Cat. No. W394-E1-12

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

PROGRAMMING 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

Programming Manual

Revised January 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.

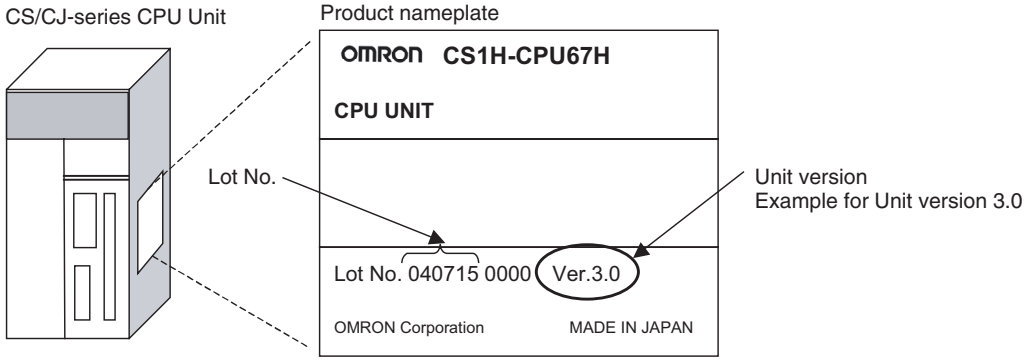
Unit Versions of CS/CJ-series CPU Units

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.

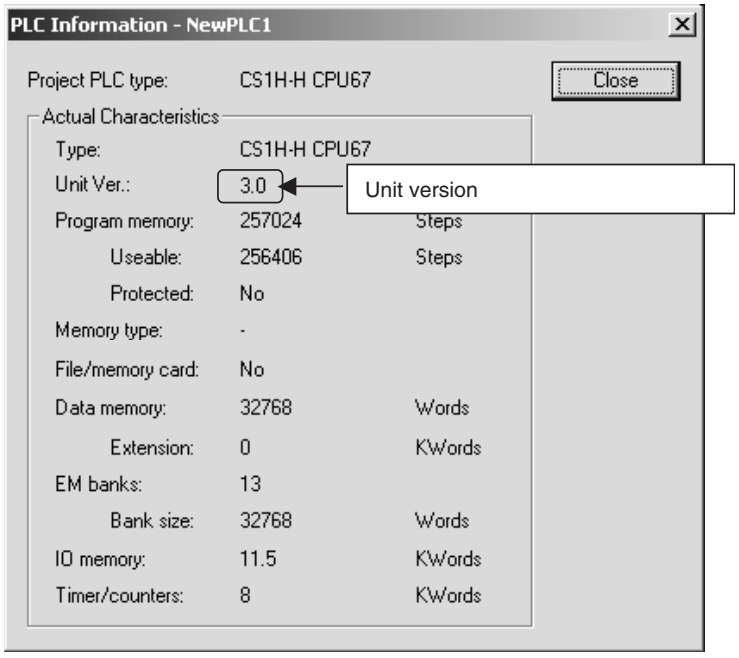
- Using the ***PLC Information***
- Using the ***Unit Manufacturing Information*** (This method can be used for Special I/O Units and CPU Bus Units as well.)

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.

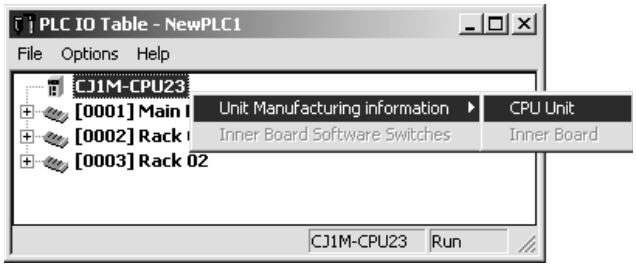
In either case, the following *PLC Information* Dialog Box will be displayed.



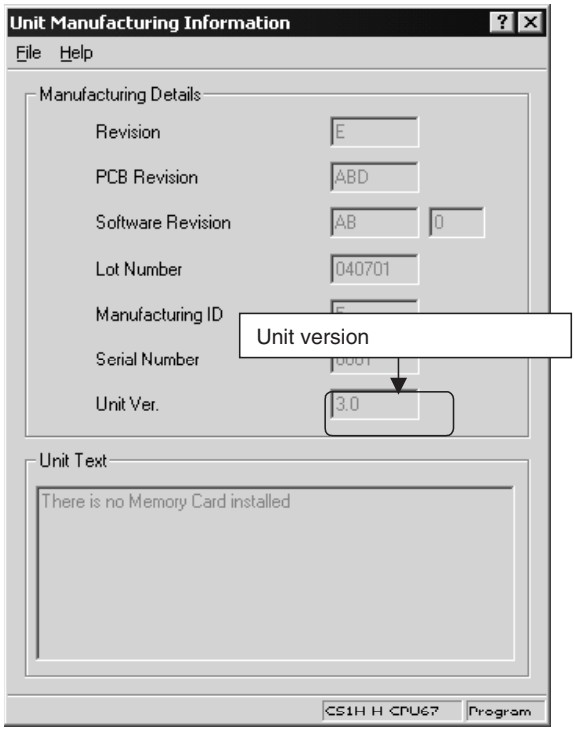
Use the above display to confirm the unit version of the CPU Unit.

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.

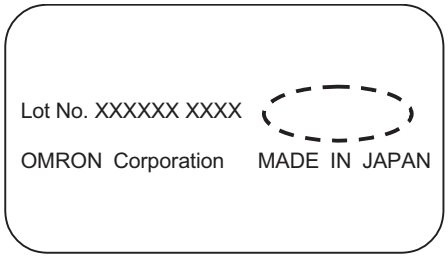
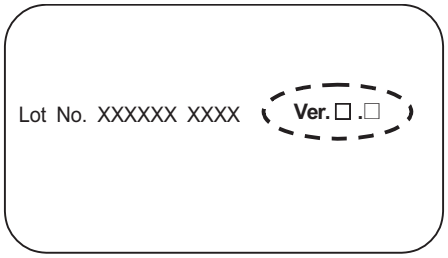
Using the Unit Version Labels

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 4.1
		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 4.1
		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.
Additional functions are supported if CX-Programmer version 7.2 or higher is used.

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	---
ST programming in task programs		OK with CX-Programmer version 7.2 or higher	---
SFC programming in task programs		OK with CX-Programmer version 7.2 or higher	---

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	---
ST programming in task programs		OK with CX-Programmer version 7.2 or higher	---
SFC programming in task programs		OK with CX-Programmer version 7.2 or higher	---

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 (CJ1□-CPU□□H)		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 notes 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 of the CJ1-H-R CPU Units. CX-Programmer version 7.22 or higher is required to use unit version 4.1 of the CJ1-H-R CPU Units. You can check the CX-Programmer version using the ***About*** menu command to display version information.
3. CX-Programmer version 7.0 or higher is required to use the functional improvements made for unit version 4.0 of the CS/CJ-series CPU Units. With CX-Programmer version 7.2 or higher, you can use even more expanded functionality.

Device Type Setting

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

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

Note When using a CJ1H-CPU□□H-R CPU Unit, set the CPU Unit model to CPU67-R, CPU66-R, CPU65-R, or CPU64-R.

Troubleshooting Problems with Unit Versions on the CX-Programmer

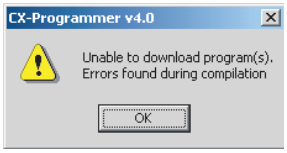
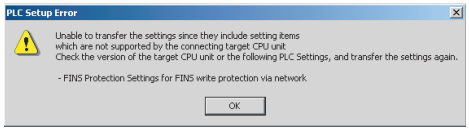
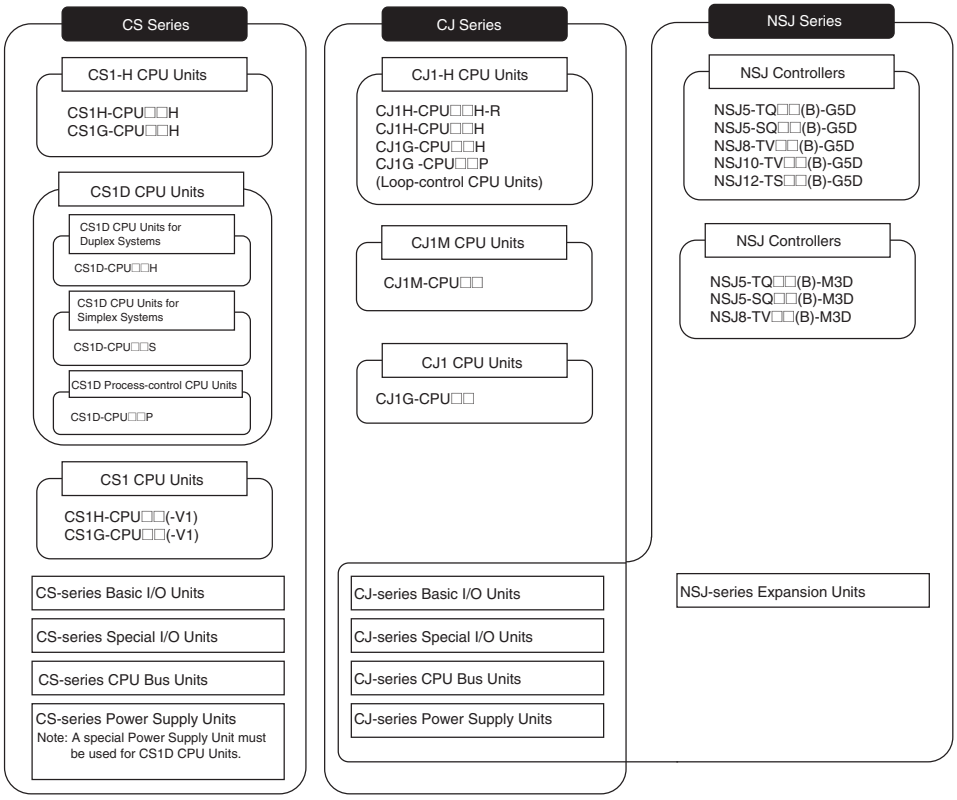
Problem	Cause	Solution
 After the above message is displayed, a compiling error will be displayed on the *Compile* Tab Page in the Output Window.	An attempt was made to download a program containing instructions supported only by later unit versions or a CPU Unit to a previous unit version.	Check the 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 programming of the CPU Units for CS/CJ-series Programmable Controllers (PLCs) and includes the sections described on the following page. The CS Series, CJ Series and NSJ Series are subdivided as shown in the following figure.



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 install or use CS/CJ-series CPU Units in a PLC System.

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 with unit version 3.0
NSJ□-TQ□□(B)-M3D	CJ1G-CPU45H 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 Racks max.	1 Rack max.
EM Area		32 Kwords × 3 banks E0_00000 - E2_32767	None
Function blocks	Max. No. of definitions	1024	128
	Max. No. of instances	2048	256

Item		CJ-series CPU Unit CJ1G-CPU45H	Controller Section in NSJ□-□□□□(B)-M3D
Capacity in built-in file memory	FB program memory	1024 KB	256 KB
	Variable tables	128 KB	64 KB

This manual contains the following sections.

Section 1 describes the basic structure and operation of the CPU Unit.

Section 2 describes basic information required to write, check, and input programs.

Section 3 outlines the instructions that can be used to write user programs.

Section 4 the operation of tasks.

Section 5 describes the functions used to manipulate file memory.

Section 6 provides details on the following advanced functions: cycle time/high-speed processing functions, index register functions, serial communications functions, startup and maintenance functions, diagnostic and debugging functions, Programming Device functions, and the Basic I/O Unit input response time settings.

Section 7 describes the processes used to transfer the program to the CPU Unit and the functions that can be used to test and debug the program.

The ***Appendices*** provide a comparison of CS/CJ-series, restrictions in using C200H Special I/O Units, and changes made to Host Link Systems.

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 Programming Manual	W394 (This manual)	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/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-series PLCs.
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 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.
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)

Name	Cat. No.	Contents
SYSMAC WS02-CXPC1-E-V7 CX-Programmer Operation Manual	W446	Provides information on how to use the CX-Programmer for all functionality except for function blocks.
SYSMAC WS02-CXP□□-E CX-Programmer Operation Manual: Function Blocks (CS1G-CPU□□H, CS1H-CPU□□H, CJ1G-CPU□□H, CJ1H-CPU□□H, CJ1M-CPU□□, CP1H-X□□□□-□, CP1H-XA□□□□-□, and CP1H-Y□□□□-□ CPU Units)	W447	Describes specifications and operation methods related to function blocks. This information is required only when using function blocks.
SYSMAC CS/CJ Series Programming Consoles Operation Manual CQM1H-PRO01-E, CQM1-PRO01-E, C200H-PRO27-E	W341	Provides information on how to program and operate CS/CJ-series PLCs using a Programming Console. When programming, use this manual together with the Programmable Controllers Operation Manual (W339 for CS-series PLCs and W393 for CJ-series PLCs), CS/CJ-series Programmable Controllers Programming Manual (W394,) and the CS/CJ-series Programmable Controllers Instructions Reference Manual (W340).
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 use of standard system protocols for OMRON products. Refer to the CS/CJ Series Communications Commands Reference Manual (W342) for details on sending commands in host link mode from a Serial Communications Board or Unit's port. Refer to the WS02-PSTC1-E CX-Protocol Operation Manual (W344) for details on creating protocol macros.
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□□CEV2/AL□□D-EV2 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□□CEV2/AL□□D-EV2 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.

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. Abnormal operation may result in serious accidents.

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

⚠**Caution** Confirm safety at the destination node before transferring a program to another node or 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 required 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.

廃電池請回收

- 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):
CS Series: EN61131-2 and EN61000-6-2
CJ Series: EN61000-6-2
EMI (Electromagnetic Interference):
EN61000-6-4
(Radiated emission: 10-m regulations)

Low Voltage Directive

Always ensure that devices operating at voltages of 50 to 1,000 V AC and 75 to 1,500 V DC meet the required safety standards for the 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 connected to DC Power Supply Units and I/O Units.
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

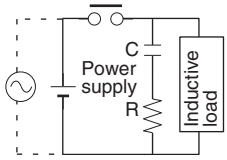
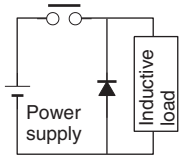
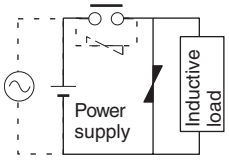
(Refer to EN61000-6-4 for more details.)

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.

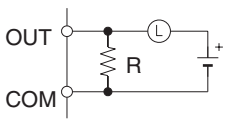
Countermeasure Examples

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.	---

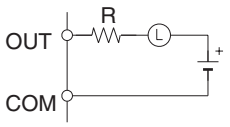
When switching a load with a high inrush current such as an incandescent lamp, suppress the inrush current as shown below.

Countermeasure 1



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

Countermeasure 2



Providing a limiting resistor

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.

SECTION 1
CPU Unit Operation

This section describes the basic structure and operation of the CPU Unit.

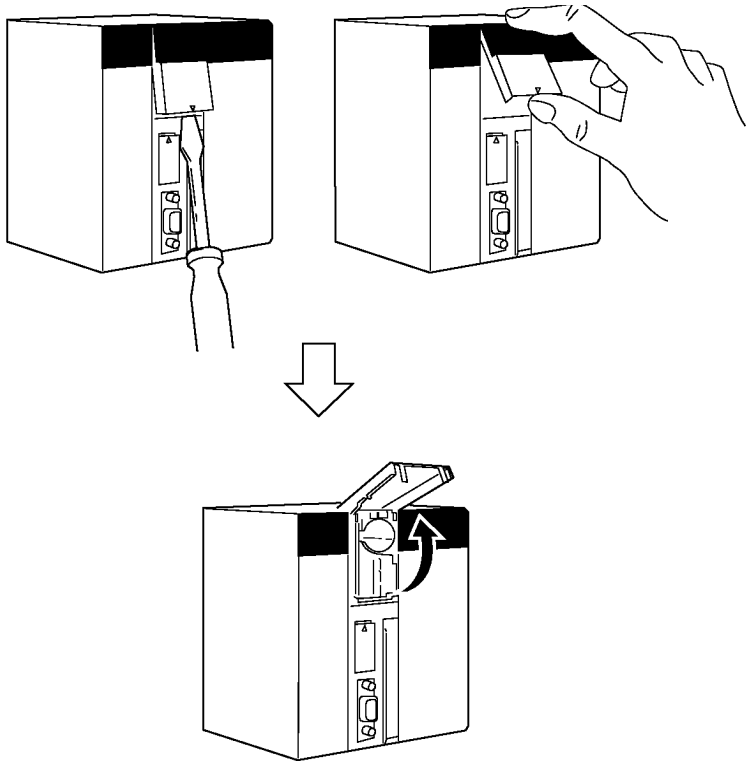
1-1 Initial Setup (CS1 CPU Units Only)

Battery Installation

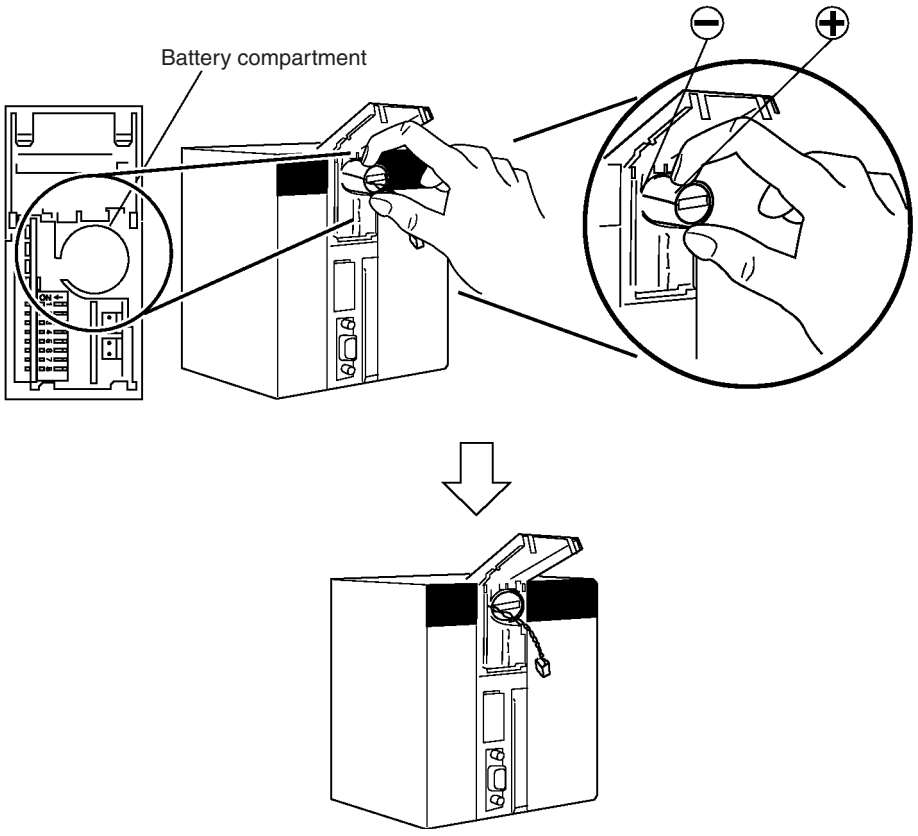
Before using a CS1CPU Unit, you must install the Battery Set in the CPU Unit using the following procedure.

1,2,3...

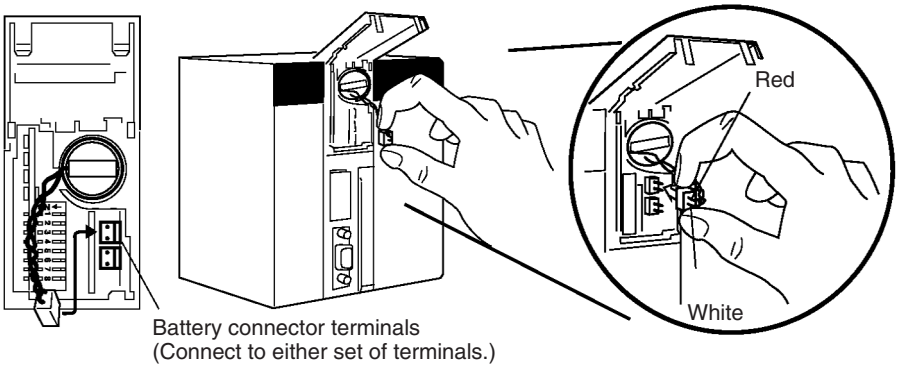
1. Insert a flat-blade screwdriver in the small gap at the bottom of the battery compartment and flip the cover upward to open it.



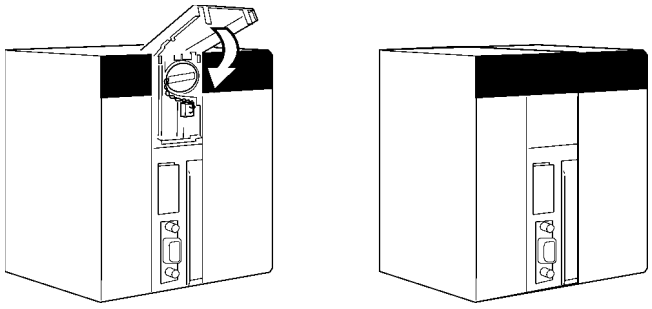
2. Hold the Battery Set with the cable facing outward and insert it into the battery compartment.



3. Connect the battery connector to the battery connector terminals. Connect the red wire to the top and the white wire to the bottom terminal. There are two sets of battery connector terminals; connect the battery to either one. It does not matter whether the top terminals or bottom terminals are used.



4. Fold in the cable and close the cover.



Clearing Memory

After installing the battery, clear memory using the memory clear operation to initialize the RAM inside the CPU Unit.

Programming Console

Use the following procedure from a Programming Console.

Note You cannot specify more than one cyclic task when clearing memory from a Programming Console. You can specify one cyclic task and one interrupt task, or one cyclic task and no interrupt task. Refer to the *Operation Manual* for more information on the memory clear operation. Refer to *SECTION 1 CPU Unit Operation* and *SECTION 4 Tasks* for more information on tasks.

CX-Programmer

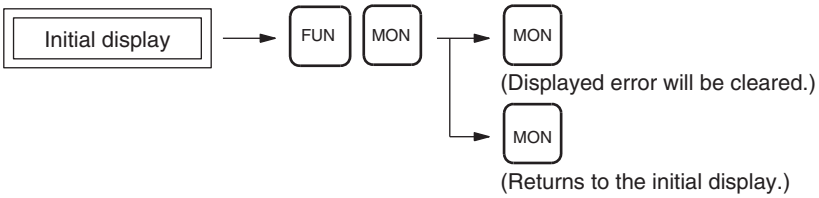
Memory can also be cleared from the CX-Programmer. Refer to the *CX-Programmer Operation Manual* for the actual procedure.

