				3.9	3.9	4.1	6.3	5.6	6.9	WAIT condition not satisfied
ONE CYCLE AND WAIT	WAIT (relay number)	805	2	12.0	12.0	13.4	16.5	27.2	30.0	WAIT condition satisfied
				6.1	6.1	6.5	9.6	10.0	11.4	WAIT condition not satisfied
ONE CYCLE AND WAIT (NOT)	WAIT NOT (relay number)	805	2	12.2	12.2	13.8	17.0	27.8	30.6	WAIT condition satisfied
				6.4	6.4	6.9	10.1	10.5	11.8	WAIT condition not satisfied
COUNTER WAIT	CNTW	814	4	17.9	17.9	22.6	27.4	41.0	43.5	First execution
				19.1	19.1	23.9	28.7	42.9	45.7	Normal execution
	CNTWX	818	4	17.9	17.9	22.6	---	41.0	43.5	First execution
				19.1	19.1	23.9	---	42.9	45.7	Normal execution
TEN-MS TIMER WAIT	TMHW	815	3	25.8	25.8	27.9	34.1	47.9	53.7	First execution
				20.6	20.6	22.7	28.9	40.9	46.2	Normal execution
	TMHWX	817	3	25.8	25.8	27.9	---	47.9	53.7	First execution
				20.6	20.6	22.7	---	40.9	46.2	Normal execution
Loop Control	LOOP	809	1	7.9	7.9	9.1	12.3	15.6	17.6	---
Loop Control	LEND (execution condition)	810	1	7.7	7.7	8.4	10.9	13.5	15.5	LEND condition satisfied
				6.8	6.8	8.0	9.8	17.5	19.8	LEND condition not satisfied

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
Loop Control	LEND (relay number)	810	2	9.9	9.9	10.7	14.4	17.5	19.9	LEND condition satisfied
				8.9	8.9	10.3	13.0	21.6	24.5	LEND condition not satisfied
Loop Control	LEND NOT (relay number)	810	2	10.2	10.2	11.2	14.8	21.9	24.9	LEND condition satisfied
				9.3	9.3	10.8	13.5	17.8	20.4	LEND condition not satisfied
HUNDRED-MS TIMER WAIT	TIMW	813	3	22.3	22.3	25.2	33.1	47.4	52.0	Default setting
				24.9	24.9	27.8	35.7	46.2	53.4	Normal execution
	TIMWX	816	3	22.3	22.3	25.2	33.1	47.4	52.0	Default setting
				24.9	24.9	27.8	35.7	46.2	53.4	Normal execution

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-31 Text String Processing Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
MOV STRING	MOV$	664	3	45.6	45.6	66.0	84.3	79.3	72.7	Transferring 1 character
CONCATENATE STRING	+$	656	4	86.5	86.5	126.0	167.8	152.0	137.0	1 character + 1 character
GET STRING LEFT	LEFT$	652	4	53.0	53.0	77.4	94.3	93.6	84.8	Retrieving 1 character from 2 characters
GET STRING RIGHT	RGHT$	653	4	52.2	52.2	76.3	94.2	92.1	83.3	Retrieving 1 character from 2 characters
GET STRING MIDDLE	MID$	654	5	56.5	56.5	84.6	230.2	93.7	84.0	Retrieving 1 character from 3 characters
FIND IN STRING	FIND$	660	4	51.4	51.4	77.5	94.1	89.1	96.7	Searching for 1 character from 2 characters
STRING LENGTH	LEN$	650	3	19.8	19.8	28.9	33.4	33.8	30.1	Detecting 1 character
REPLACE IN STRING	RPLC$	661	6	175.1	175.1	258.7	479.5	300.7	267.7	Replacing the first of 2 characters with 1 character
DELETE STRING	DEL$	658	5	63.4	63.4	94.2	244.6	11.3	99.3	Deleting the leading character of 2 characters