Clearing Errors

After clearing memory, clear any errors from the CPU Unit, including the low battery voltage error.

Programming Console

Use the following procedure from a Programming Console.



CX-Programmer

Errors can also be cleared from the CX-Programmer. Refer to the *CX-Programmer Operation Manual* for the actual procedure.

Note When an Inner Board is mounted, an Inner Board routing table error may continue even after you have cancelled the error using the CX-Programmer. (A42407 will be ON for a Serial Communications Board.) If this occurs, either reset the power or restart the Inner Board, then cancel the error again.

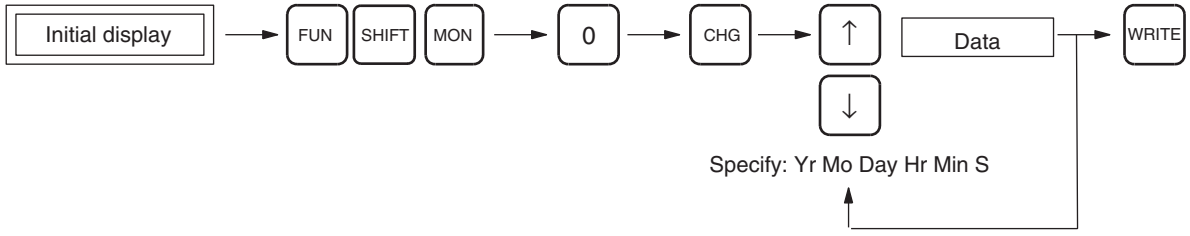
1-2 Using the Internal Clock (CS1 CPU Units Only)

The internal clock of the CPU Unit is set to “00 year, 01 month, 01 day (00-01-01), 00 hours, 00 minutes, 00 seconds (00:00:00), and Sunday (SUN)” when the Battery Set is mounted in the CS-series CPU Unit.

When using the internal clock, turn ON the power supply after mounting the Battery Set and 1) use a Programming Device (Programming Console or CX-Programmer) to set the clock time, 2) execute the CLOCK ADJUSTMENT (DATE) instruction, or 3) send a FINS command to start the internal clock from the correct current time and date.

The Programming Console operation used to set the internal clock is shown below.

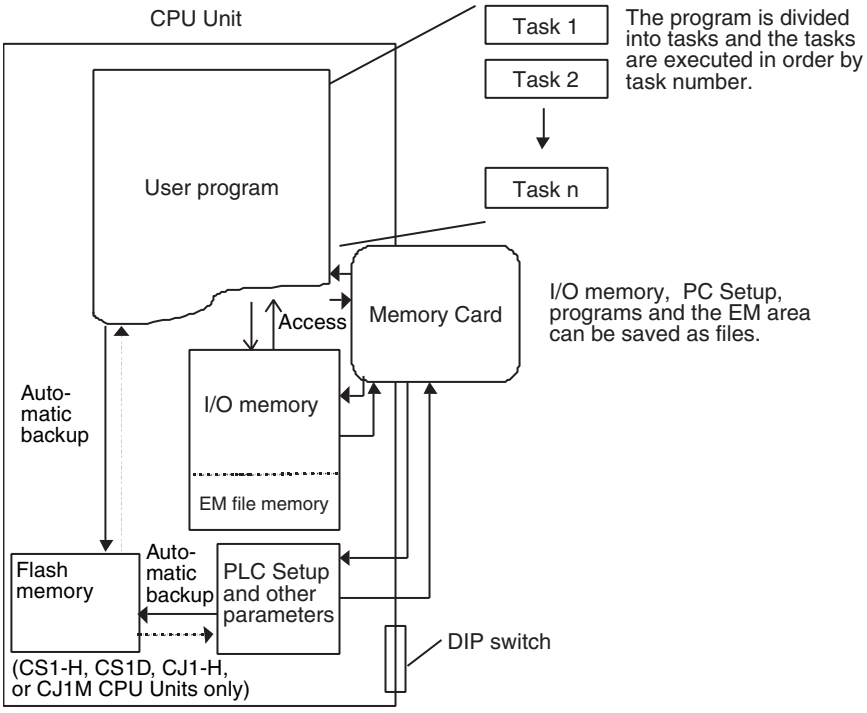
Key Sequence



1-3 Internal Structure of the CPU Unit

1-3-1 Overview

The following diagram shows the internal structure of the CPU Unit.



The User Program

The user program is created from up to 288 program tasks, including interrupt tasks. The tasks are transferred to the CPU Unit from the CX-Programmer programming software.

There are two types of tasks. The first is a cyclic task that is executed once per cycle (maximum of 32) and the other is an interrupt task that is executed only when the interrupt conditions occur (maximum of 256). Cyclic tasks are executed in numerical order.

Interrupt tasks can also be executed cyclically, like cyclic tasks. In this case, the total number of tasks that can be executed cyclically is 288.

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units are not equipped with a function to execute interrupt tasks like cyclic tasks.

Program instructions read and write to I/O memory and are executed in order starting at the top of the program. After all cyclic tasks are executed, the I/O for all Units are refreshed, and the cycle repeats again starting at the lowest cyclic task number.

Refer to the section on CPU Unit operation in the *CS/CJ Series Operation Manual* for details on refreshing I/O.

I/O Memory

I/O memory is the RAM area used for reading and writing from the user program. It is comprised of one area that is cleared when power is turned ON and OFF, and another area that will retain data.

I/O memory is also partitioned into an area that exchanges data with all Units and an area strictly for internal use. Data is exchanged with all Units once per program execution cycle and also when specific instructions are executed.

PLC Setup The PLC Setup is used to set various initial or other settings through software switches.

DIP Switches DIP switches are used to set initial or other settings through hardware switches.

Memory Cards Memory Cards are used as needed to store data such as programs, I/O memory data, the PLC Setup, and I/O comments created by Programming Devices. Programs and various system settings can be written automatically from the Memory Card when power is turned ON (automatic transfer at startup).

Flash Memory The user program and parameter area data, such as the PLC Setup, are automatically backed up in the built-in flash memory whenever the user writes data to the CPU Unit. This enables battery-free operation without using a Memory Card. I/O memory, including most of the DM Area, are not backed up without a battery.

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units are not equipped with the flash memory functions and the data is backed up by the battery. Consequently, the data is not backed up when a battery is not installed.

1-3-2 Block Diagram of CPU Unit Memory

CPU Unit memory (RAM) is comprised of the following blocks in the CS/CJ Series:

Area name	Contents	Backup method	Status when battery voltage drops
User Program Area	User program	Internal flash memory	Data is not lost.
Parameter Area	PLC Setup, Registered I/O Tables, Routing Tables, and CPU Bus Unit settings	Battery and internal flash memory	Data is not lost when the PLC Setup's *Detect Low Battery* setting is set to *Do not detect.*
I/O Memory Areas	Data areas such as the HR, DM, and EM Areas	Battery	Data is not lost.
Comment Memory Area	Comment information	Internal flash memory	Data is not lost.
Source Memory Area	Function block information	Internal flash memory	Data is not lost.

User Program Area

The CPU Unit backs up this data in internal flash memory. The user program is automatically backed up in flash memory if it was overwritten by a program transfer from the CX-Programmer, online editing, or a transfer from the Memory Card. Even if the battery voltage drops, the user program data will not be lost.

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units are not equipped with the flash memory functions and the data is backed up by the battery. Consequently, the data is not backed up when a battery is not installed.

Parameter Area (PLC Setup, Registered I/O Tables, Routing Tables, and CPU Bus Unit Settings)

If the PLC Setup's Detect Low Battery setting is set to *Do not detect*, this data is backed up in the internal flash memory. The data is automatically backed up in flash memory if the parameters are overwritten by an operation such as a

transfer from the CX-Programmer or a transfer from the Memory Card. If the PLC Setup's Detect Low Battery setting is set to *Do not detect*, the Parameter Area data will not be lost even if the battery voltage drops.

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units are not equipped with the flash memory functions and the data is backed up by the battery. Consequently, the data is not backed up when battery is not installed.

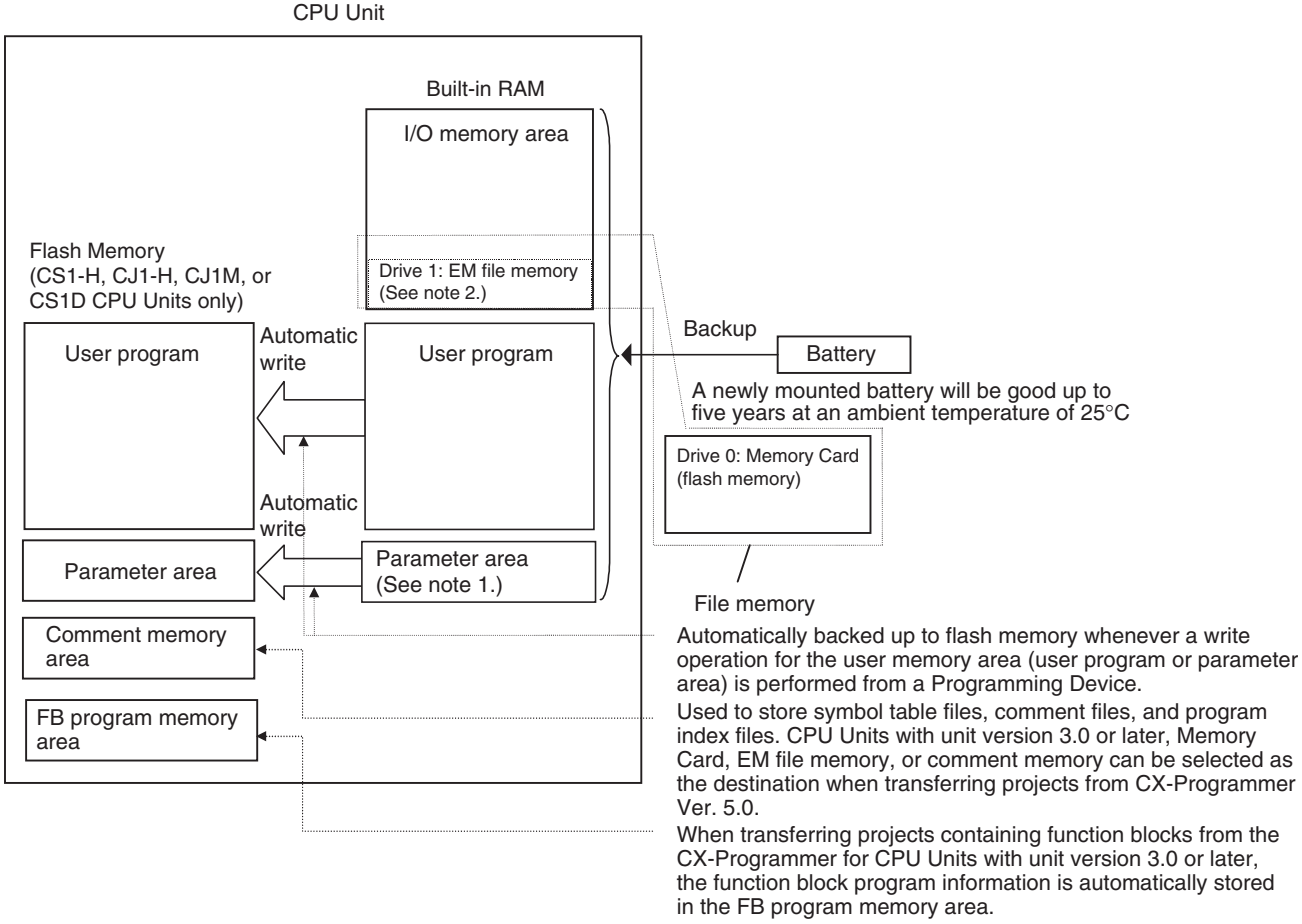
I/O Memory Areas

This data is backed up by the battery (CS1W-BAT01 in CS-series PLCs, CPM2A-BAT01 in CJ1H PLCs, or CJ1W-BAT01 in CJ1M PLCs). If the battery voltage drops, this data will be lost.

Comment Memory Area and Source Memory Area

The CPU Unit backs up this data in its internal flash memory. When a comment file or function block used in the program is transferred from the CX-Programmer, it is stored in flash memory. Even if the battery voltage drops, the data will not be lost.

Only CPU Units with unit version 3.0 or later are equipped with this memory.



Note

1. The BKUP indicator on the front of the CPU Unit will light while data is being written to flash memory. Do not turn OFF the power supply to the CPU Unit until the backup operation has been completed (i.e., until the BKUP indicator goes out). Refer to *6-6-11 Flash Memory* for details.
2. A Battery is mounted to a CS1-H, CJ1, CJ1-H, CJ1M, or CS1D CPU Unit when it is shipped from the factory. There is no need to clear memory or set the time.

3. Always install the battery provided (CS1W-BAT01) before using a CS1G/H-CPU□□ or CJ1□-CPU□□ CPU Unit for the first time. After installing the battery, use a Programming Device to clear the PLC's RAM (parameter area, I/O memory area, and user program).

1-4 Operating Modes

1-4-1 Description of Operating Modes

The following operating modes are available in the CPU Unit. These modes control the entire user program and are common to all tasks.

PROGRAM Mode

Program execution stops in PROGRAM mode, and the RUN indicator is not lit. This mode is used when editing the program or making other preparations operation, such as the following:

- Registering the I/O table.
- Changing PLC Setup and other settings.
- Transferring and checking programs.
- Force-setting and resetting bits to check wiring and bit allocation.

In this mode, all cyclic and interrupt tasks are non-executing (INI), that is they stop. See *1-6 Description of Tasks* for more details on tasks.
I/O refreshing is performed in PROGRAM mode. Refer to the *Operation Manual* for information on refreshing I/O.

⚠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.

MONITOR Mode

The following operations can be performed through Programming Devices while the program is executing in MONITOR mode. The RUN indicator will be lit. This mode is used to make test runs or other adjustments.

- Online Editing.
- Force-setting and force-resetting bits.
- Changing values in I/O memory.

In this mode, the cyclic tasks specified for execution at startup (see note) and those are made executable by TKON(820) will be executed when program execution reaches their task number. Interrupt tasks will be executed if their interrupt conditions occur.

Note The tasks that are executed at startup are specified in the program properties from the CX-Programmer.

RUN Mode

This mode is used for normal program execution. The RUN indicator will be lit. Some Programming Device operations like online editing, force-set/force-reset, and changing I/O memory values are disabled in this mode, but other Programming Device operations like monitoring the status of program execution (monitoring programs and monitoring I/O memory) are enabled.

Use this mode for normal system operation. Task execution is the same as in MONITOR mode.

See *10-2 CPU Unit Operating Modes* in the *Operation Manual* for more details on operations that are available in each operating mode.

1-4-2 Initialization of I/O Memory

The following table shows which data areas will be cleared when the operating mode is changed from PROGRAM mode to RUN/MONITOR mode or vice-versa.

Mode change	Non-held Areas (Note 1)	Held Areas (Note 2)
RUN/MONITOR → PROGRAM	Clear (See note 3.)	Retained
PROGRAM → RUN/MONITOR	Clear (See note 3.)	Retained
RUN ↔ MONITOR	Retained	Retained

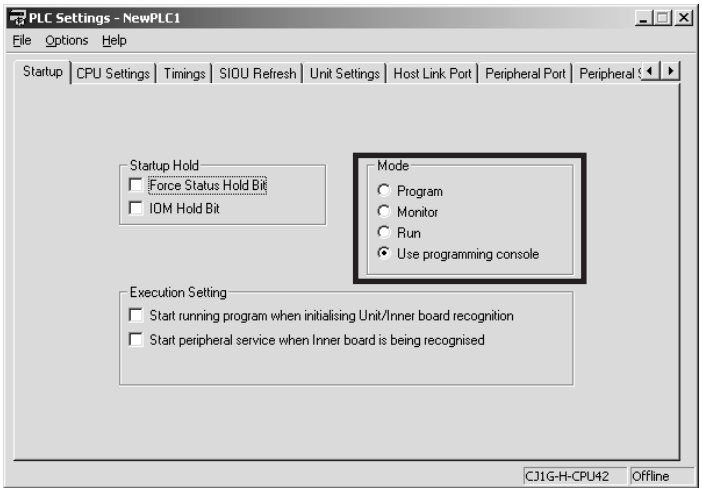
Note

1. Non-held areas: CIO Area, Work Area, Timer PVs, Timer Completion Flags, Index Registers, Data Registers, Task Flags, and Condition Flags. (The statuses of some addresses in the Auxiliary Area are held and others are cleared.)
2. Held areas: Holding Area, DM Area, EM Area, Counter PVs, and Counter Completion Flags.
3. Data in I/O memory will be retained when the IOM Hold Bit (A50012) is ON. When the IOM Hold Bit (A50012) is ON and operation is stopped due to a fatal error (including FALS(007)), the contents of I/O memory will be retained but outputs on Output Units will all be turned OFF.

1-4-3 Startup Mode

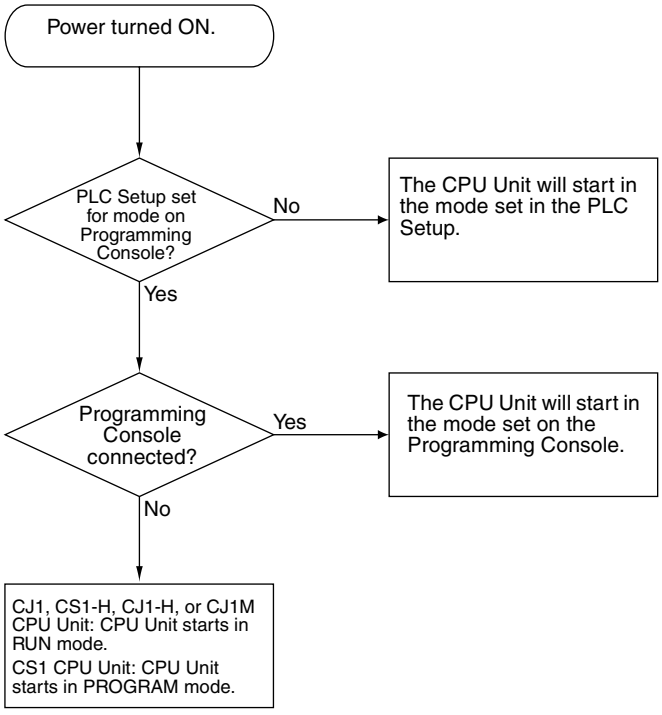
Refer to the *Operation Manual* for details on the Startup Mode setting for the CPU Unit.

Note The CPU Unit will start in RUN Mode if the PLC Setup's *Mode Setting* is set to *Use Programming Console* but a Programming Console is not connected.



Note With CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units, the CPU Unit will start in PROGRAM mode.

Conditions	CS1-H/CJ1-H/CJ1M/ CS1D CPU Unit	CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units
The PLC will start in the mode shown at the right if both of the following conditions are met. 1. The PLC Setup is set to start PLC operation in the mode set on the Programming Console (Use Programming Console), but a Programming Console is not connected. 2. A Programming Console is not connected.	RUN mode	PROGRAM mode


Power turned ON.
PLC Setup set for mode on Programming Console?
No
The CPU Unit will start in the mode set in the PLC Setup.
Yes
Programming Console connected?
Yes
The CPU Unit will start in the mode set on the Programming Console.
No
CJ1, CS1-H, CJ1-H, or CJ1M CPU Unit: CPU Unit starts in RUN mode.
CS1 CPU Unit: CPU Unit starts in PROGRAM mode.

1-5 Programs and Tasks

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

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

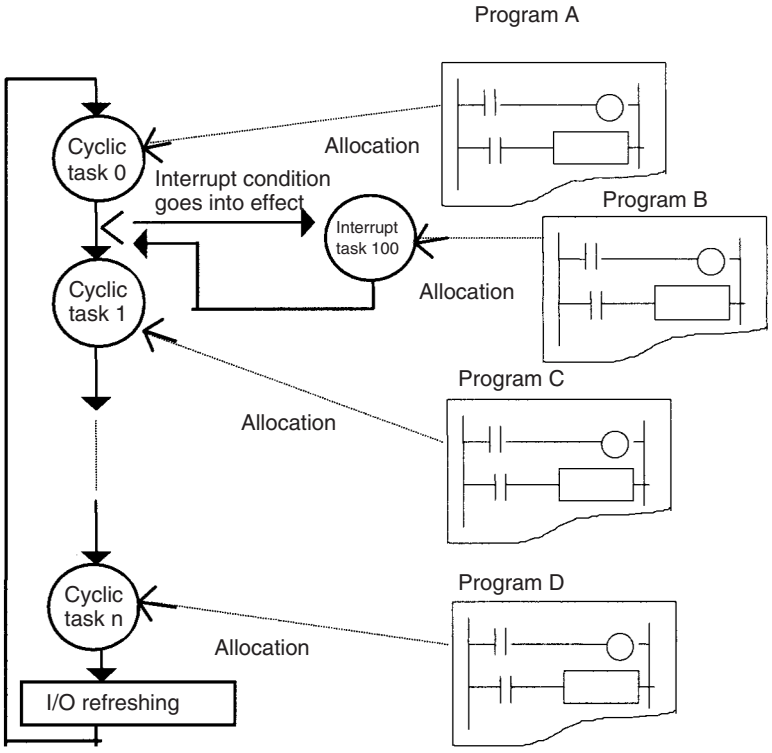
2. Tasks executed by interrupt conditions that are called interrupt tasks.

With the CS1-H, CJ1-H, CJ1M, or CS1D CPU Units, interrupt tasks can be executed cyclically in the same way as cyclic tasks. These are called “extra cyclic tasks.”

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support extra cyclic tasks.