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
EXCHANGE STRING	XCHG$	665	3	60.6	60.6	87.2	99.0	105.2	95.3	Exchanging 1 character with 1 character
CLEAR STRING	CLR$	666	2	23.8	23.8	36.0	37.8	42.0	36.8	Clearing 1 character
INSERT INTO STRING	INS$	657	5	136.5	136.5	200.6	428.9	204.0	208.0	Inserting 1 character after the first of 2 characters
String Comparison Instructions	LD, AND, OR +=$	670	4	48.5	48.5	69.8	86.2	79.9	68.5	Comparing 1 character with 1 character
	LD, AND, OR +<>$	671								
	LD, AND, OR +<$	672								
	LD, AND, OR +>$	674								
	LD, AND, OR +>=$	675								

Note When a double-length operand is used, add 1 to the value shown in the length column in the following table.

4-2-32 Task Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
TASK ON	TKON	820	2	19.5	19.5	26.3	26.3	33.1	32.5	---
TASK OFF	TKOF	821	2	13.3	13.3	19.0	26.3	19.7	20.2	---

4-2-33 Model Conversion Instructions (CPU Unit Ver. 3.0 or later only)

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
BLOCK TRANSFER	XFERC	565	4	6.4	6.4	6.5	---	33.1	31.1	Transferring 1 word
				481.6	481.6	791.6	---	3,056.1	2,821.1	Transferring 1,000 words

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
SINGLE WORD DISTRIBUTE	DISTC	566	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	567	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
				2,052.3	2,052.3	2,097.5	---	8,310.1	7,821.1	Stack operation 1,000 word FIFO Read
MOVE BIT	MOVBC	568	4	4.5	4.5	4.88	---	28.1	22.1	---
BIT COUNTER	BCNTC	621	4	4.9	4.9	5	---	30.6	28.8	Counting 1 word
				1,252.4	1,252.4	1284.4	---	5,814.1	5,223.8	Counting 1,000 words

4-2-34 Special Function Block Instructions (CPU Unit Ver. 3.0 or Later Only)

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)						Conditions
				CPU6□H-R	CPU6□H	CPU4□H	CPU4□	CJ1M excluding CPU11/21	CJ1M CPU11/21	
GET VARIABLE ID	GETID	286	4	14	14	22.2	---	23.4	21.3	

4-2-35 Number of Function Block Program Steps (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 CS/CJ-series 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			CS/CJ-series 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 I/O variable (BOOL)	6 steps
		1-word I/O variable (INT, UINT, WORD)	6 steps
		2-word I/O variable (DINT, UDINT, DWORD, REAL)	6 steps
		4-word I/O variable (LINT, ULINT, LWORD, LREAL)	12 steps
p	Number of instruction steps in function block definition	The total number of instruction steps (same as standard user program) + 27 steps.	

Example:
Input variables with a 1-word data type (INT): 5
Output variables with a 1-word data type (INT): 5
Function block definition section: 100 steps
Number of steps for 1 instance = 57 + (5 + 5) × 6 steps + 100 steps + 27 steps = 244 steps

4-2-36 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 CJ-series PLCs.

Add the following value (n) to the program capacity (unit: words) of the previous PLCs for each instruction to obtain the program capacity (unit: steps) of the CJ-series PLCs.

CJ-series steps = "a" (words) of previous PLC + n			
Instructions	**Variations**	**Value of n when converting from C200HX/HG/HE to CJ Series**	**Value of n when converting from CV-series PLC or CVM1 to CJ 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 the previous PLC would be 2 words per instruction and that of the CJ-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 CV-series PLC would be 4 words per instruction and that of the CJ-series PLC would be 7 (4 + 3) steps.

4-2-37 Function Block Instance Execution Time (CPU Units with Unit Version 3.0 or Later)

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 CS/CJ-series 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	CS1H-CPU6□H CJ1H-CPU6□H	CS1G-CPU4□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 I/O variable (BOOL)	0.24 μs	0.4 μs	0.7 μs	1.0 μs
		1-word I/O variable (INT, UINT, WORD)	0.19 μs	0.3 μs	0.6 μs	0.8 μs
		2-word I/O variable (DINT, UDINT, DWORD, REAL)	0.19 μs	0.5 μs	0.8 μs	1.1 μs
		4-word I/O variable (LINT, ULINT, LWORD, LREAL)	0.38 μs	1.0 μs	1.6 μs	2.2 μs
C	Function block definition instruction execution time	Total instruction processing time (same as standard user program)				