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

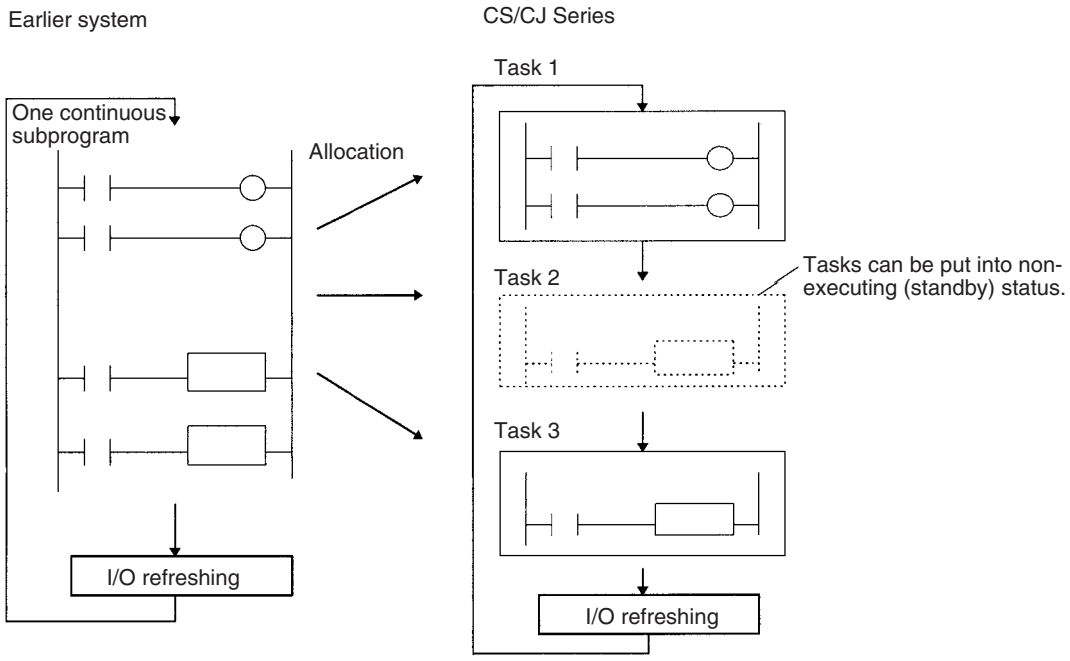
Refer to the section on CPU Unit operation in the *CS/CJ Series Operation Manual* for information in refreshing I/O.



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

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

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

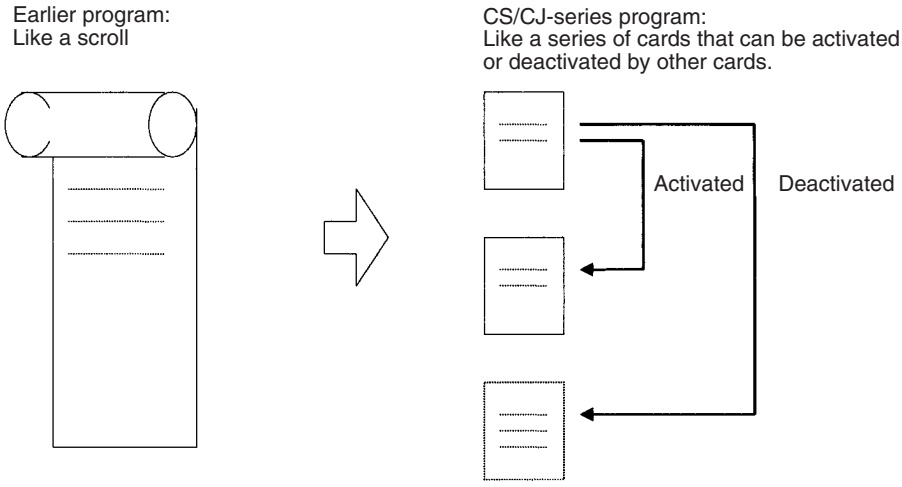


A task that has been executed will be executed in subsequent cycles, and a task that is on standby will remain on standby in subsequent cycles unless it is executed again from another task.

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

- All cards are read in a preset sequence starting from the lowest number.
- All cards are designated as either active or inactive, and cards that are inactive will be skipped. (Cards are activated or deactivated by task control instructions.)

- A card that is activated will remain activated and will be read in subsequent sequences. A card that is deactivated will remain deactivated and will be skipped until it is reactivated by another card.



1-6 Description of Tasks

Tasks are broadly grouped into the following types:

1,2,3... 1. Cyclic tasks (32 max.)

Tasks that will be executed once per cycle if executable. Execution can also be disabled for cyclic tasks if required.

2. Interrupt tasks (256 max.)

Tasks that are executed when the interrupt occurs whether or not a cyclic task is being executed. There are 5 kinds of interrupt tasks, as described below:

Note The CS1G/H-CPU□□ (-V1) and CJ1v-CPU□□ CPU Units do not support extra cyclic tasks, so they have only 4 kinds of interrupt tasks.

a) Power OFF interrupt task (Not supported by CS1D CPU Units for Duplex-CPU Systems):

Executed when power is interrupted. (1 max.)

b) Scheduled interrupt task (Not supported by CS1D CPU Units for Duplex-CPU Systems):

Executed at specified intervals. (2 max.).

c) I/O interrupt task (Not supported by CJ1 or CS1D CPU Units for Duplex-CPU Systems):

Executed when an Interrupt Input Unit input turns ON (32 max.).

d) External interrupt task (Not supported by CJ1 or CS1D CPU Units for Duplex-CPU Systems):

Executed (256 max.) when requested by an Special I/O Unit, CPU Bus Unit, or Inner Board (CS Series only).

e) Extra cyclic tasks (Not supported by CS1G/H-CPU□□□(-V1) and CJ1□-CPU□□□ CPU Units):

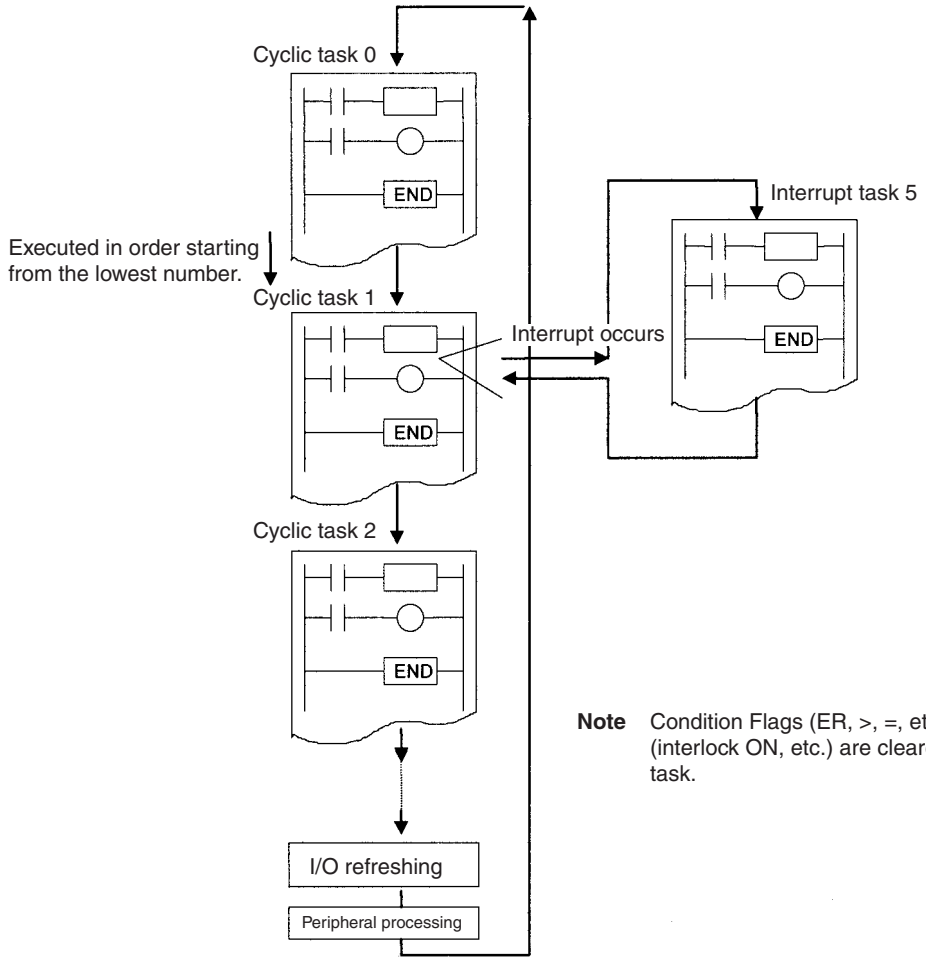
Interrupt tasks that are treated as cyclic tasks. Extra cyclic tasks are executed once every cycle as long as they are in an executable condition.

A total of 288 tasks with 288 programs can be created and controlled with the CX-Programmer. These include up to 32 cyclic tasks and 256 interrupt tasks.

Note

1. CJ1 CPU Units do not currently support I/O interrupt tasks and external interrupt tasks. The maximum number of tasks for a CJ1 CPU Unit is thus 35, i.e., 32 cyclic tasks and 3 interrupt tasks. The total number of programs that can be created and managed is also 35.
2. The CS1D CPU Units do not support any interrupt tasks. Interrupt tasks, however, can be used as extra cyclic tasks with CS1D CPU Units.

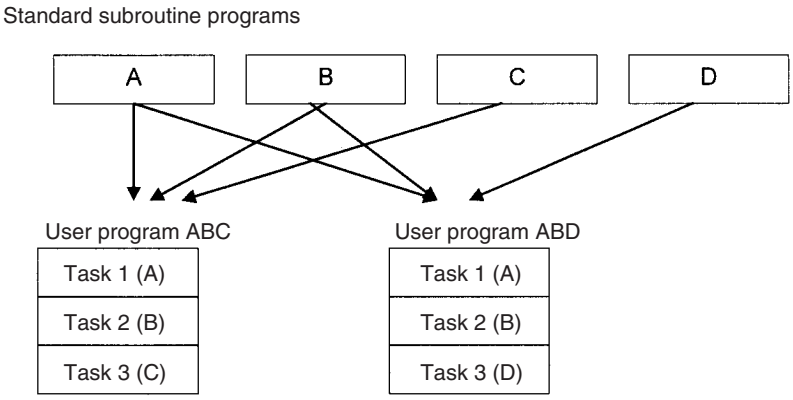
Each program is allocated 1:1 to a task through individual program property settings set with the CX-Programmer.



Note Condition Flags (ER, >, =, etc.) and instruction conditions (interlock ON, etc.) are cleared at the beginning of each task.

Program Structure

Standard subroutine programs can be created and allocated to tasks as needed to create programs. This means that programs can be created in modules (standard components) and that tasks can be debugged individually.



When creating modular programs, addresses can be specified by symbols to facilitate standardization.

Executable and Standby Status

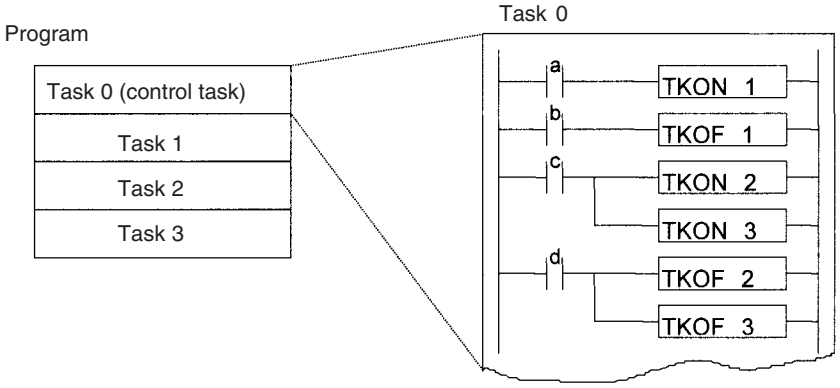
The TASK ON and TASK OFF instructions (TKON(820) and TKOF(821)) can be executed in one task to place another task in executable or standby status.

Instructions in tasks that are on standby will not be executed, but their I/O status will be maintained. When a task is returned to executable status, instructions will be executed with the I/O status that was maintained.

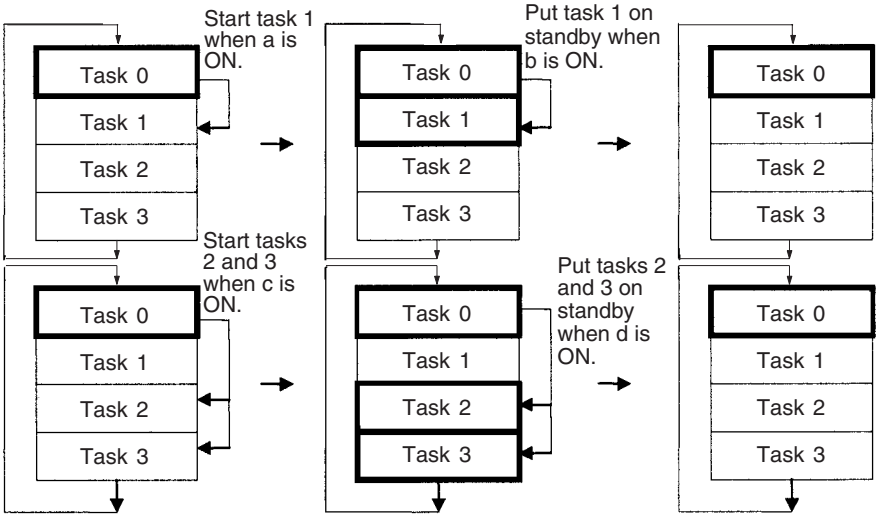
Example: Programming with a Control Task

In this example, task 0 is a control task that is executed first at the start of operation. Other tasks can be set from the CX-Programmer (but not a Programming Console) to start or not to start at the beginning of operation.

Once program execution has been started, tasks can be controlled with TKON(820) and TKOF(821).

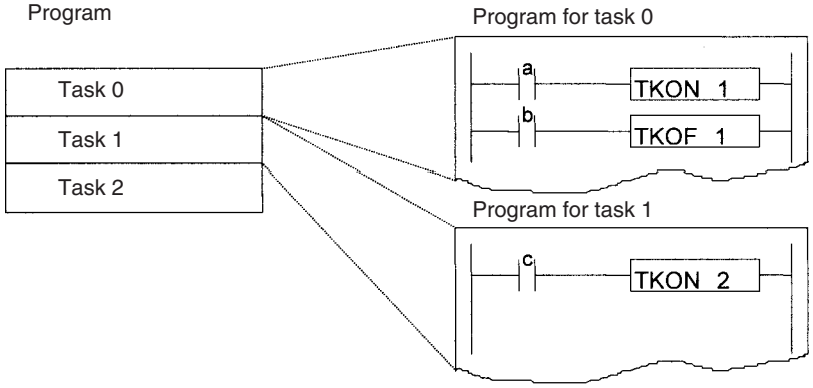


Example: Task 0 is set to be executed at the start of operation (set in the program properties from the CX-Programmer).
Task 1 is executable when a is ON.
Task 1 is put on standby when b is ON.
Tasks 2 and 3 are executable when c is ON.
Tasks 2 and 3 are put on standby when d is ON.

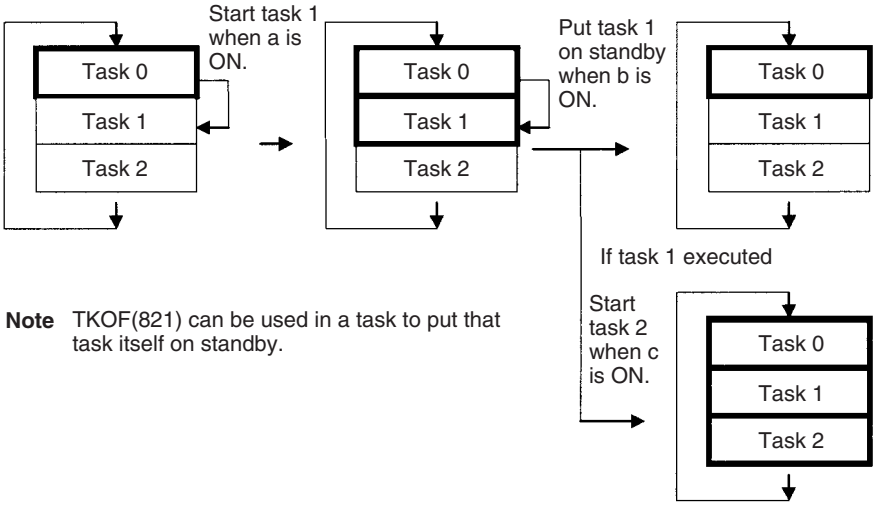

Start task 1 when a is ON.
Put task 1 on standby when b is ON.
Start tasks 2 and 3 when c is ON.
Put tasks 2 and 3 on standby when d is ON.
Task 0
Task 1
Task 2
Task 3

Example: Each Task Controlled by Another Task

In this example, each task is controlled by another task.



Example: Task 1 is set to be executed at the start of operation unconditionally.
Task 1 executable when a is ON.
Task 1 put on standby when b is ON.
Task 2 is executable when c is ON and task 1 has been executed.



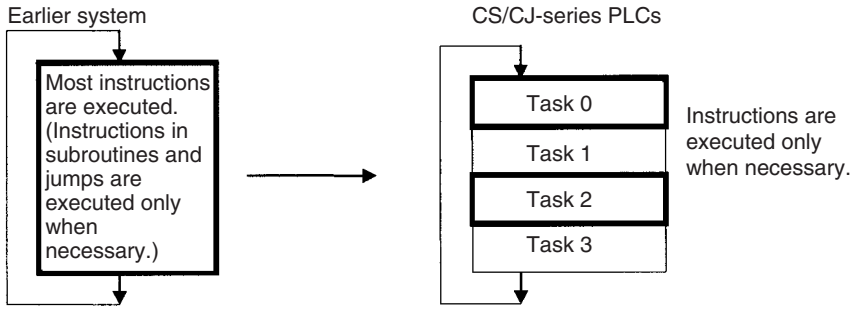
Note TKOF(821) can be used in a task to put that task itself on standby.

Task Execution Time

While a task is on standby, instructions in that task are not executed, so their OFF instruction execution time will not be added to the cycle time.

Note From this standpoint, instructions in a task that is on standby are just like instructions in a jumped program section (JMP-JME).

Since instructions in a non-executed task do not add to the cycle time, the overall system performance can be improved significantly by splitting the system into an overall control task and individual tasks that are executed only when necessary.



SECTION 2
Programming

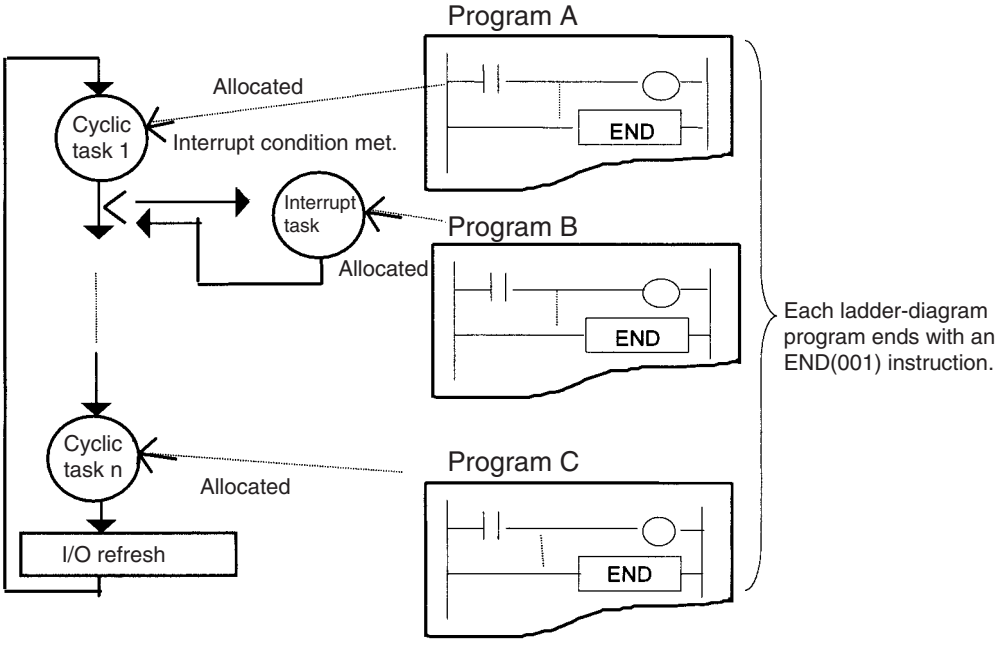
This section basic information required to write, check, and input programs.

2-1 Basic Concepts

2-1-1 Programs and Tasks

CS/CJ-series PLCs execute ladder-diagram programs contained in tasks. The ladder-diagram program in each task ends with an END(001) instruction just as with conventional PLCs.

Tasks are used to determine the order for executing the ladder-diagram programs, as well as the conditions for executing interrupts.



This section describes the basic concepts required to write CS/CJ-series programs. See *SECTION 4 Tasks* for more information on tasks and their relationship to ladder-diagram programs.

Note **Tasks and Programming Devices**

Tasks are handled as described below on the Programming Devices. Refer to *4-4 Programming Device Operations for Tasks* and to the *CS/CJ-series Programming Consoles Operation Manual (W341)* and *CX-Programmer Operation Manual* for more details.

CX-Programmer

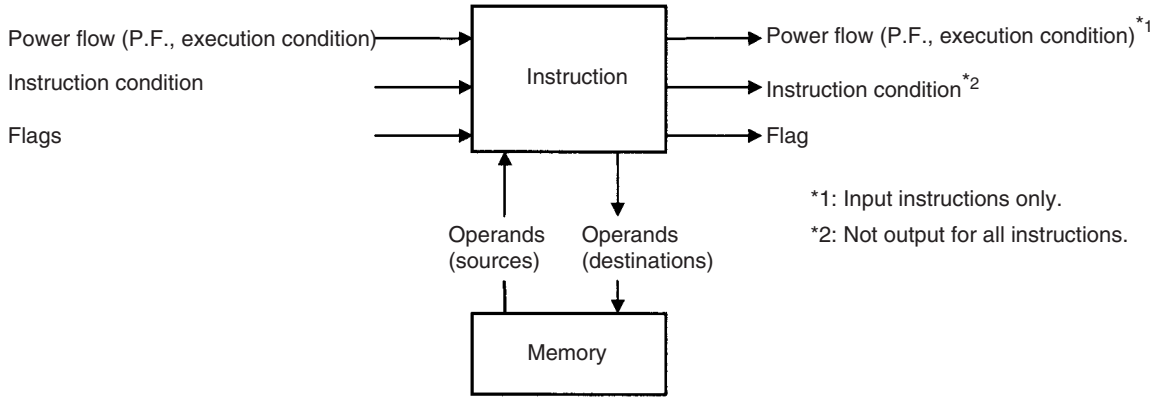
The CX-Programmer is used to designate task types and task numbers as attributes for individual programs.

Programming Console

Programs are accessed and edited on a Programming Console by specifying CT00 to CT 31 for cyclic tasks and IT00 to IT255 for interrupt tasks. When the memory clear operation is performed with a Programming Console, only cyclic task 0 (CT00) can be written in a new program. Use CX-Programmer to create cyclic tasks 1 through 31 (CT01 through CT31).