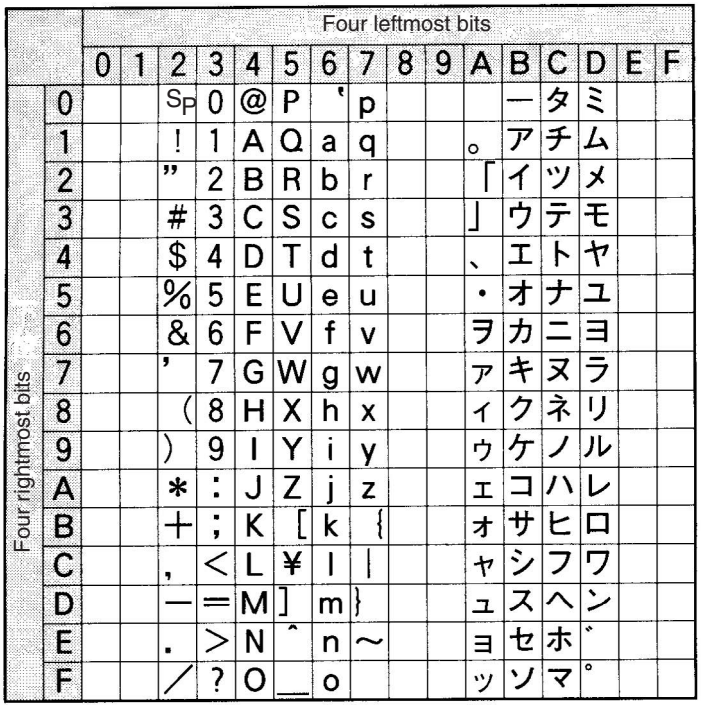
Example: CJ1H-CPU67H-R
Input variables with a 1-word data type (INT): 3
Output variables with a 1-word data type (INT): 2
Total instruction processing time in function block definition section: 10 μs
Execution time for 1 instance = 3.3 μs + (3 + 2) × 0.19 μs + 10 μs = 14.25 μs

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.

Appendix A

ASCII Code Table

ASCII



Four rightmost bits \ Four leftmost bits	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p				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	°		

Index

A

B

C

D

E

F–G

H

I

N

O

P

R

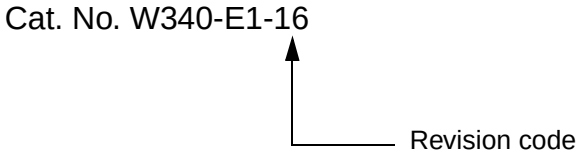
S

T

U–W

Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.



The following table outlines the changes made to the manual during each revision. Page numbers refer to the previous version.