2-1-2 Basic Information on Instructions

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

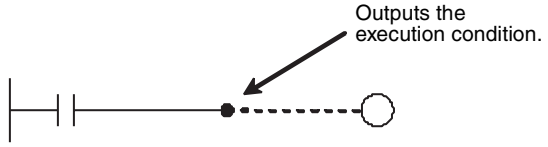


Power Flow

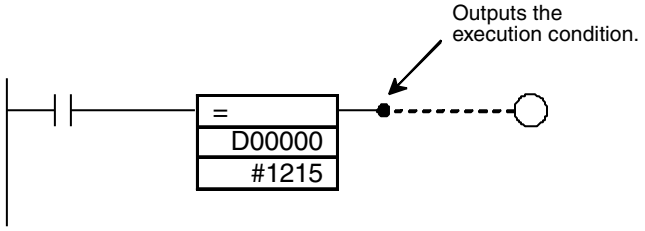
The power flow is the execution condition that is used to control the execute and instructions when programs are executing normally. In a ladder program, power flow represents the status of the execution condition.

Input Instructions

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

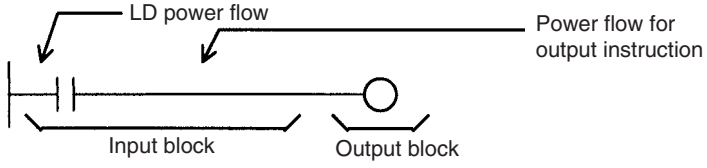


- Intermediate instructions input the power flow as an execution condition and output the power flow to an intermediate or output instruction.



Output Instructions

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



Instruction Conditions

Instruction conditions are special conditions related to overall instruction execution that are output by the following instructions. Instruction conditions have a higher priority than power flow (P.F.) when it comes to deciding whether or not to execute an instruction. An instruction may become not be executed or may act differently depending on instruction conditions. Instruction conditions

are reset (canceled) at the start of each task, i.e., they are reset when the task changes.

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

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

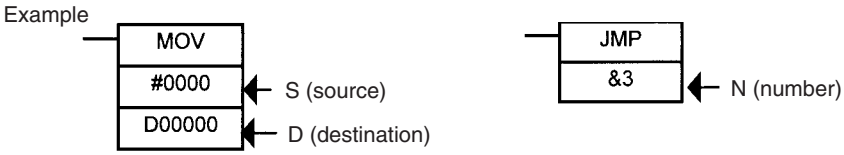
Flags

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

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

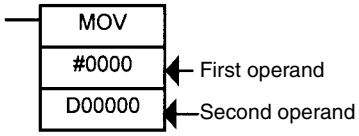
Operands

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



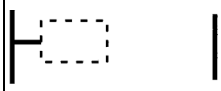
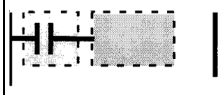
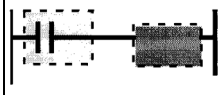
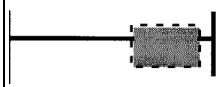
Operand types		Operand symbol	Description	
Source	Specifies the address of the data to be read or a constant.	S	Source Operand	Source operand other than control data (C)
		C	Control data	Compound data in a source operand that has different meanings depending bit status.
Destination (Results)	Specifies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

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



2-1-3 Instruction Location and Execution Conditions

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

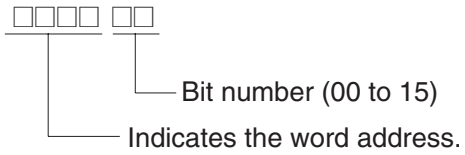
Instruction type		Possible location	Execution condition	Diagram	Examples
Input instructions	Logical start (Load instructions)	Connected directly to the left bus bar or is at the beginning of an instruction block.	Not required.		LD, LD TST(350), LD > (and other symbol comparison instructions)
	Intermediate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

Note

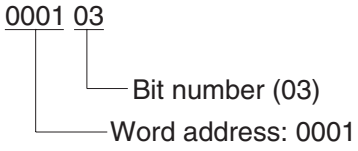
1. There is another group of instruction that executes a series of mnemonic instructions based on a single input. These are called block programming instructions. Refer to the *CS/CJ Series CPU Units Instruction Reference Manual* for details on these block programs.
2. If an instruction requiring an execution condition is connected directly to the left bus bar without a logical start instruction, a program error will occur when checking the program on a Programming Device (CX-Programmer or Programming Console).

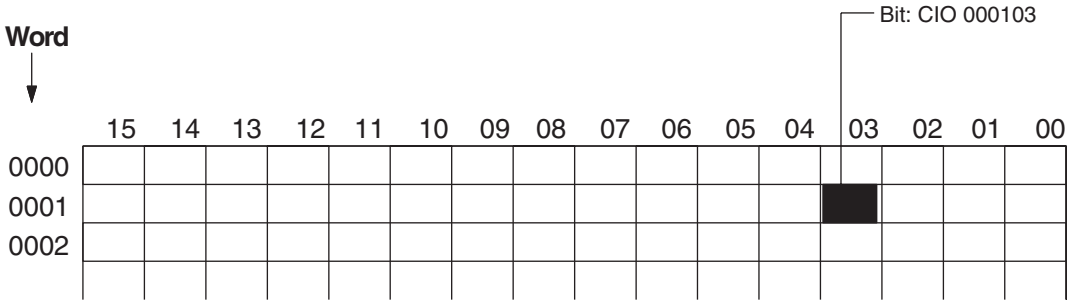
2-1-4 Addressing I/O Memory Areas

Bit Addresses

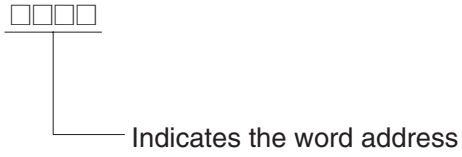


Example: The address of bit 03 in word 0001 in the CIO Area would be as shown below. This address is given as “CIO 000103” in this manual.

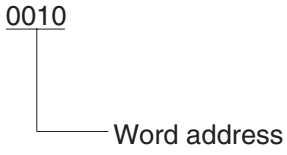




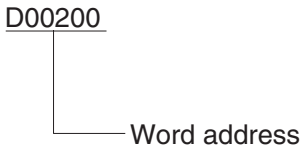
Word Addresses



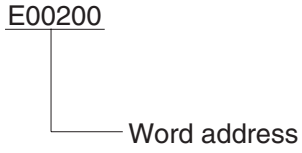
Example: The address of bits 00 to 15 in word 0010 in the CIO Area would be as shown below. This address is given as “CIO 0010” in this manual.



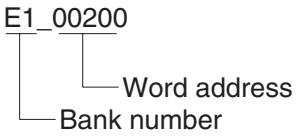
DM and EM Areas addresses are given with “D” or “E” prefixes, as shown below for the address D00200.



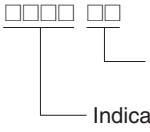
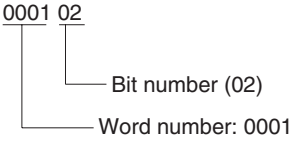
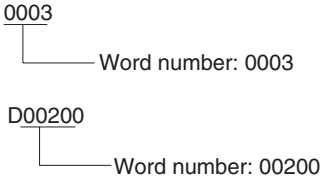
Example: The address of word 2000 in the current bank of the Extended Data Memory would be as follows:

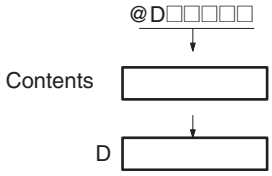


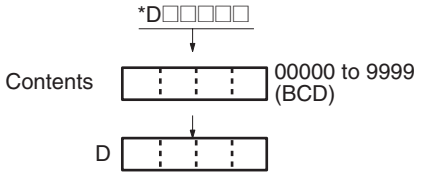
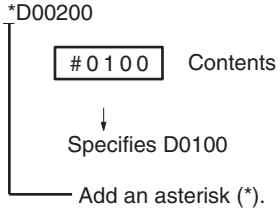
The address of word 2000 in the bank 1 of the Extended Data Memory would be as follows:



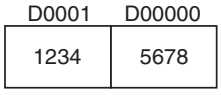
2-1-5 Specifying Operands

Operand	Description	Notation	Application examples
Specifying bit addresses	The word and bit numbers are specified directly to specify a bit (input bits).  **Note** The same addresses are used to access timer/counter Completion Flags and Present Values. There is also only one address for a Task Flag.		
Specifying word addresses	The word number is specified directly to specify the 16-bit word. 		MOV 0003 D00200

<table>
<tr><th>Operand</th><th>Description</th><th>Notation</th><th>Application examples</th></tr>
<tr><td rowspan="5">Specifying indirect DM/ EM addresses in Binary Mode</td><td>The offset from the beginning of the area is specified. The contents of the address will be treated as binary data (00000 to 32767) to specify the word address in Data Memory (DM) or Extended Data Memory (EM). Add the @ symbol at the front to specify an indirect address in Binary Mode.
 
</td><td></td><td></td></tr>
<tr><td>1) D00000 to D32767 are specified if @D(□□□□□) contains 0000 Hex to 7FFF Hex (00000 to 32767).</td><td>@D00300
&256 Contents
Hex: #0100
Specifies D00256.
Add the @ symbol.</td><td rowspan="3">MOV #0001 @00300</td></tr>
<tr><td>2) E0 _00000 to E0 _32767 of bank 0 in Extended Data Memory (EM) are specified if @D(□□□□□) contains 8000 Hex to FFFF Hex (32768 to 65535).</td><td>@D00300
&32769 Contents
Hex: #8001
Specifies E0 00001.</td></tr>
<tr><td>3) E□_00000 to E□_32767 in the specified bank are specified if @E□_□□□□□ contains 0000 Hex to 7FFF Hex (00000 to 32767).</td><td>@E1_00200
&257 Contents
Hex: #0101
Specifies E1_00257.</td><td rowspan="2">MOV #0001 @E1_00200</td></tr>
<tr><td>4) E(□+1)_00000 to E(□+1)_32767 in the bank following the specified bank □ are specified if @E□_□□□□□ contains 8000 Hex to FFFF Hex (32768 to 65535).</td><td>@E1_00200
&32770 Contents
Hex: #8002
Specifies E2_00002.</td></tr>
<tr><td colspan="3">Note When specifying an indirect address in Binary Mode, treat Data Memory (DM) and Extended Data Memory (EM) (banks 0 to C) as one series of addresses. If the contents of an address with the @ symbol exceeds 32767, the address will be assumed to be an address in the Extended Data Memory (EM) continuing on from 00000 in bank No. 0.
Example: If the Data Memory (DM) word contains 32768, E1_00000 in bank 0 in Extended Data Memory (EM) would be specified.
Note If the Extended Data Memory (EM) bank number is specified as “n” and the contents of the word exceeds 32767, the address will be assumed to be an address in the Extended Data Memory (EM) continuing on from 00000 in bank N+1.
Example: If bank 2 in Extended Data Memory (EM) contains 32768, E3_00000 in bank number 3 in Extended Data Memory (EM) would be specified.</td></tr>
</table>

Operand	Description	Notation	Application examples
Specifying indirect DM/ EM addresses in BCD Mode	The offset from the beginning of the area is specified. The contents of the address will be treated as BCD data (0000 to 9999) to specify the word address in Data Memory (DM) or Extended Data Memory (EM). Add an asterisk (*) at the front to specify an indirect address in BCD Mode. 		MOV #0001 *D00200

Operand	Description		Notation	Application examples
Specifying a register directly	An index register (IR) or a data register (DR) is specified directly by specifying IR□ (□: 0 to 15) or DR□ (□: 0 to 15).		IR0 IR1	MOVR 000102 IR0 Stores the PLC memory address for CIO 0010 in IR0. MOVR 0010 IR1 Stores the PLC memory address for CIO 0010 in IR1.
Specifying an indirect address using a register	Indirect address (No offset)	The bit or word with the PLC memory address contained in IR□ will be specified. Specify ,IR□ to specify bits and words for instruction operands.	,IR0 ,IR1	LD ,IR0 Loads the bit with the PLC memory address in IR0. MOV #0001 ,IR1 Stores #0001 in the word with the PLC memory in IR1.
	Constant offset	The bit or word with the PLC memory address in IR□ + or – the constant is specified. Specify *+/– constant* ,IR□. Constant offsets range from –2048 to +2047 (decimal). The offset is converted to binary data when the instruction is executed.	+5,IR0 +31,IR1	LD +5 ,IR0 Loads the bit with the PLC memory address in IR0 + 5. MOV #0001 +31 ,IR1 Stores #0001 in the word with the PLC memory address in IR1 + 31
	DR offset	The bit or word with the PLC memory address in IR□ + the contents of DR□ is specified. Specify DR□ ,IR□. DR (data register) contents are treated as signed-binary data. The contents of IR□ will be given a negative offset if the signed binary value is negative.	DR0 ,IR0 DR0 ,IR1	LD DR0 ,IR0 Loads the bit with the PLC memory address in IR0 + the value in DR0. MOV #0001 DR0 ,IR1 Stores #0001 in the word with the PLC memory address in IR1 + the value in DR0.
	Auto Increment	The contents of IR□ is incremented by +1 or +2 after referencing the value as an PLC memory address. +1: Specify ,IR□+ +2: Specify ,IR□ + +	,IR0 ++ ,IR1 +	LD ,IR0 ++ Increments the contents of IR0 by 2 after the bit with the PLC memory address in IR0 is loaded. MOV #0001 ,IR1 + Increments the contents of IR1 by 1 after #0001 is stored in the word with the PLC memory address in IR1.
	Auto Decrement	The contents of IR□ is decremented by –1 or –2 after referencing the value as an PLC memory address. –1: Specify ,–IR□ –2: Specify ,– –IR□	,– –IR0 ,–IR1	LD ,– –IR0 After decrementing the contents of IR0 by 2, the bit with the PLC memory address in IR0 is loaded. MOV #0001 ,–IR1 After decrementing the contents of IR1 by 1, #0001 is stored in the word with the PLC memory address in IR1.

Data	Operand	Data form	Symbol	Range	Application example
16-bit constant	All binary data or a limited range of binary data	Unsigned binary	#	#0000 to #FFFF	MOV #0100 D00000 Stores #0100 hex (&256 decimal) in D00000. +#0009 #0001 D00001 Stores #000A hex (&10 decimal) in D00001.
		Signed decimal	±	–32768 to +32767	MOV –100 D00000 Stores –100 decimal (#FF9C hex) in D00000. +–9 –1 D00001 Stores –10 decimal (#FFF6 hex) in D00001.
		Unsigned decimal	& (See Note.)	&0 to &65535	MOV &256 D00000 Stores –256 decimal (#0100 hex) in D00000. +&9 &1 D00001 Stores –10 decimal (#000A hex) in D00001.
	All BCD data or a limited range of BCD data	BCD	#	#0000 to #9999	MOV #0100 D00000 Stores #0100 (BCD) in D00000. +B #0009 #0001 D00001 Stores #0010 (BCD) in D00001.
32-bit constant	All binary data or a limited range of binary data	Unsigned binary	#	#00000000 to #FFFFFFFF	MOVL #12345678 D00000 Stores #12345678 hex in D00000 and D00001. 
		Signed binary	+	–2147483648 to +2147483647	MOVL –12345678 D00000 Stores –12345678 decimal in D00000 and D00001.
		Unsigned decimal	& (See Note.)	&0 to &429467295	MOVL &12345678 D00000 Stores &12345678 decimal in D00000 and D00001.
	All BCD data or a limited range of BCD data	BCD	#	#00000000 to #99999999	MOVL #12345678 D00000 Stores #12345678 (BCD) in D00000 and D00001

<table>
<tr><th>Data</th><th colspan="2">Operand</th><th>Data form</th><th>Symbol</th><th>Range</th><th>Application example</th></tr>
<tr><td rowspan="2">Text string</td><td colspan="3">Description</td><td>Symbol</td><td>Examples</td><td>---</td></tr>
<tr><td colspan="3">Text string data is stored in ASCII (one byte except for special characters) in order from the leftmost to the rightmost byte and from the rightmost (smallest) to the leftmost word.
00 Hex (NUL code) is stored in the rightmost byte of the last word if there is an odd number of characters.
0000 Hex (2 NUL codes) is stored in the leftmost and rightmost vacant bytes of the last word + 1 if there is an even number of characters.</td><td>---</td><td>'ABCDE'
<table><tr><td>'A'</td><td>'B'</td></tr><tr><td>'C'</td><td>'D'</td></tr><tr><td>'E'</td><td>NUL</td></tr></table>=
<table><tr><td>41</td><td>42</td></tr><tr><td>43</td><td>44</td></tr><tr><td>45</td><td>00</td></tr></table>
'ABCD'
<table><tr><td>'A'</td><td>'B'</td></tr><tr><td>'C'</td><td>'D'</td></tr><tr><td>NUL</td><td>NUL</td></tr></table>=
<table><tr><td>41</td><td>42</td></tr><tr><td>43</td><td>44</td></tr><tr><td>00</td><td>00</td></tr></table></td><td>MOV$ D00100 D00200
<table><tr><td>D00100</td><td>41</td><td>42</td></tr><tr><td>D00101</td><td>43</td><td>44</td></tr><tr><td>D00102</td><td>45</td><td>00</td></tr></table>↓
<table><tr><td>D00200</td><td>41</td><td>42</td></tr><tr><td>D00201</td><td>43</td><td>44</td></tr><tr><td>D00202</td><td>45</td><td>00</td></tr></table></td></tr>
<tr><td colspan="6">ASCII characters that can be used in a text string includes alphanumeric characters, Katakana and symbols (except for special characters). The characters are shown in the following table.</td></tr>
</table>

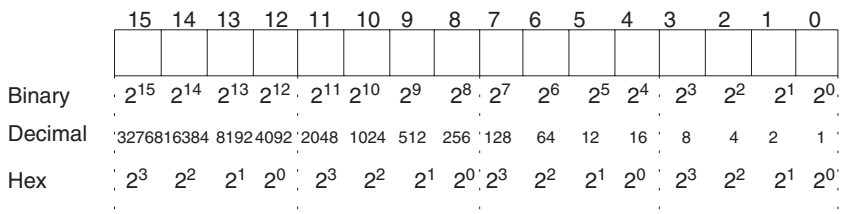
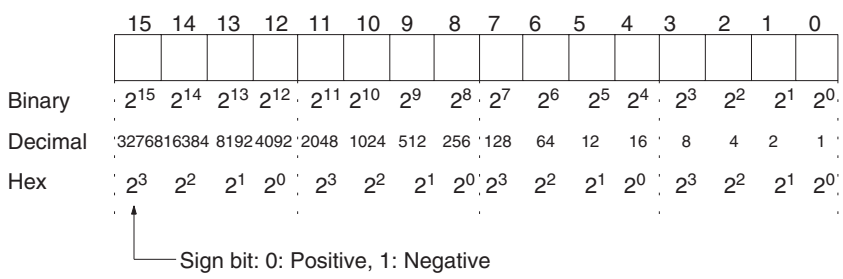
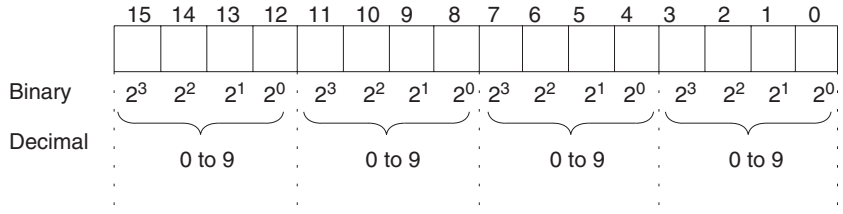
Note Unsigned decimal notation if used for the CX-Programmer only.

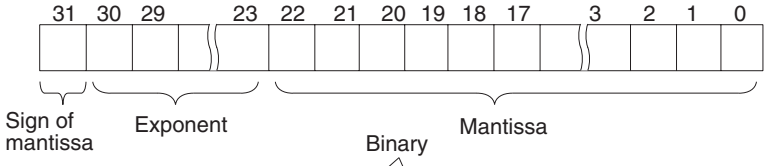
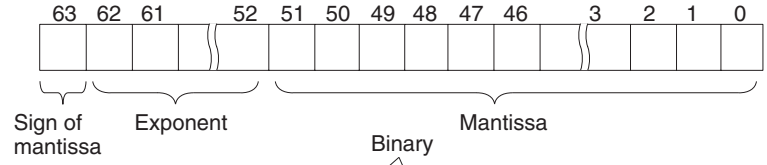
ASCII Characters

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

2-1-6 Data Formats

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

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

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

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

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

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

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

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

Complements

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

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

Two's Complements

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

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

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

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

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

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

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

Signed BCD Data

Signed BCD data is a special data format that is used to express negative numbers in BCD. Although this format is found in applications, it is not strictly defined and depends on the specific application. The CS/CJ Series supports the following instructions to convert the data formats: SIGNED BCD-TO-BINARY: BINS(470), DOUBLE SIGNED BCD-TO-BINARY: BISL(472),

SIGNED BINARY-TO-BCD: BCDS(471), and DOUBLE SIGNED BINARY-TO-BCD: BDSL(473). Refer to the *CS/CJ-series Programmable Controllers Instructions Reference Manual (W340)* for more information.

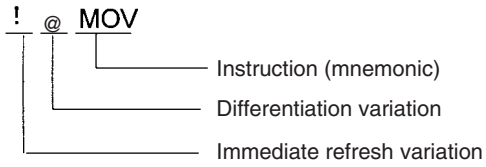
Values Represented in 1-word Data

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

2-1-7 Instruction Variations

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

Variation		Symbol	Description
Differentiation	ON	@	Instruction that differentiates when the execution condition turns ON.
	OFF	%	Instruction that differentiates when the execution condition turns OFF.
Immediate refreshing		!	Refreshes data in the I/O area specified by the operands or the Special I/O Unit words when the instruction is executed. (Immediate refreshing is not supported by the CS1D CPU Units for Duplex-CPU Systems.)



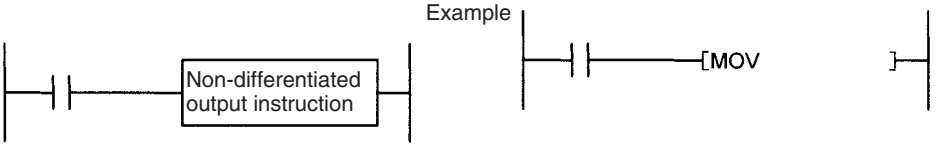
2-1-8 Execution Conditions

The CS/CJ Series offers the following types of basic and special instructions.

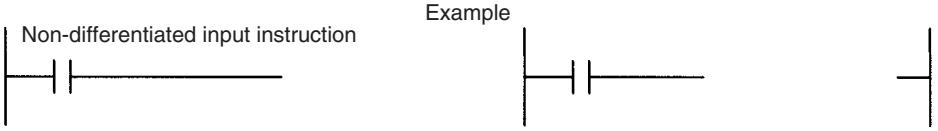
- Non-differentiated instructions executed every cycle
- Differentiated instructions executed only once

Non-differentiated Instructions

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



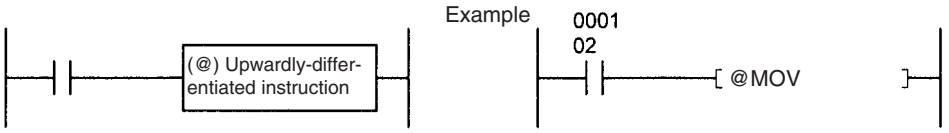
Input instructions that create logical starts and intermediate instructions read bit status, make comparisons, test bits, or perform other types of processing every cycle. If the results are ON, power flow is output (i.e., the execution condition is turned ON).



Input-differentiated Instructions

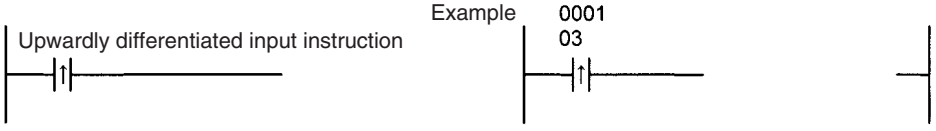
Upwardly Differentiated Instructions (Instruction Preceded by @)

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



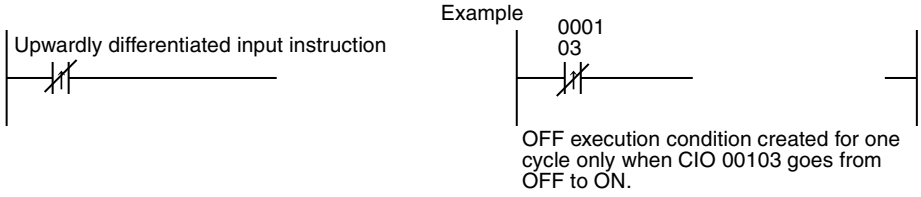
Executes the MOV instruction once when CIO 000102 goes OFF → ON.

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



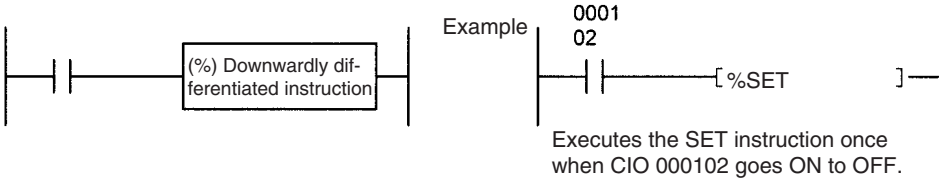
ON execution condition created for one cycle only when CIO 000103 goes from OFF to ON.

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

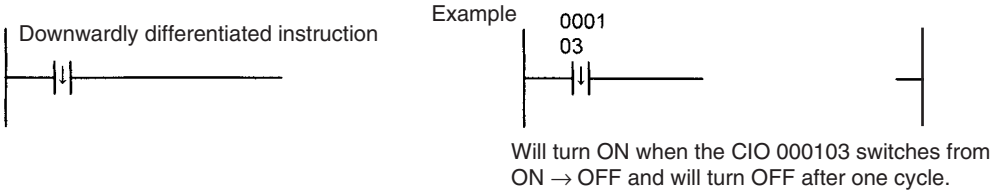


Downwardly Differentiated Instructions (Instruction preceded by %)

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

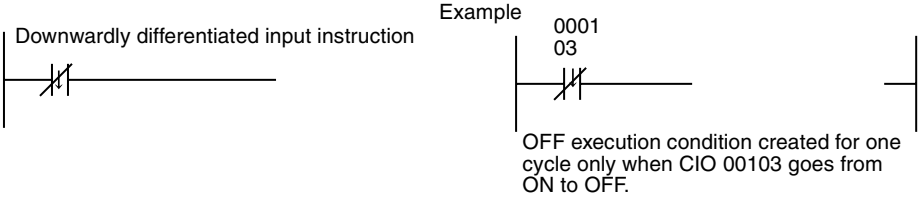


- **Input Instructions (Logical Starts and Intermediate Instructions)**: The instruction reads bit status, makes comparisons, tests bits, or perform other types of processing every cycle and will output the execution condition (power flow) when results switch from ON to OFF. The execution condition will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



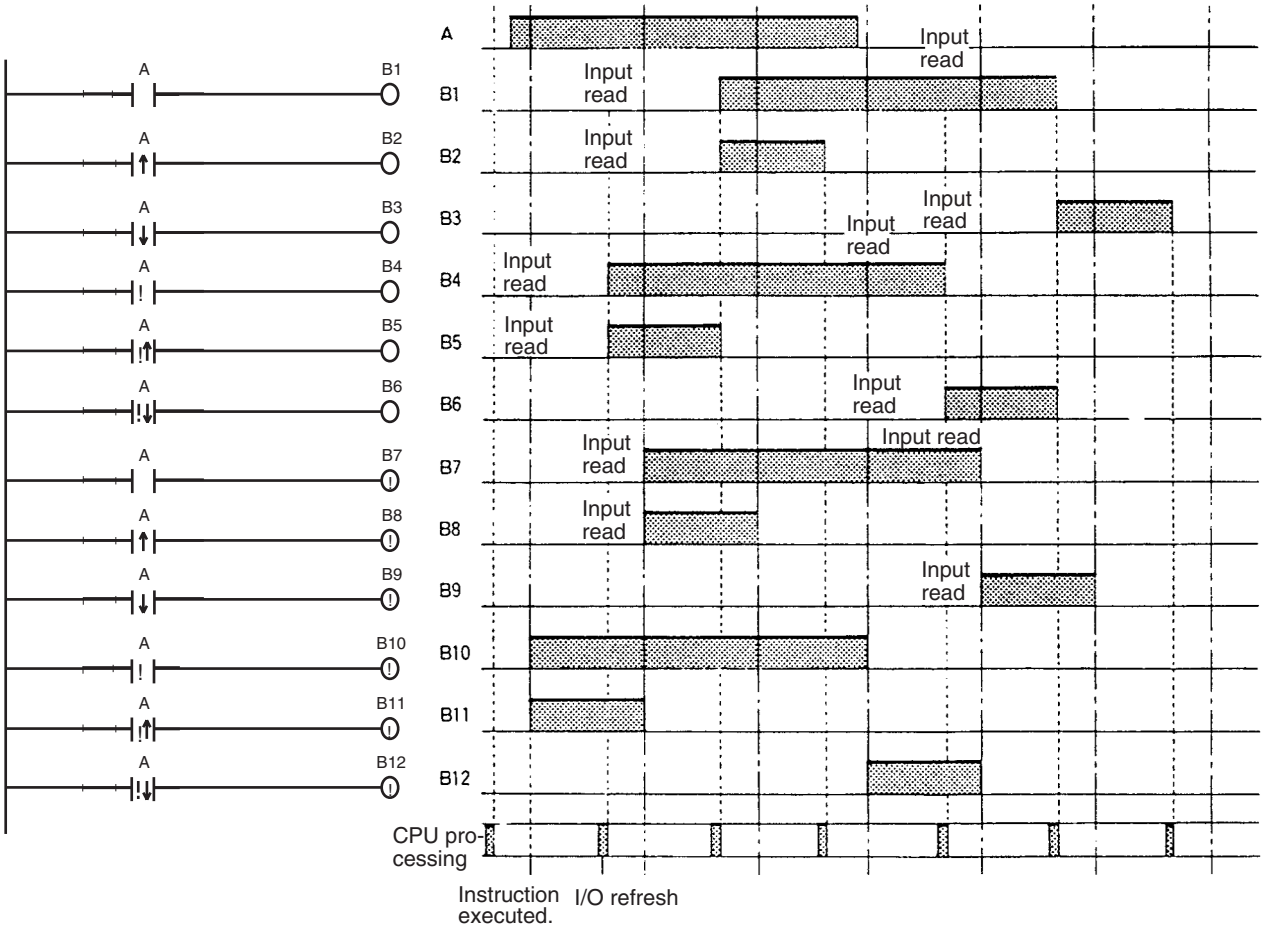
Note Unlike the upwardly differentiated instructions, downward differentiation variation (%) can only be added to LD, AND, OR, SET and RSET instructions. To execute downward differentiation with other instructions, combine the instructions with a DIFD or a DOWN instruction. NOT can be added to instructions only when using a CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit.

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



2-1-9 I/O Instruction Timing

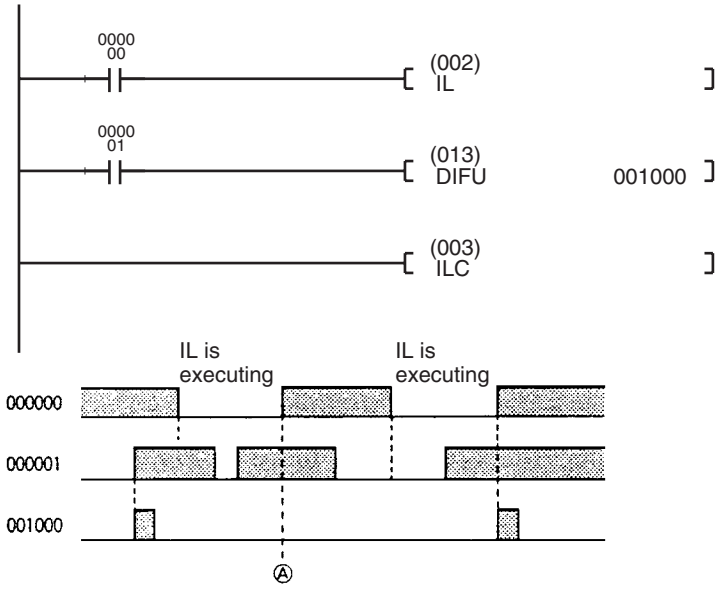
The following timing chart shows different operating timing for individual instructions using a program comprised of only LD and OUT instructions.



Differentiated Instructions

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

- **Use in Interlocks (IL - ILC Instructions)**
  In the following example, the previous value flag for the differentiated instruction maintains the previous interlocked value and will not output a differentiated output at point A because the value will not be updated while the interlock is in effect.



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

  Note a) Do not use the Always ON Flag or A20011 (First Cycle Flag) as the input bit for an upwardly differentiated instruction. The instruction will never be executed.

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

2-1-10 Refresh Timing

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

- Cyclic refresh
- Immediate refresh (instruction with the ! specification, IORF(097), FIORF(225), or DLNK(226))

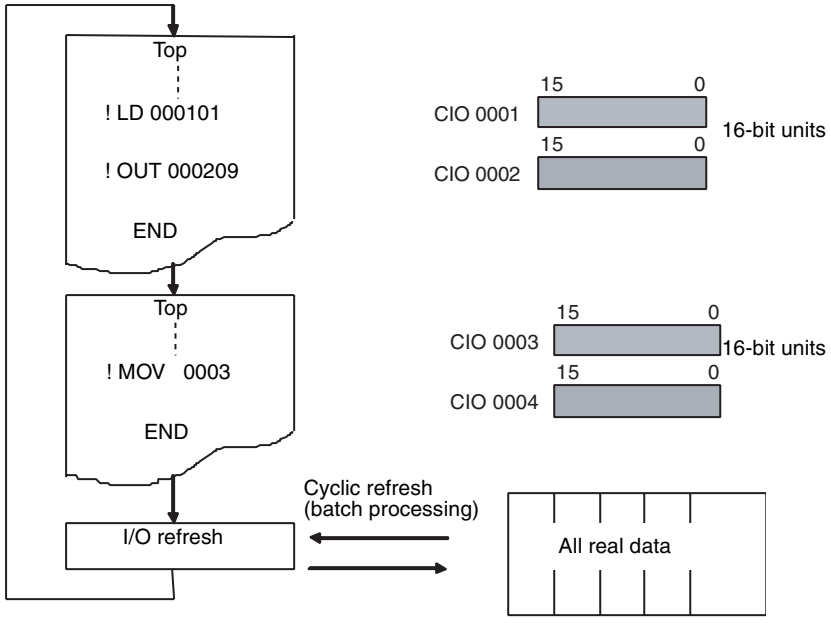
  Note FIORF(225) is supported by CJ1-H-R CPU Units only.

Refer to the section on CPU Unit operation in the *CS/CJ Series Operation Manual* for details on the I/O refresh.

Cyclic Refresh

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

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



Execute an IORF instruction for all required words prior to the END(001) instruction if I/O refreshing is required in other tasks.

Immediate Refresh

Instructions with Refresh Variation (!)

I/O will be refreshed as shown below when an instruction is executing if an real I/O bit is specified as an operand.

Units	Refreshed data
C200H Basic I/O Units (CS Series only)	I/O will be refreshed for the 16 bits containing the bit.
CJ Basic I/O Units	

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

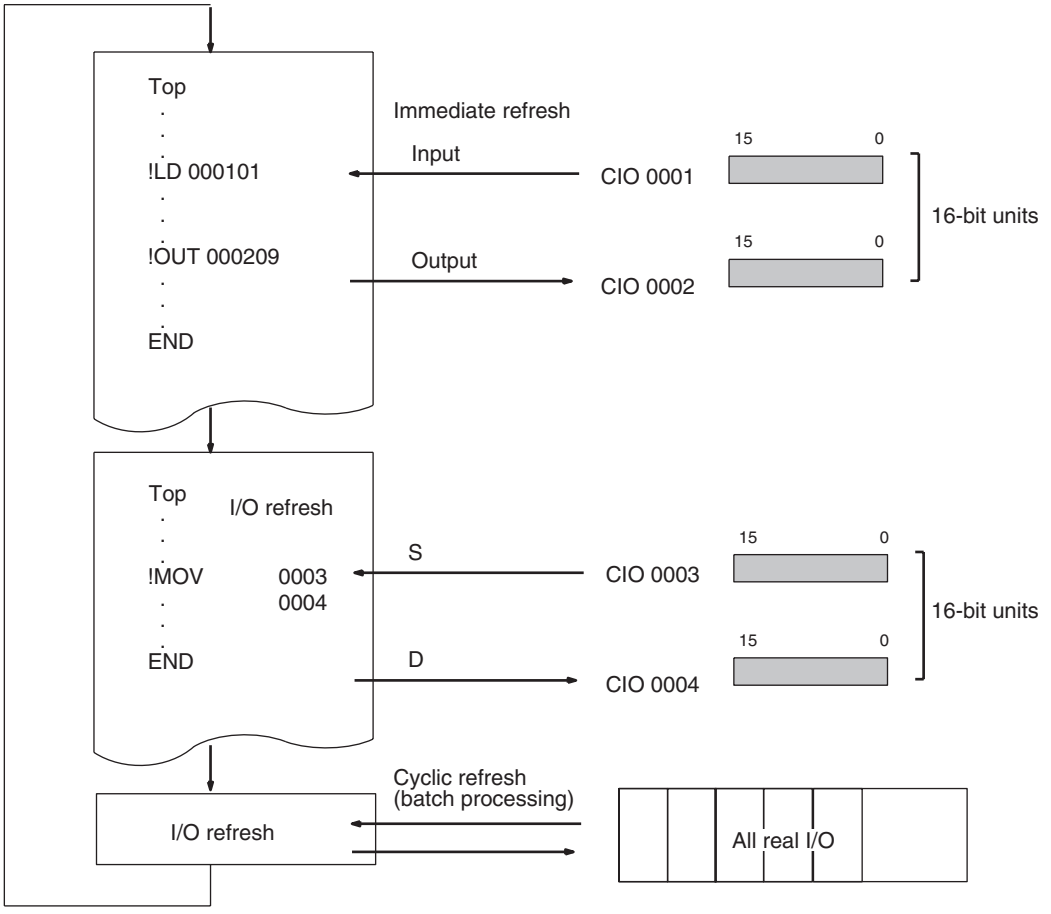
Add an exclamation mark (!) (immediate refresh option) in front of the instruction.

Note Immediate refreshing is not supported by the CS1D CPU Units for Duplex-CPU Systems, but they do support refreshing for IORF(097) and DLNK(226) instructions.

Units Refreshed by the Immediate Refreshing (!) Specification

Location	CPU or Expansion I/O Rack (but not SYSMAC BUS Slave Racks)		
Units	Basic I/O Units	CS/CJ-series Basic I/O Units	Refreshed
		C200H Basic I/O Unit (See note.)	Refreshed
		C200H Group-2 High-density I/O Units (See note.)	Not refreshed
	Special I/O Units		Not refreshed
	CPU Bus Units		Not refreshed

Note C200H I/O Units cannot be mounted to CJ-series PLCs.



Immediate Refreshing by IORF(097), FIORF(225), or DLNK(226)

There are three instructions available to immediately refresh the words allocated to different kinds of Units: I/O REFRESH (IORF(097)), SPECIAL I/O UNIT I/O REFRESH (FIORF(225)), and CPU BUS UNIT I/O REFRESH (DLNK(226)).

- IORF(097): I/O REFRESH

  IORF(097) can immediately refresh a specified range of I/O words in the CIO Area, or a range of CIO words allocated to Special I/O Units.

- FIORF(225): SPECIAL I/O UNIT I/O REFRESH (See note 2.)

  FIORF(225) can immediately refresh the words allocated to a specified Special I/O Unit. This instruction refreshes the words allocated to the Special I/O Unit in both the CIO Area and DM Area.

- DLNK(226): CPU BUS UNIT REFRESH (See note 3.)

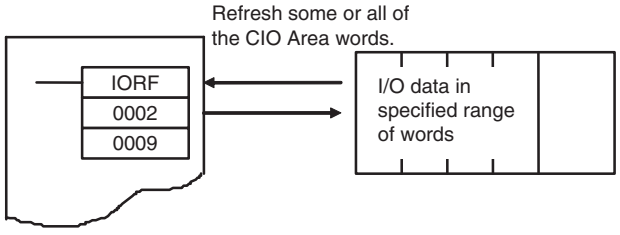
  DLNK(226) can immediately refresh the words allocated to a specified CPU Bus Unit. This instruction refreshes the words allocated to the CPU Bus Unit in both the CIO Area and DM Area.

Note

1. Both IORF(097) and FIORF(225) can immediately refresh the CIO Area words allocated to a Special I/O Unit, but FIORF(225) has a much faster instruction execution time. FIORF(225) is also easier to use because the Unit's words are specified by its unit number, rather than directly specified as a range of words.
2. FIORF(225) is supported by CJ1-H-R CPU Units only.
3. DLNK(226) is not supported by the CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units.

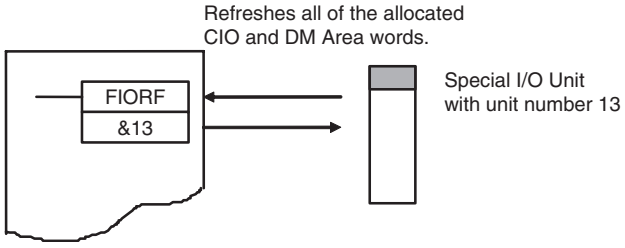
Units Refreshed by IORF(097)

Location	CPU or Expansion I/O Rack (but not SYSMAC BUS Slave Racks)		
Units	Basic I/O Units	CS/CJ-series Basic I/O Units	Refreshed
		C200H Basic I/O Units	Refreshed
		C200H Group-2 High-density I/O Units	Refreshed
	Special I/O Units (allocated CIO words only)		Refreshed
	CPU Bus Units		Not refreshed



Units Refreshed by FIORF(225)

Location	CPU or Expansion I/O Rack (but not SYSMAC BUS Slave Racks)	
Units	Basic I/O Units	Not refreshed
	Special I/O Units	Not refreshed
	CPU Bus Units Refreshes words allocated to the Unit in both the CIO Area and DM Area.	Refreshed
	CPU Bus Units	Not refreshed

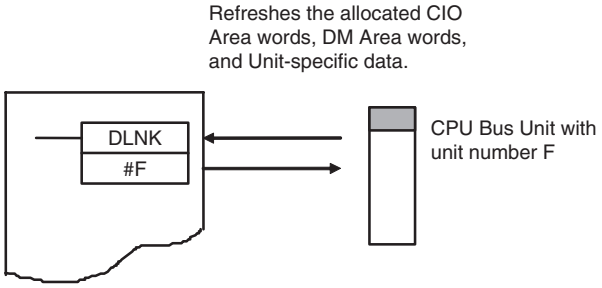


Note FIORF(225) is supported by the CJ1-H-R CPU Units only.

Units Refreshed by DLNK(226)

Location	CPU or Expansion I/O Rack (but not SYSMAC BUS Slave Racks)	
Units	Basic I/O Units	Not refreshed
	Special I/O Units	Not refreshed
	CPU Bus Units Refreshes the following data: • Words allocated to the Unit in CIO Area • Words allocated to the Unit in DM Area • Data specific to the CPU Bus Unit, such as data link data or DeviceNet Remote I/O Communications data	Refreshed

Note DLNK(226) is not supported by the CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units.



2-1-11 Program Capacity

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

Each instruction is from 1 to 7 steps long. Refer to *10-5 Instruction Execution Times and Number of Steps* in the *Operation Manual* for the specific number

of steps in each instruction. (The length of each instruction will increase by 1 step if a double-length operand is used.)