Revision code	Date	Revised content
01	February 1999	Original production
02	October 1999	Revisions and additions for version–1 CPU Units. See page 118 for a list.
03	May 2000	Revisions and changes as follows: **Page xiii:** Precaution added. **Page 8:** Note removed. **Pages 162, 166, 177, 180, 183, 189, 196, 198, 262, 531, 560, and 705:** Index registers removed from operand specifications. **Page 170:** Sentence starting “An error will occur if a JMP0(515)” removed. **Pages 178, 181, and 184:** Precaution on timer numbers added and precaution on use in program jumps changed. **Page 181:** Precaution on refreshing Completion Flag added. **Pages 179, 182, 184:** Precaution on refreshing changed. **Page 554:** Parenthetic information removed from precaution. **Pages 576, 577, 579, 581, and 583:** Description changed to include CS1W–INT01. **Page 578:** Note added on using CLI with MSKS. **Pages 578 and 583:** Interrupt priority precaution changed. **Pages 639, 647, 651, and 655:** Serial port designation changed. **Page 642:** Manual reference added. **Page 675:** Information on file structure added. **Page 709:** Precaution added on long cycle times.
04	November 2000	Revisions and changes as follows: **Pages 169 and 170:** Precaution related to the cycle time deleted. **Pages 176, 180, 183, 186, 196, 199, 743, 746, and 749:** Timer number, counter number, and set value indications corrected. **Pages 189 and 192:** PV and SV range indications corrected. **Pages 209 and 210:** Ladder program modified and caution deleted. **Page 342:** Description about the CLEAR CARRY instruction deleted from precautions. **Page 395:** ON condition of Error Flag rewritten. **Page 531:** PID constant update timing designation added to the diagram. **Pages 533 and 534:** Description on PID added to the end of description and example. **Page 536:** Bit 01 of C+5 added to the table. **Pages 567, 572, 730, 732, 788, and 791:** Note under the flags table deleted. **Page 580:** Note 1 at the top of the page changed. **Page 613:** CIO addresses changed. **Page 704:** FAL numbers in operands table changed.
05	May 2001	Name of manual changed, “CS1 Series” changed to “CS Series” or “CS/CJ Series,” CJ-series PCs added, and “CS Series only” added to specified restricted functions. Other changes and additions for the above were made to the following pages: xv, 2, 661, 667, 678, **Page 116:** Section 3-2 removed. **Pages 589, 590, 594, and 595:** Information added for S and D. **Page 598:** Headings changed.
06	October 2001	New products added to the manual, including the new High-speed CPU Units (CS1-H and CJ1-H CPU Units) and the new instructions they support. (Extensive changes too numerous to list.)
06A	February 2002	**Page 666:** Bit specifications in *Control data* column for Bits 04 to 07 of C+6 and Bits 00 to 03 of C+6 reversed.

Revision code	Date	Revised content
07	July 2002	Manual revised to add CJ1M CPU Units and the new instructions that they support (including support for binary refreshing for timer/counter PV). (Extensive changes too numerous to list.) New timer and counter instructions added: TIMX, TIMHX, TMHHX, TTIMX, TIMLX, MTIMX, CNTX, CNTRX, and CNRX. BCMP2 added. "PC" changed globally to "PLC" when the meaning is Programmable Controller. **Page x:** Manual added and product versions updated. **Pages 379 and 389:** Example programming changed. **Page 489:** Less than symbol changed to less than or equals symbol. **Page 490:** Graphic changed. **Page 628:** Operand changed in example and note added to example. **Pages 648 and 651:** First entry for error flag changed. **Page 666:** Bit numbers corrected in table. **Page 701:** Graphic for R+1 changed. **Pages 728 to 748:** Instructions reworked. **Pages 787, 814, 816 to 832:** Information added on automatic port allocation. **Pages 820 and 825:** Precautions added. **Page 833:** Precautions on using Memory Cards added. **Page 873:** Bottom half of page modified.
08	September 2002	Manual revised to add CS1D CPU Units. The following changes were also made. **Page xiii:** Caution added. **Pages xiv to xviii:** Application Precautions replaced with same section from *Programming Manual.* **Page 4:** Description of the operation of immediate refreshing changed. **Page 9:** Data types added. **Pages 222 and 225:** "Do not use" added to graphic. **Page 683:** Ramp response graphic corrected.
09	June 2003	**Pages 10 and 11:** Note with examples added on instructions executable when input conditions are OFF. **Page 24:** Table updated and note added for instructions not supported by CS1D CPU Units and CS1 CPU Units with -V1 suffix. **Pages 26 to 28:** Table updated and note added for instructions not supported by CS1D CPU Units. **Pages 36 and 37:** Table updated and note added for instructions not supported by CS1D CPU Units. **Pages 144, 148, and 152:** Tables updated and notes added for new CPU Unit models. **Page 233:** Note added with information on adding counters using online editing. **Page 293:** Information on condition of first destination word removed. **Page 679:** Information added to graphic. **Pages 681 and 691:** Terms added to table to clarify meaning of parameter settings. **Page 692:** Bit numbers corrected (swapped) for output range and integral and derivative unit. **Page 710:** Information on outputting negative values in scaling results changed. **Page 781:** Error Flag conditions added to table. **Page 791:** Information added to note on executing PLS2(887). **Page 794:** Corrections made to table. **Page 797:** Information added to note on executing PLS2(887). **Page 824:** Ladder programming corrected for process B. **Page 831:** "I/O Unit's" corrected to "Special I/O Unit's." **Pages 844 and 845:** Information on first send and read words/addresses changed. **Page 894:** Reference manual changed. **Page 899:** Information on data file structure from page 912 of previous manual moved to this page. **Page :** Information on data file structure from pages 912 to 913 of previous manual moved to this page. **Page 1110:** ASCII code table from page 916 added.