Series	CPU Unit	Max. program capacity	I/O points
CS Series	CS1H-CPU67H/CPU67-E	250K steps	5,120
	CS1D-CPU67H	250K steps	
	CS1D-CPU67S	250K steps	
	CS1H-CPU66H/CPU66-E	120K steps	
	CS1H-CPU65H/CPU65-E	60K steps	
	CS1D-CPU65H	60K steps	
	CS1D-CPU65S	60K steps	
	CS1H-CPU64H/CPU64-E	30K steps	
	CS1H-CPU63H/CPU63-E	20K steps	
	CS1G-CPU45H/CPU45-E	60K steps	
	CS1G-CPU44H/CPU44-E	30K steps	1,280
	CS1D-CPU44S	30K steps	
	CS1G-CPU43H/CPU43-E	20K steps	960
	CS1G-CPU42H/CPU42-E	10K steps	
	CS1D-CPU42S	10K steps	
CJ Series	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/CPU45	60K steps	1280
	CJ1G-CPU44H/CPU44	30K steps	
	CJ1G-CPU43H	20K steps	960
	CJ1G-CPU42H	10K steps	
	CJ1M-CPU23/CPU13	20K steps	640
	CJ1M-CPU22/CPU12	10K steps	320
	CJ1M-CPU21/CPU11	5K steps	160

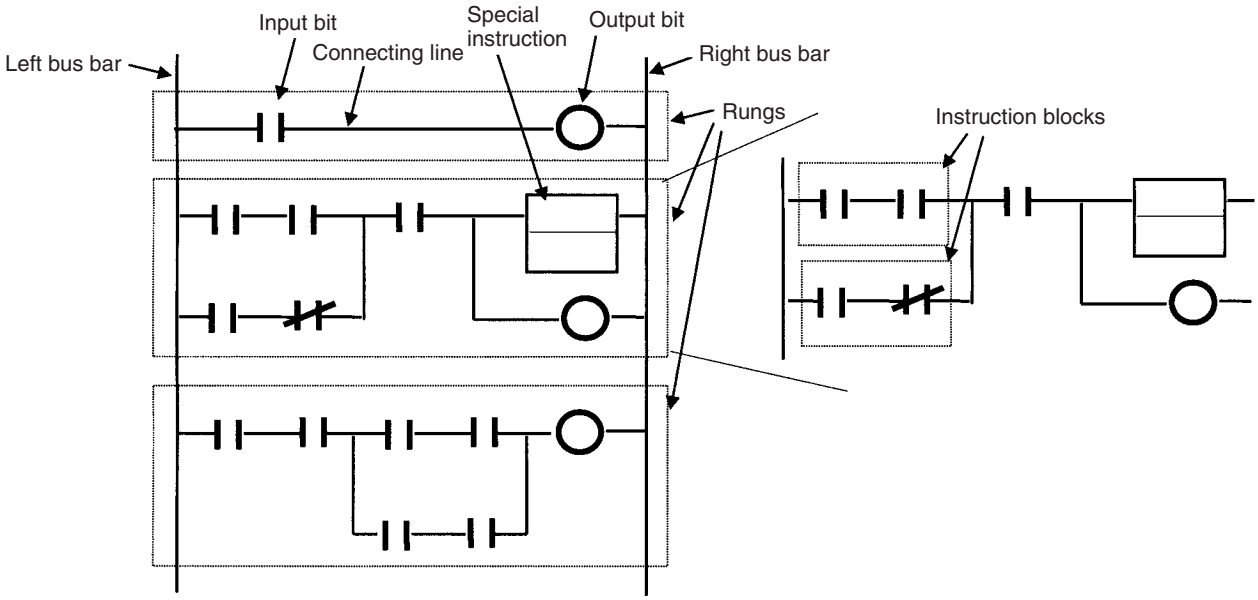
Note Memory capacity for CS/CJ-series PLCs is measured in steps, whereas memory capacity for previous OMRON PLCs, such as the C200HX/HG/HE and CV-series PLCs, was measured in words. Refer to the information at the end of *10-5 Instruction Execution Times and Number of Steps* in the *Operation Manual* for your PLC for guidelines on converting program capacities from previous OMRON PLCs.

2-1-12 Basic Ladder Programming Concepts

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

General Structure of the Ladder Diagram

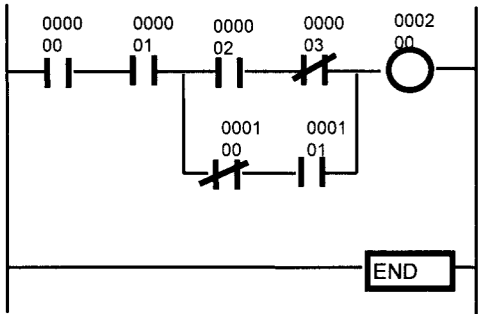
A ladder diagram consists of left and right bus bars, connecting lines, input bits, output bits, and special instructions. A program consists of one or more program runs. A program rung is a unit that can be partitioned when the bus is split horizontally. In mnemonic form, a rung is all instructions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



Mnemonics

A mnemonic program is a series of ladder diagram instructions given in their mnemonic form. It has program addresses, and one program address is equivalent to one instruction. Program addresses contain six digits starting from 000000.

Example

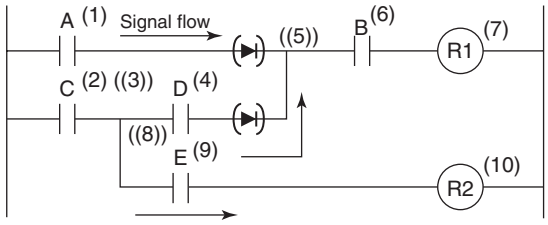


Program Address	Instruction (Mnemonic)	Operand
000000	LD	000000
000001	AND	000001
000002	LD	000002
000003	AND NOT	000003
000004	LD NOT	000100
000005	AND	000101
000006	OR LD	
000007	AND LD	
000008	OUT	000200
000009	END	

Basic Ladder Program Concepts

1,2,3... 1. When ladder diagrams are executed by PLCs, the signal flow (power flow) is always from left to right. Programming that requires power flow from right to left cannot be used. Thus, flow is different from when circuits are made up of hard-wired control relays. For example, when the circuit “a” is implemented in a PLC program, power flows as though the diodes in brackets were inserted and coil R2 cannot be driven with contact D included. The actual order of execution is indicated on the right with mnemonics. To achieve operation without these imaginary diodes, the circuit must be rewritten. Also, circuit “b” power flow cannot be programmed directly and must be rewritten.

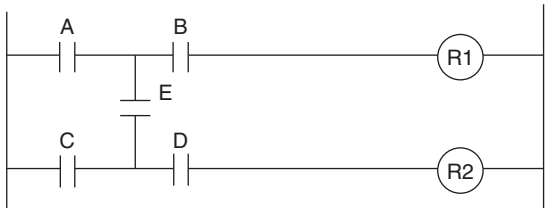
Circuit "a"



Order of execution (mnemonic)

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

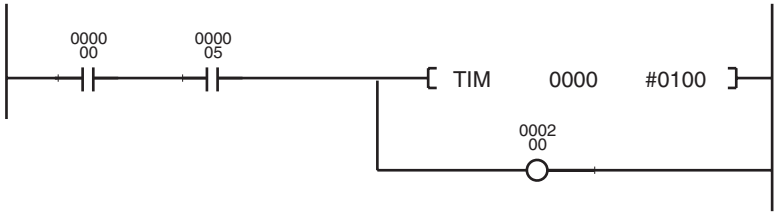
Circuit "b"



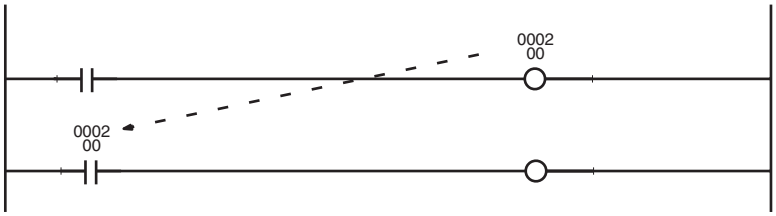
In circuit “a,” coil R2 cannot be driven with contact D included.

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

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



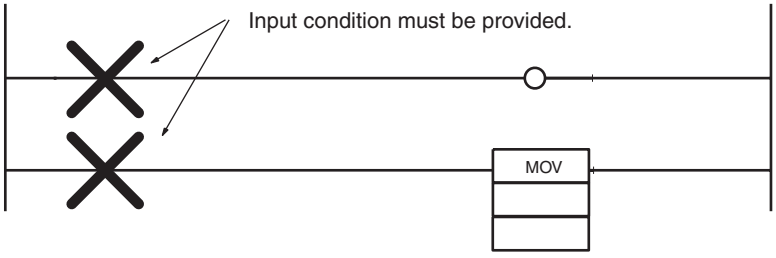
5. Output bits can also be used as input bits.



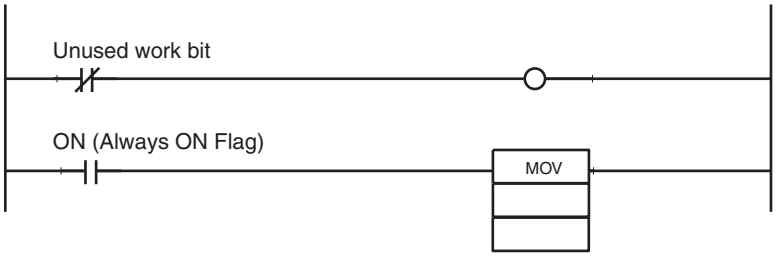
Restrictions

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

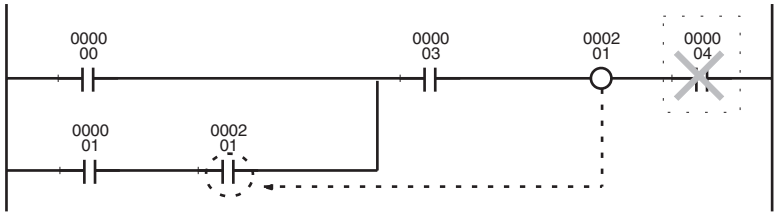
2. Output bits, timers, counters and other output instructions cannot be connected directly to the left bus bar. If one is connected directly to the left bus bar, a rung error will occur during the programming check by a Programming Device. (The program can be executed, but the OUT and MOV(021) will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



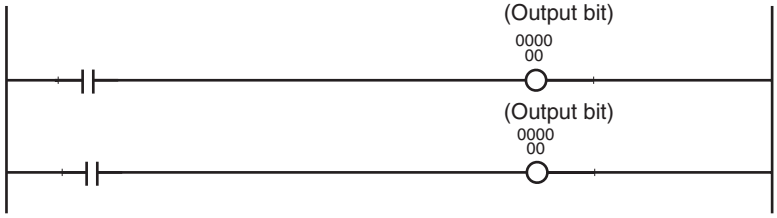
Insert an unused N.C. work bit or the ON Condition Flag (Always ON Flag) if the input must be kept ON at all times.



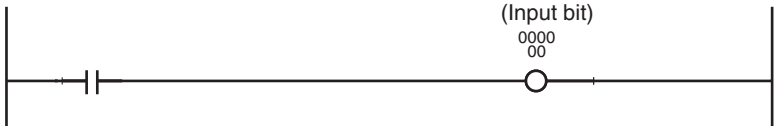
3. An input bit must always be inserted before and never after an output instruction like an output bit. If it is inserted after an output instruction, then a location error will occur during a Programming Device program check.



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

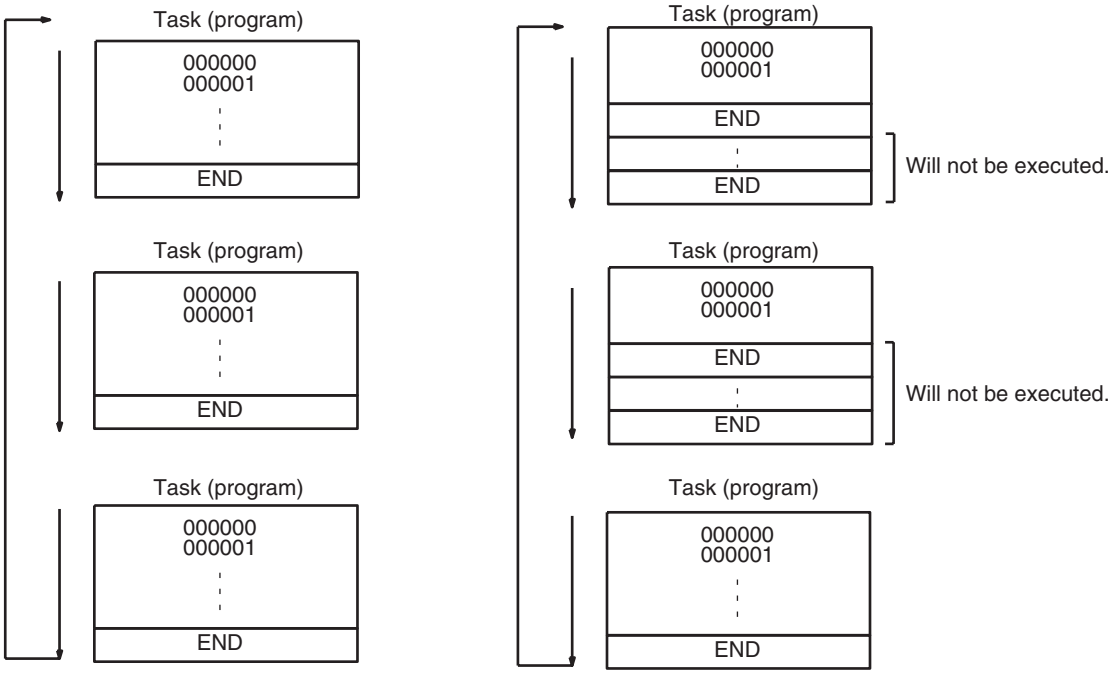


5. An input bit cannot be used in an OUTPUT instruction (OUT).



6. An END(001) instruction must be inserted at the end of the program in each task.
   - If a program without an END(001) instruction starts running, a program error indicating No End Instruction will occur, the ERR/ALM LED on the front of the CPU Unit will light, and the program will not be executed.
   - If a program has more than one END(001) instruction, then the program will only run until the first END(001) instruction.

- Debugging programs will run much smoother if an END(001) instruction is inserted at various break points between sequence rungs and the END(001) instruction in the middle is deleted after the program is checked.

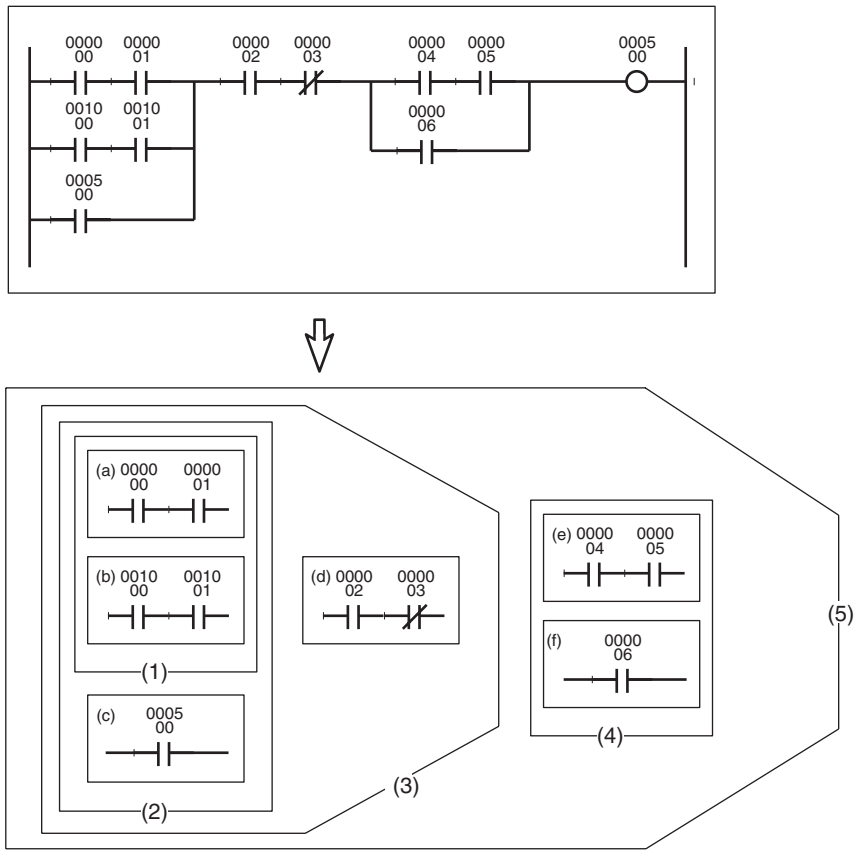


2-1-13 Inputting Mnemonics

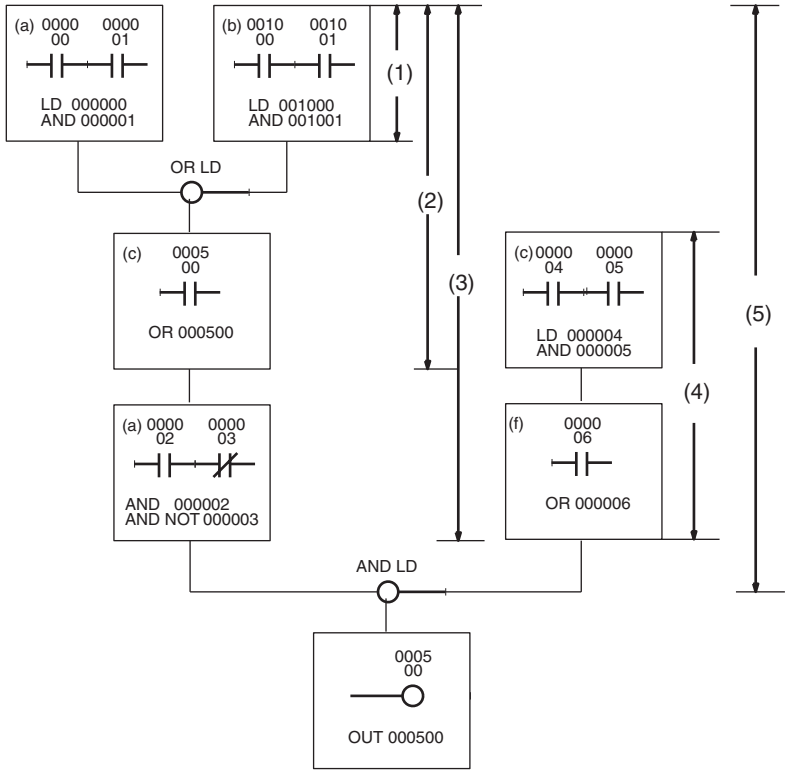
A logical start is accomplished using an LD/LD NOT instruction. The area from the logical start until the instruction just before the next LD/LD NOT instruction is considered a single instruction block.

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

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



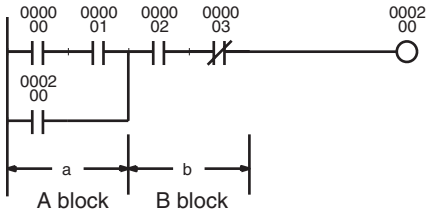
- Program the blocks from top to bottom and then from left to right.



	Address	Instruction	Operand
(a)	000200	LD	000000
	000201	AND	000001
(b)	000202	LD	001000
	000203	AND	001001
	000204	OR LD	---
(c)	000205	OR	000500
(d)	000206	AND	000002
	000207	AND NOT	000003
(e)	000208	LD	000004
	000209	AND	000005
(f)	000210	OR	000006
	000211	AND LD	---
	000212	OUT	000500

2-1-14 Program Examples

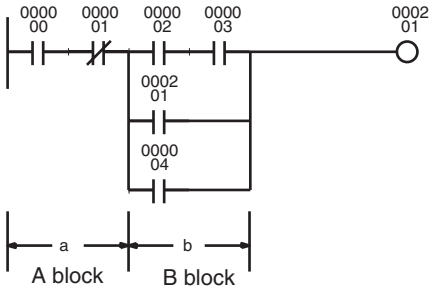
1,2,3... 1. Parallel/Series Rungs



Instruction	Operands	
LD	000000	a
AND	000001	
OR	000200	
AND	000002	b
AND NOT	000003	
OUT	000200	

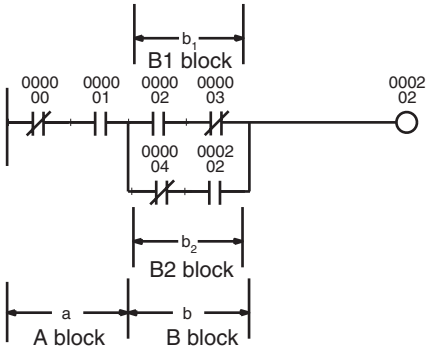
Program the parallel instruction in the A block and then the B block.

2. Series/Parallel Rungs



Instruction	Operands	
LD	000000	a
AND NOT	000001	
LD	000002	b
AND	000003	
OR	000201	
OR	000004	
AND LD	---	
OUT	000201	

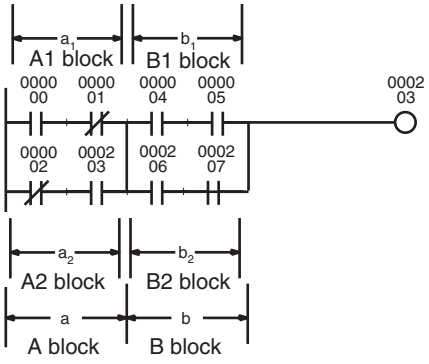
- Separate the rung into A and B blocks, and program each individually.
- Connect A and B blocks with an AND LD.
- Program A block.



Instruction	Operands	
LD NOT	000000	a
AND	000001	
LD	000002	b_1
AND NOT	000003	
LD NOT	000004	b_2
AND	000202	
OR LD	---	$b_1 + b_2$
AND LD	---	$a \cdot b$
OUT	000202	

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

3. Example of series connection in a series rung



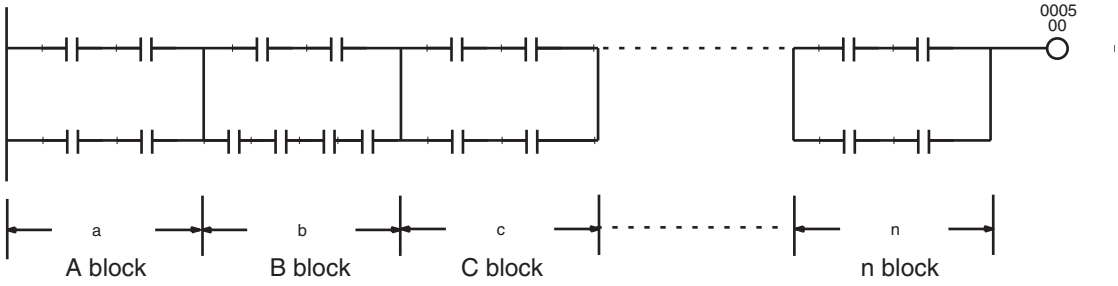
Instruction	Operands	
LD	000000	a_1
AND NOT	000001	
LD NOT	000002	a_2
AND	000003	
OR LD	---	$a_1 + a_2$
LD	000004	b_1
AND	000005	
LD	000006	b_2
AND	000007	
OR LD	---	$b_1 + b_2$
AND LD	---	$a \cdot b$
OUT	000203	

Program A_1 block, program A_2 block, and then connect A_1 and A_2 blocks with an OR LD.

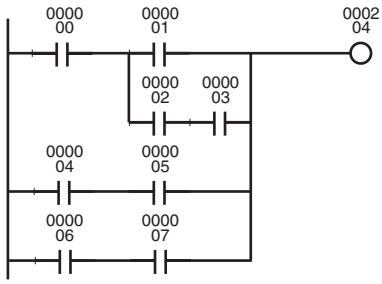
Program B_1 and B_2 the same way.