Revision code	Date	Revised content
10	December 2003	Information added on functions supported by new unit versions of CPU Units (too numerous to list). **Pages xi to xx:** PLP information updated.
11	July 2004	Manual revised for CPU Unit Ver. 3.0 and the new instructions that are supported. (Extensive changes too numerous to list.) New instructions: TXDU, RXDU, XFERC, DISTC, COLLC, MOVBC, BCNTC, and GETID Revised instructions: TXD, RXD, PRV, PRV2, network instructions CPU Unit added: CJ1H-CPU67H The following corrections and changes were also made. **Page 99:** Function codes corrected for CNTWX and TWHWX. **Pages 183 and 229:** Precautions added. **Page 271:** Mnemonics corrected in table. **Page 428:** Heading corrected. **Page 676:** Precaution replaced. **Page 677:** Record numbers corrected. **Page 857:** Port specifier table replaced.
12	January 2006	**Page v:** Information on general precautions notation added. **Page xxiii:** Information on liability and warranty added.
13	September 2006	New instructions: STR4, STR8, STR16, NUM4, NUM8, NUM16, and TWRIT.
14	April 2007	Information was added on the CJ1H-CPU□□H-R CPU Units.
15	January 2008	Added information on unit version 4.1 of the CJ1H-CPU□□H-R CPU Units (CJ1-H-R). **Page 244:** Changes made in five cells in top table and note added under top table. **Pages 245 and 253:** Information added to note on timer accuracy and information added to note on timer numbers. **Page 254:** Information added to note on timer numbers. **Page 255:** Third paragraph of *Precautions* and last paragraph on page changed. **Page 256:** Information added on timer numbers. **Page 342:** Description of Control Word corrected, including callouts. **Page 575:** “Signed” corrected to “unsigned” in note in top figure. **Page 578:** Third precaution corrected. **Page 579:** Figure added. **Page 764:** Sentence removed from description of output range. **Page 997:** Sentence starting “If more data is received” deleted and last part of last sentence on page deleted. **Page 999:** One bulleted item in note deleted and one added. **Pages 1208 and 1215:** Precaution added.
16	August 2008	**Page x:** Added unit version 4.2. **Pages xviii:** Changed note 2. **Page xxii:** Added the CJ2 CPU Units. **Pages xxx and xxxi:** Changed name of W446, W447, W464, and W463, and removed version number from description of W447. **Pages 246 and 254:** Changed note. **Page 257:** Changed information before first table. **Pages 761, 774, and 815:** Added precaution. **Pages 875 and 878:** Added a precautions section. **Pages 945, 949, 954, and 958:** Added information to precaution on I/O refreshing. **Pages 986 and 998:** Added paragraph to description of operation of Error Flag. **Pages 987 and 1000:** Added paragraph to list of reasons for Error Flag turning ON. **Page 1017:** Added information to paragraph starting "If an end code is specified." **Page 1279:** Removed rows for Auxiliary Bit Area and current bank of EM Area from the second table.

OMRON Corporation
Industrial Automation Company
Control Devices Division H.Q.
PLC Division
Shiokoji Horikawa, Shimogyo-ku,
Kyoto, 600-8530 Japan
Tel: (81) 75-344-7084/Fax: (81) 75-344-7149

Regional Headquarters
OMRON EUROPE B.V.
Wegalaan 67-69-2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC
One Commerce Drive Schaumburg,
IL 60173-5302 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark, Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

Printed in Japan
OMRON Industrial Automation Global: **www.ia.omron.com**
Cat. No. W340-E1-16
0808