Connect A block and B block with an AND LD.

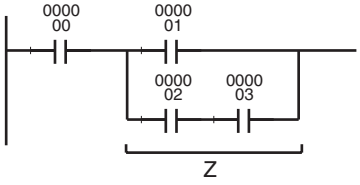
Repeat for as many A to n blocks as are present.



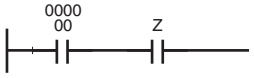
4. Complex Rungs



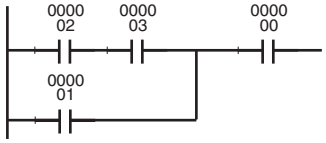
Instruction	Operand	
LD	000000	
LD	000001	Z
LD	000002	Z
AND	000003	Z
OR LD	---	Z
AND LD	---	
LD	000004	
AND	000005	
OR LD	---	
LD	000006	
AND	000007	
OR LD	---	
OUT	000204	

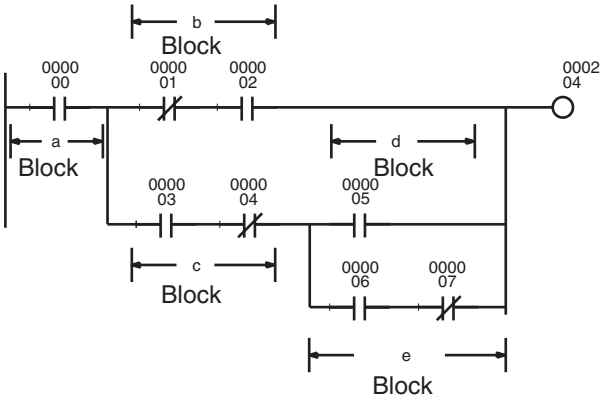


The diagram above is based on the diagram below.

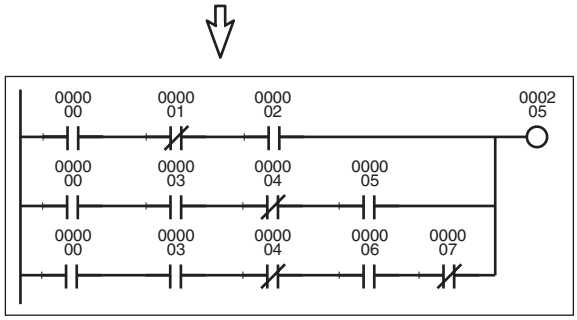


A simpler program can be written by rewriting this as shown below.

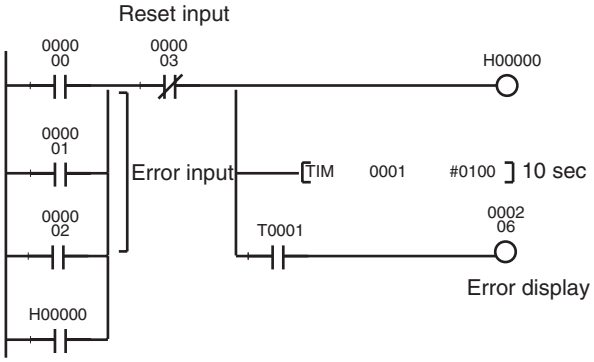




The above rung can be rewritten as follows:



Instruction	Operand	
LD	000000	a
LD NOT	000001	b
AND	000002	b
LD	000003	c
AND NOT	000004	c
LD	000005	d
LD	000006	d
AND NOT	000007	e
OR LD	---	e
AND LD	---	d + e
OR LD	---	(d + e) · c
AND LD	---	(d + e) · c + b
OUT	000205	((d + e) · c + b) · a



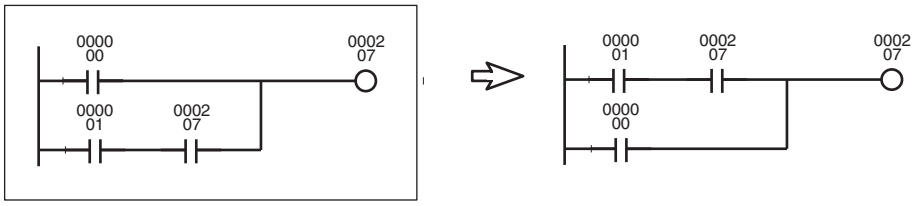
Instruction	Operand
LD	000000
OR	000001
OR	000002
OR	H00000
AND NOT	000003
OUT	H00000
TIM	0001
	0100
AND	T0001
OUT	000206

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

5. Rungs Requiring Caution or Rewriting

OR and OL LD Instructions

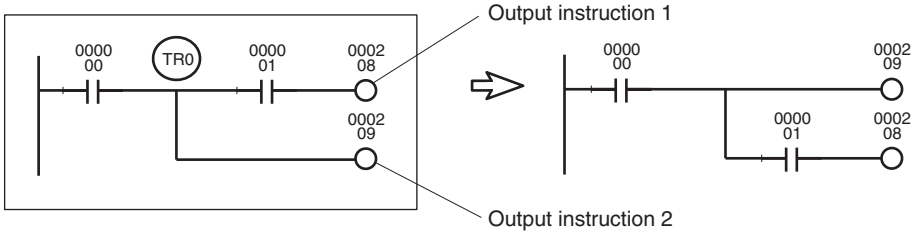
With an OR or OR NOT instruction, an OR is taken with the results of the ladder logic from the LD or LD NOT instruction to the OR or OR NOT instruction, so the rungs can be rewritten so that the OR LD instruction is not required.



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

Output Instruction Branches

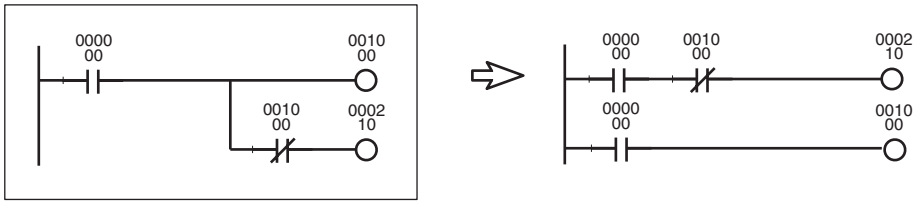
A TR bit will be needed if there is a branch before an AND or AND NOT instruction. The TR bit will not be needed if the branch comes at a point that is connected directly to output instructions and the AND or AND NOT instruction or the output instructions can be continued as is.



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

Mnemonic Execution Order

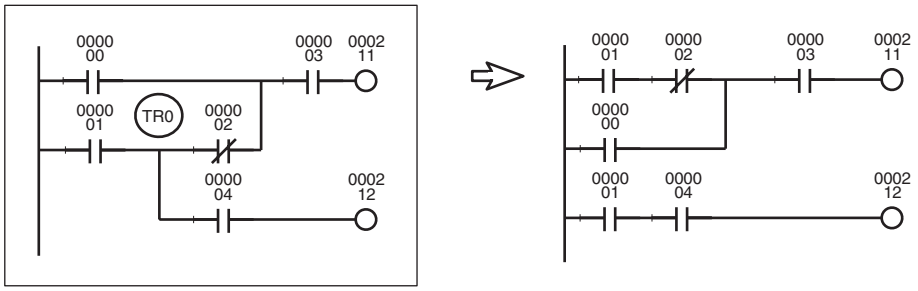
PLCs execute ladder programs in the order the mnemonics are entered so instructions may not operate as expected, depending on the way rungs are written. Always consider mnemonic execution order when writing ladder diagrams.



Example: CIO 000210 in the above diagram cannot be output. By rewriting the rung, as shown above, CIO 000210 can be turned ON for one cycle.

Rungs Requiring Rewriting

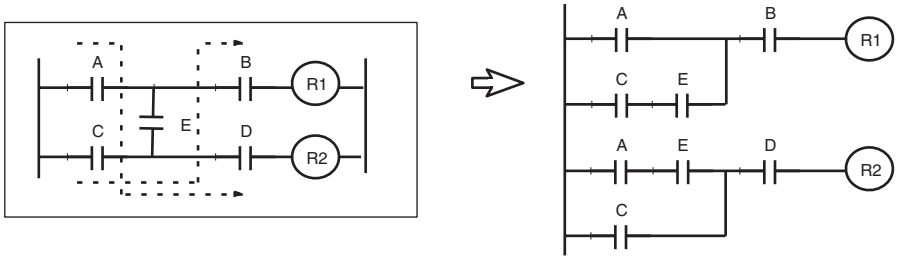
PLCs execute instructions in the order the mnemonics are entered so the signal flow (power flow) is from left to right in the ladder diagram. Power flows from right to left cannot be programmed.



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

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

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



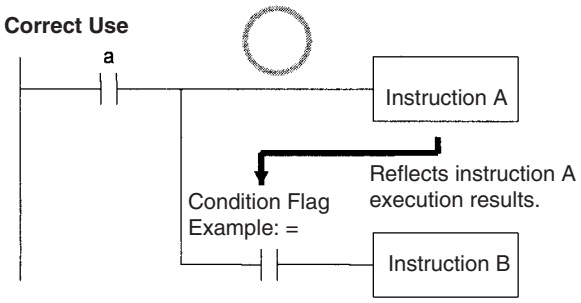
2-2 Precautions

2-2-1 Condition Flags

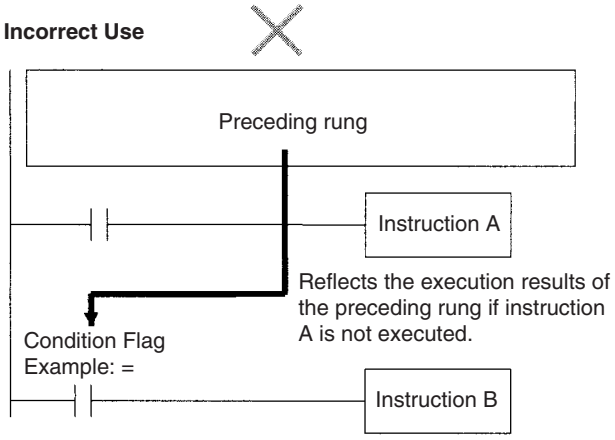
Using Condition Flags

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

Example: Using Instruction A Execution Results



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

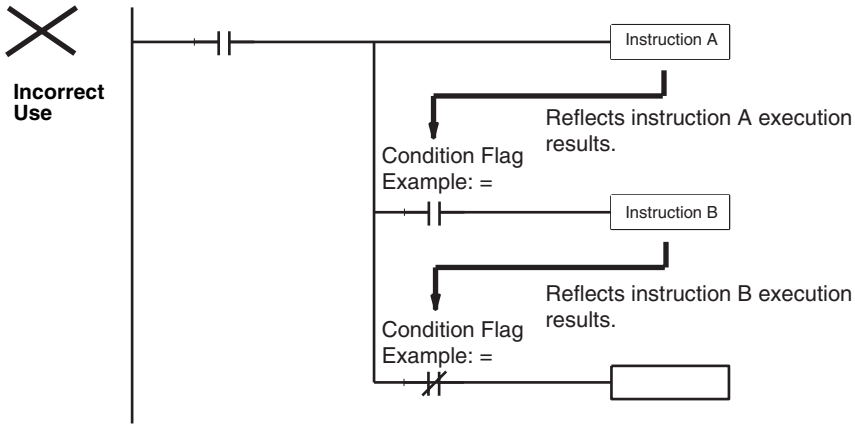


If the Condition Flag is connected directly to the left bus bar, instruction B will be executed based on the execution results of a previous rung if instruction A is not executed.

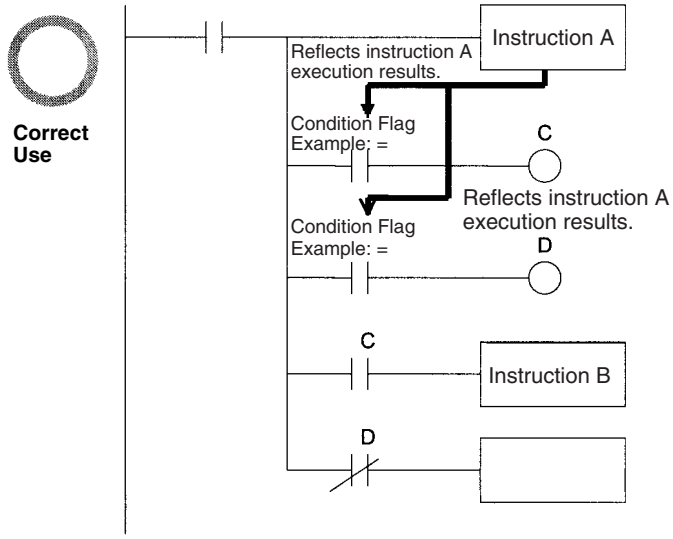
Note When interrupt tasks are being used, an interrupt task will operate when its start conditions are met, even during execution of a cyclic task. In this case, the Condition Flags are returned to their original status when processing switches back from the interrupt task to the cyclic task, even if those flags were turned ON/OFF in the interrupt task.

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

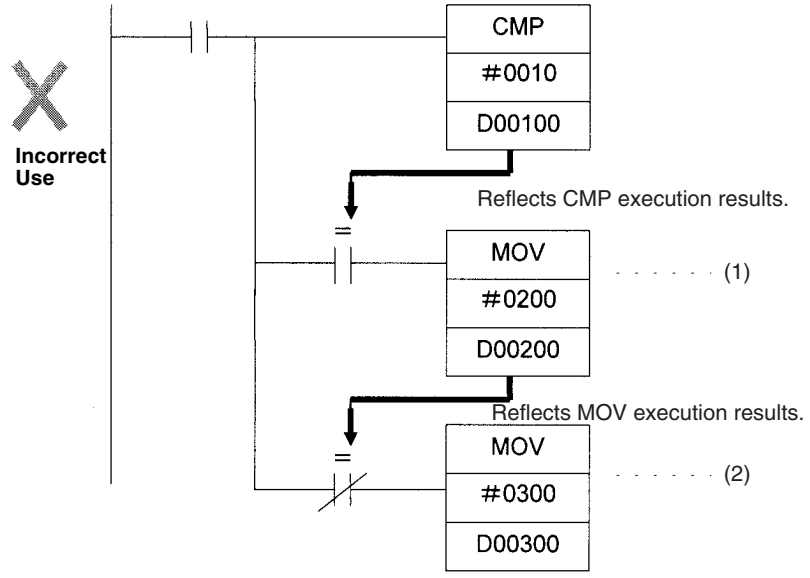
The Condition Flags will pick up instruction B execution results as shown in the example below even though the N.C. and N.O. input bits are executed from the same output branch.



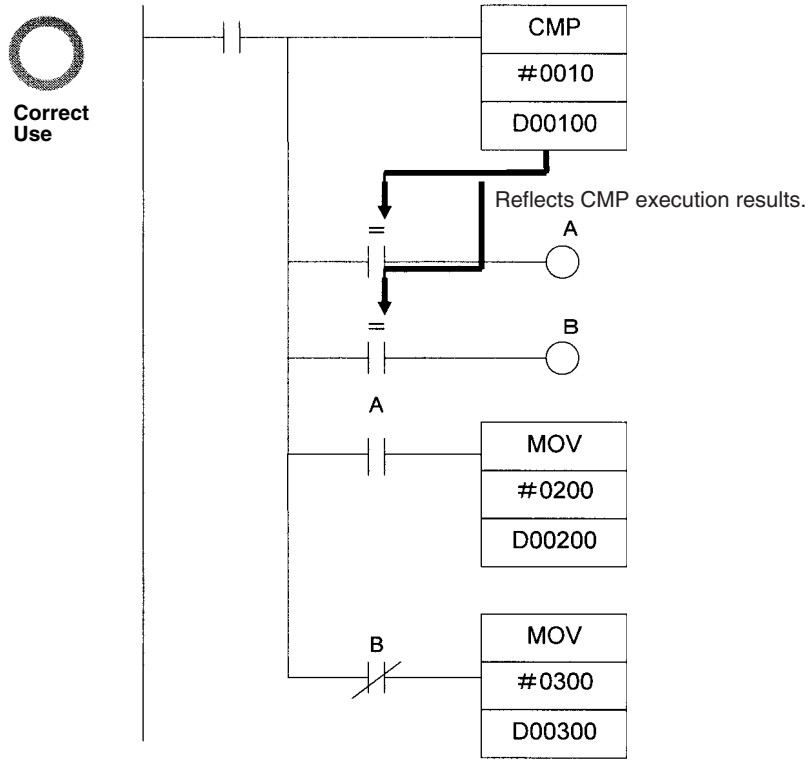
Make sure each of the results is picked up once by an OUTPUT instruction to ensure that execution results for instruction B will be not be picked up.



Example: The following example will move #0200 to D00200 if D00100 contains #0010 and move #0300 to D00300 if D00100 does not contain #0010.



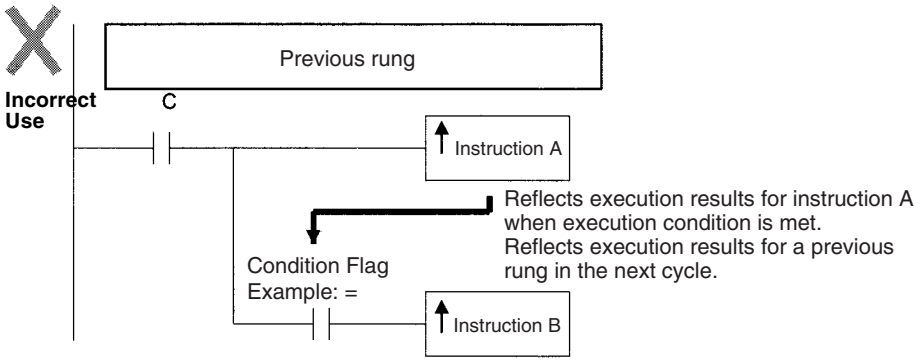
The Equals Flag will turn ON if D00100 in the rung above contains #0010. #0200 will be moved to D00200 for instruction (1), but then the Equals Flag will be turned OFF because the #0200 source data is not 0000 Hex. The MOV instruction at (2) will then be executed and #0300 will be moved to D0300. A rung will therefore have to be inserted as shown below to prevent execution results for the first MOVE instruction from being picked up.



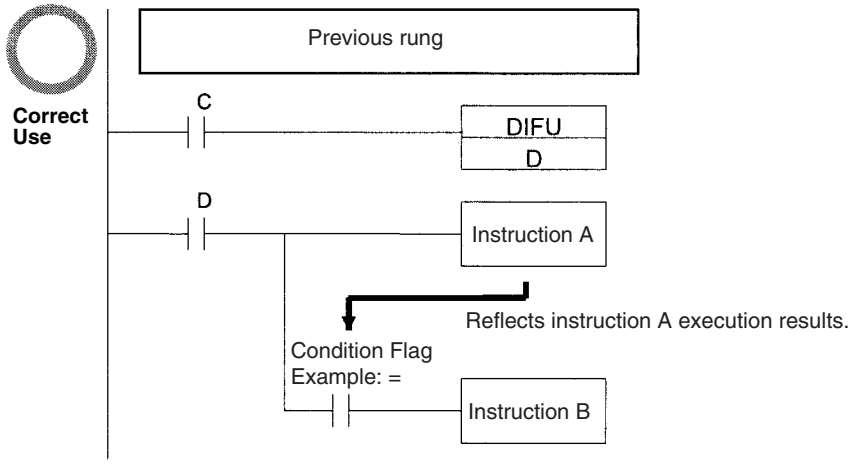
Using Execution Results from Differentiated Instructions

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

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



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



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

The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support the CCS and CCL instructions.

Main Conditions Turning ON Condition Flags

Error Flag (P_ER)

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

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

Refer to the descriptions of individual instructions in the *CS/CJ-series Programmable Controllers Instructions Reference Manual (W340)* for the conditions that will cause the ER Flag to turn ON. Caution is required because some instructions will turn OFF the ER Flag regardless of conditions.

Note The PLC Setup Settings for when an instruction error occurs determines whether operation will stop when the ER Flag turns ON. In the default setting, operation will continue when the ER Flag turns ON. If Stop Operation is specified when the ER Flag turns ON and operation stops (treated as a program error), the program address at the point where operation stopped will be stored at in A298 to A299. At the same time, A29508 will turn ON.

Equals Flag (P_EQ)

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

Carry Flag (P_CY)

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

Note

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

Less Than and Greater Than Flags (P_LT, P_GT)

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

Negative Flag (P_N)

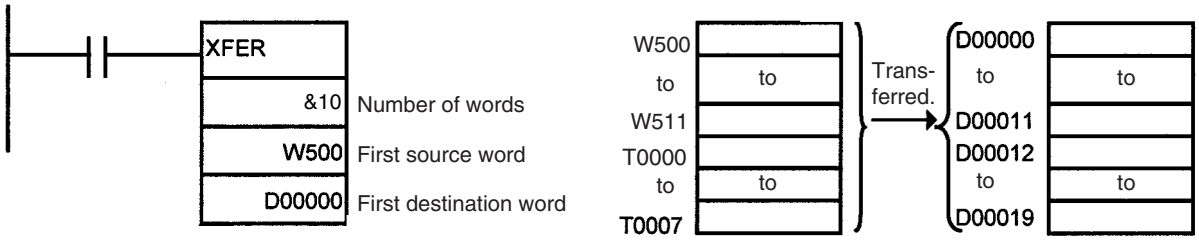
The N Flag is turned OFF when the leftmost bit of the instruction execution results word is “1” for certain instructions and it is turned OFF unconditionally for other instruction.

Specifying Operands for Multiple Words

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

As an example, consider the results of executing a block transfer with XFER(070) if 20 words are specified for transfer beginning with W500. Here, the Work Area, which ends at W511, will be exceeded, but the instruction will be executed without turning ON the Error Flag. In the PLC memory addresses, the present values for timers are held in memory after the Work Area, and thus for the following instruction, W500 to W511 will be transferred to D00000 to D00011 and the present values for T0000 to T0007 will be transferred to D00012 to D00019.

Note Refer to the appendix *Memory Map of PLC Memory Addresses* for specific PLC memory addresses.



2-2-2 Special Program Sections

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

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

Instruction Combinations

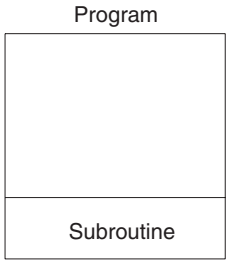
The following table shows which of the special instructions can be used inside other program sections.

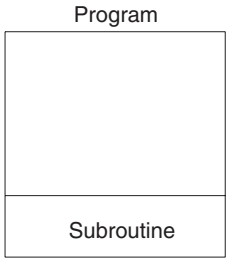
	Subroutine	IL - ILC section	Step ladder section	FOR - NEXT loop	JMP0 - JME0 section	Block program section
Subroutine	Not possible.	Not possible.	Not possible.	Not possible.	Not possible.	Not possible.
IL - ILC	**OK**	Not possible.	Not possible.	**OK**	**OK**	Not possible.
Step ladder section	Not possible.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

Note Instructions that specify program areas cannot be used for programs in other tasks. Refer to *4-2-2 Task Instruction Limitations* for details.

Subroutines

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





Instructions Not Available in Subroutines

The following instructions cannot be placed in a subroutine.

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

Note **Block Program Sections**

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

Instructions Not Available in Step Ladder Program Sections

Function	Mnemonic	Instruction
Sequence Control	FOR(512), NEXT(513), and BREAK(514)	FOR, NEXT, and BREAK LOOP
	END(001)	END
	IL(002) and ILC(003)	INTERLOCK and INTERLOCK CLEAR
	JMP(004) and JME(005)	JUMP and JUMP END
	CJP(510) and CJPN(511)	CONDITIONAL JUMP and CONDITIONAL JUMP NOT
	JMP0(515) and JME0(516)	MULTIPLE JUMP and MULTIPLE JUMP END
Subroutines	SBN(092), RET(093), GSBN (751) and GRET(752)	SUBROUTINE ENTRY and SUBROUTINE RETURN
Block Programs	IF(802) (NOT), ELSE(803), and IEND(804)	Branching instructions
	BPRG(096) and BEND(801)	BLOCK PROGRAM BEGIN/ END
	EXIT(806) (NOT)	CONDITIONAL BLOCK EXIT (NOT)
	LOOP(809) and LEND(810) (NOT)	Loop control
	WAIT(805) (NOT)	ONE CYCLE WAIT (NOT)
	TIMW(813)	TIMER WAIT
	TMHW(815)	HIGH-SPEED TIMER WAIT
	CNTW(814)	COUNTER WAIT
	BPPS(811) and BPRS(812)	BLOCK PROGRAM PAUSE and RESTART

Note

1. A step ladder program section can be used in an interlock section (between IL and ILC). The step ladder section will be completely reset when the interlock is ON.
2. A step ladder program section can be used between MULTIPLE JUMP (JMP0) and MULTIPLE JUMP END (JME0).

Instructions Not Supported in Block Program Sections

The following instructions cannot be placed in block program sections.

Classification by Function	Mnemonic	Instruction
Sequence Control	FOR(512), NEXT(513), and BREAK(514)	FOR, NEXT, and BREAK LOOP
	END(001)	END
	IL(002) and ILC(003)	INTERLOCK and INTERLOCK CLEAR
	JMP0(515) and JME0(516)	MULTIPLE JUMP and MULTIPLE JUMP END
Sequence Input	UP(521)	CONDITION ON
	DOWN(522)	CONDITION OFF
Sequence Output	DIFU	DIFFERENTIATE UP
	DIFD	DIFFERENTIATE DOWN
	KEEP	KEEP
	OUT	OUTPUT
	OUT NOT	OUTPUT NOT
Timer/Counter	TIM	HUNDRED-MS TIMER
	TIMH	TEN-MS TIMER
	TMHH(540)	ONE-MS TIMER
	TIMU (See note 5.)	TENTH-MS TIMER
	TMUH (See note 5.)	HUNDREDTH-MS TIMER
	TTIM(087)	ACCUMULATIVE TIMER
	TIML(542)	LONG TIMER
	MTIM(543)	MULTI-OUTPUT TIMER
	CNT	COUNTER
	CNTR	REVERSIBLE COUNTER
Subroutines	SBN(092) and RET(093)	SUBROUTINE ENTRY and SUBROUTINE RETURN
Data Shift	SFT	SHIFT
Ladder Step Control	STEP(008) and SNXT(009)	STEP DEFINE and STEP START
Data Control	PID	PID CONTROL
Block Program	BPRG(096)	BLOCK PROGRAM BEGIN
Damage Diagnosis	FPD(269)	FAILURE POINT DETECTION

Note

1. Block programs can be used in a step ladder program section.
2. A block program can be used in an interlock section (between IL and ILC). The block program section will not be executed when the interlock is ON.
3. A block program section can be used between MULTIPLE JUMP (JMP0) and MULTIPLE JUMP END (JME0).
4. A JUMP instruction (JMP) and CONDITIONAL JUMP instruction (CJP/CJPN) can be used in a block program section. JUMP (JMP) and JUMP END (JME) instructions, as well as CONDITIONAL JUMP (CJP/CJPN) and JUMP END (JME) instructions cannot be used in the block program section unless they are used in pairs. The program will not execute properly unless these instructions are paired.
5. These instructions are supported by the CJ1-H-R CPU Units only.

2-3 Checking Programs

CS/CJ-series programs can be checked at the following stages.

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

2-3-1 Errors during Programming Device Input

CX-Programmer

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

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

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

Programming Console

Errors at the following points will be displayed on the Programming Console during input.

Error display	Cause
CHK MEM	Pin 1 on the DIP switch on the CPU Unit is set to ON (write-protect).
IO No. ERR	An illegal I/O input has been attempted.

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

The user program can be checked in the CX-Programmer. When the program is checked, the user can specify program check in any of four levels: A, B, or C (in order of the seriousness of the errors) or a custom check level.

The CX-Programmer does not check range errors for indirectly addressed operands in instructions. If an instruction's operand data is invalid, the ER Flag will be turned ON during the program execution check, which is described in the next section. For details, refer to the *CS/CJ-series Programmable Controllers Instructions Reference Manual* (W340).

For details on the CX-Programmer's checks, refer to the *CX-Programmer Operation Manual* (W446).

2-3-3 Program Execution Check

Operand and instruction location checks are performed on instructions during input from Programming Devices (including Programming Consoles) as well as during program checks from Programming Devices (excluding Programming Consoles). However, these are not final checks.

The following checks are performed during instruction execution.

Type of error	Flag that turns ON for error	Stop/Continue operation
1.Instruction Processing Error	ER Flag The Instruction Processing Error Flag (A29508) will also turn ON if Stop Operation is specified when an error occurs.	A setting in the PLC Setup can be used to specify whether to stop or continue operation for instruction processing errors. The default is to continue operation. A program error will be generated and operation will stop only if Stop Operation is specified.
2.Access Error	AER Flag The Access Error Flag (A29510) will turn ON if Stop Operation is specified when an error occurs.	A setting in the PLC Setup can be used to specify whether to stop or continue operation for instruction processing errors. The default is to continue operation. A program error will be generated and operation will stop only if Stop Operation is specified.
3.Illegal Instruction Error	Illegal Instruction Error Flag (A29514)	Fatal (program error)
4.UM (User Memory) Overflow Error	UM Overflow Error Flag (A29515)	Fatal (program error)

Instruction Processing Errors

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

The ER Flag (error Flag) will turn OFF if the instruction (excluding input instructions) ends normally. Conditions that turn ON the ER Flag will vary with individual instructions. See descriptions of individual instructions in the *CS/CJ-series Programmable Controllers Programming Manual (W340)* for more details.

If Instruction Errors are set to Stop Operation in the PLC Setup, then operation will stop (fatal error) and the Instruction Processing Error Flag (A29508) will turn ON if an instruction processing error occurs and the ER Flag turns ON.

Illegal Access Errors

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

a) A read or write was executed for a parameter area.

b) A write was executed in a memory area that is not mounted (see note).

c) A write was executed in an EM area specified as EM File Memory.

d) A write was executed in a read-only area.

e) The value specified in an indirect DM/EM address in BCD mode was not BCD (e.g., *D000001 contains #A000).

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

Note An access error will occur for the following:

- When a specified EM address exceeds 32767 (example: E32768) for the current bank.

- The final bank (example: C) is specified for an indirect EM address in BIN mode and the specified word contains 8000 to FFFF Hex (example: @EC_00001 contains #8000).
- The current bank (example: C) is specified for an indirect EM address in BIN mode and the specified words contains 8000 to FFFF Hex (example: @EC_00001 contains #8000)
- An IR register containing the internal memory address of a bit is used as a word address or an IR containing the internal memory address of a word is used as a bit address.

If Instruction Errors are set to Stop Operation in the PLC Setup, then operation will stop (fatal error) and the "Illegal Access Error Flag" (A29510) will turn ON if an illegal access error occurs and the AER Flag turns ON.

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

Other Errors

Illegal Instruction Errors

Illegal instruction errors indicate that an attempt was made to execute instruction data other than that defined in the system. This error will normally not occur as long as the program is created on a CS/CJ-series Programming Device (including Programming Consoles).

In the rare even that this error does occur, it will be treated as a program error, operation will stop (fatal error), and the Illegal Instruction Flag (A29514) will turn ON.

UM (User Memory) Overflow Errors

UM overflow errors indicate that an attempt was made to execute instruction data stored beyond the last address in the user memory (UM) defined as program storage area. This error will normally not occur as long as the program is created on a CS/CJ-series Programming Device (including Programming Consoles).

In the rare even that this error does occur, it will be treated as a program error, operation will stop (fatal error), and the UM Overflow Flag (A29515) will turn ON.

2-3-4 Checking Fatal Errors

The following errors are fatal program errors and the CPU Unit will stop running if one of these occurs. When operation is stopped by a program error, the task number where operation stopped will be stored in A294 and the program

address will be stored in A298/A299. The cause of the program error can be determined from this information.

Address	Description	Stored Data
A294	The type of task and the task number at the point where operation stopped will be stored here if operation stops due to a program error. FFFF Hex will be stored if there are no active cyclic tasks in a cycle, i.e., if there are no cyclic tasks to be executed.	Cyclic task: 0000 to 001F Hex (cyclic tasks 0 to 31) Interrupt task: 8000 to 80FF Hex (interrupt tasks 0 to 255)
A298/A299	The program address at the point where operation stopped will be stored here in binary if operation stops due to a program error. If the END(001) instruction is missing (A29511 will be ON), the address where END(001) was expected will be stored. If there is a task execution error (A29512 will be ON), FFFFFFFF Hex will be stored in A298/A299.	A298: Rightmost portion of program address A299: Leftmost portion of program address

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

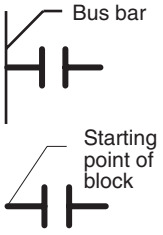
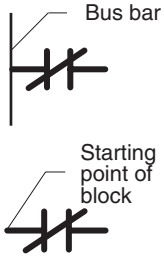
Program error	Description	Related flags
No END Instruction	An END instruction is not present in the program.	The No END Flag (A29511) turns ON.
Error During Task Execution	No task is ready in the cycle. No program is allocated to a task. The corresponding interrupt task number is not present even though the execution condition for the interrupt task was met.	The Task Error Flag (29512) turns ON.
Instruction Processing Error (ER Flag ON) and Stop Operation set for Instruction Errors in PLC Setup	The wrong data values were provided in the operand when an attempt was made to execute an instruction.	The ER Flag turns ON and the Instruction Processing Error Flag (A29508) turns ON if Stop Operation set for Instruction Errors in PLC Setup.
Illegal Access Error (AER Flag ON) and Stop Operation set for Instruction Errors in PLC Setup	A read or write was executed for a parameter area. A write was executed in a memory area that is not mounted (see note). A write was executed in an EM area specified as EM File Memory. A write was executed in a read-only area. The value specified in an indirect DM/EM address in BCD mode was not BCD.	AER Flag turns ON and the Illegal Access Error Flag (A29510) turns ON if Stop Operation set for Instruction Errors in PLC Setup
Indirect DM/EM BCD Error and Stop Operation set for Instruction Errors in PLC Setup	The value specified in an indirect DM/EM address in BCD mode is not BCD.	AER Flag turns ON and the DM/EM Indirect BCD Error Flag (A29509) turns ON if Stop Operation set for Instruction Errors in PLC Setup
Differentiation Address Overflow Error	During online editing, more than 131,071 differentiated instructions have been inserted or deleted.	The Differentiation Overflow Error Flag (A29513) turns ON.
UM (User Memory) Overflow Error	An attempt was made to execute instruction data stored beyond the last address in user memory (UM) defined as program storage area.	The UM (User Memory) Overflow Flag (A29516) turns ON.
Illegal Instruction Error	An attempt was made to execute an instruction that cannot be executed.	The Illegal Instruction Flag (A29514) turns ON.

SECTION 3
Instruction Functions

This section outlines the instructions that can be used to write user programs.

3-1 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
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
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
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
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
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
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

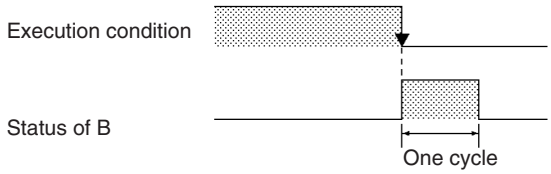
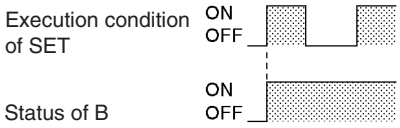
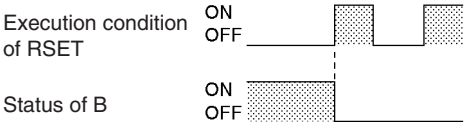
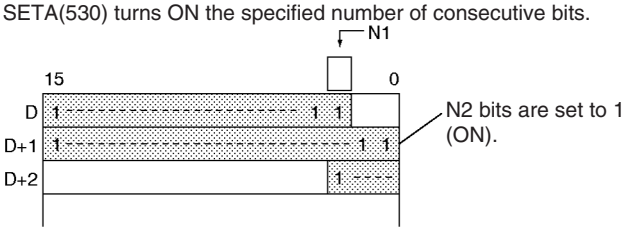
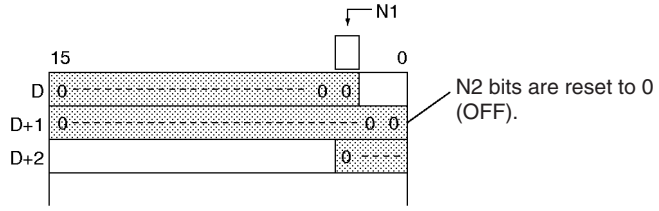
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
AND LOAD AND LD	Logic block Logic block	Takes a logical AND between logic blocks. LD to) Logic block A LD to) Logic block B AND LD ····· Serial connection between logic block A and logic block B.	Continues on rung Required
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
NOT NOT 520	---	Reverses the execution condition.	Continues on rung Required
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
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
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
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
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
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

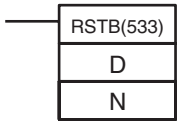
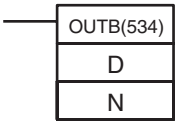
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
BIT TEST OR TST 350	TST(350) S N **S**: Source word **N**: Bit number	LD TST(350), AND TST(350), and OR TST(350) are used in the program like LD, AND, and OR; the execution condition is ON when the specified bit in the specified word is ON and OFF when the bit is OFF.	Continues on rung Required
BIT TEST OR 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 Required

3-2 Sequence Output Instructions

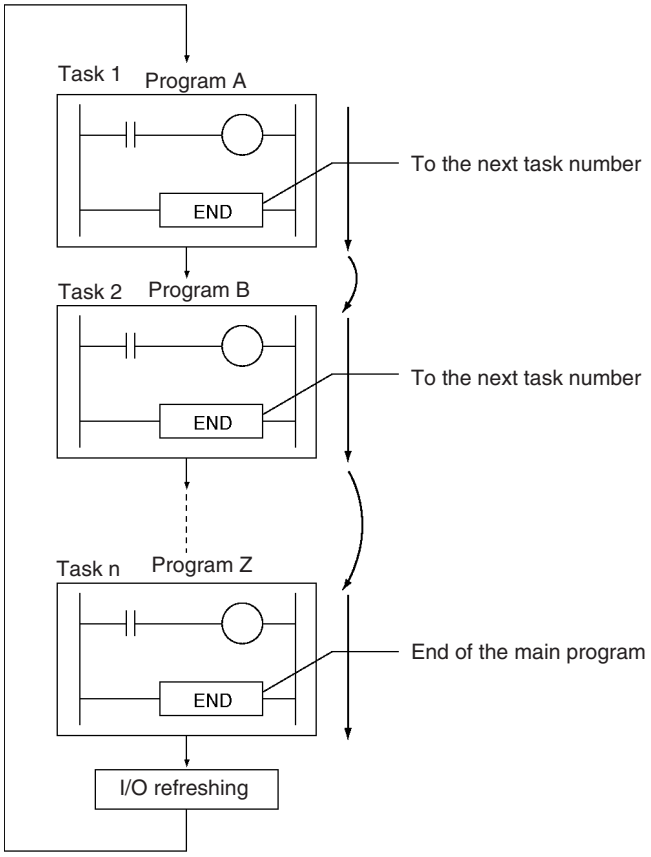
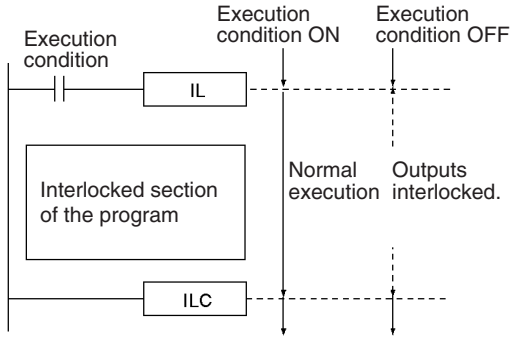
*1: Not supported by CS1D CPU Units for Duplex-CPU Systems.

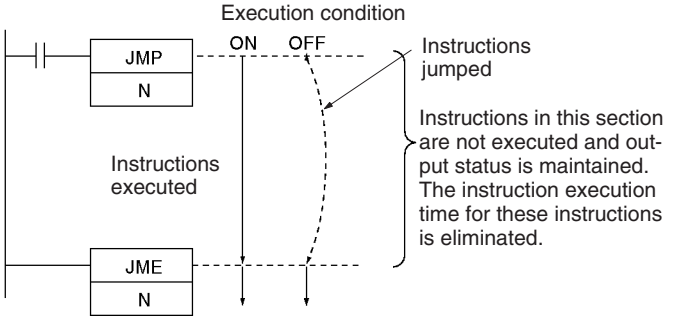
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
OUTPUT OUT !OUT*1		Outputs the result (execution condition) of the logical processing to the specified bit.	Output Required
OUTPUT NOT OUT NOT !OUT NOT*1		Reverses the result (execution condition) of the logical processing,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

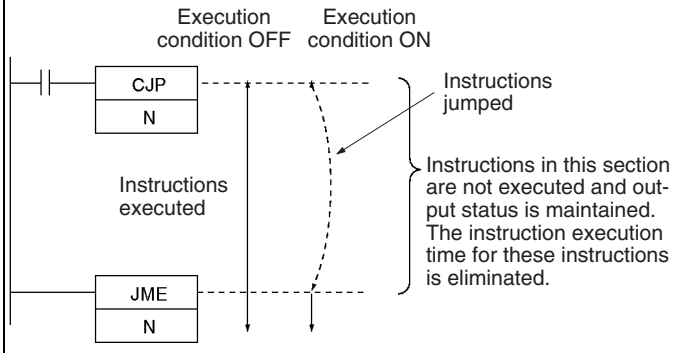
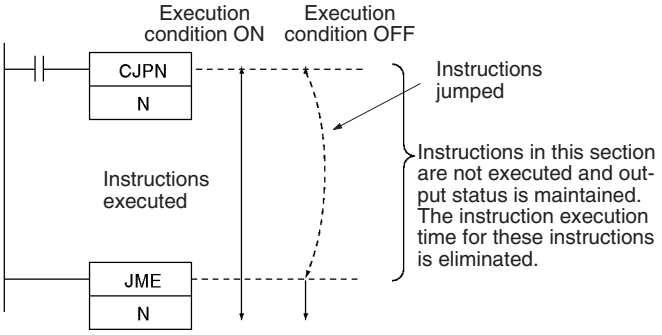
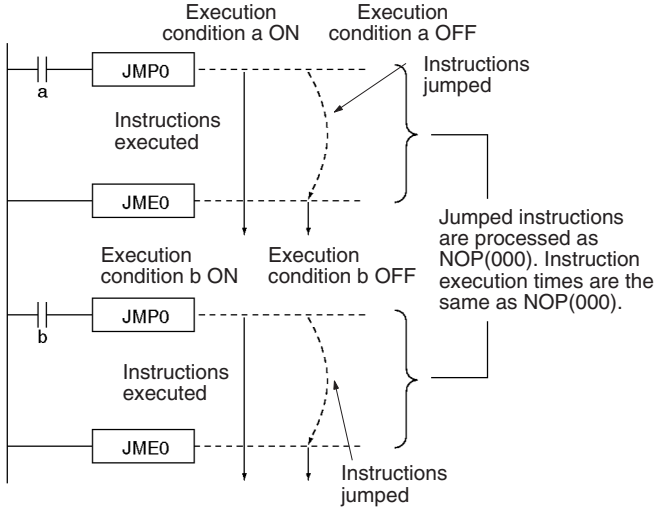
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DIFFERENTIATE DOWN DIFD !DIFD*1 014	DIFD(014) B **B**: Bit	DIFD(014) turns the designated bit ON for one cycle when the execution condition goes from ON to OFF (falling edge). 	Output Required
SET SET @SET %SET !SET*1 !@SET*1 !%SET*1	SET B **B**: Bit	SET turns the operand bit ON when the execution condition is ON. 	Output Required
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
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
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
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

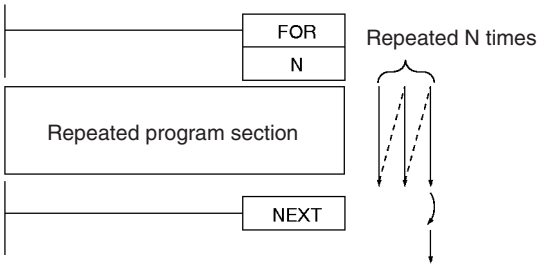
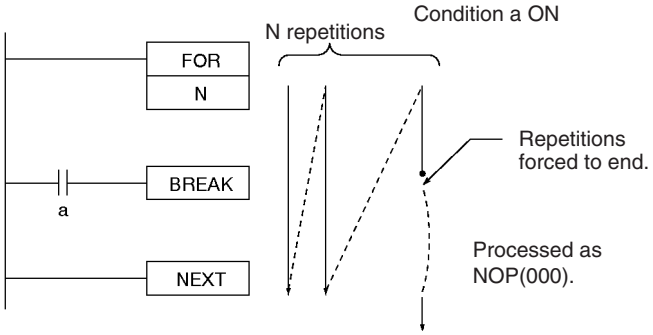
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SINGLE BIT RESET (CS1-H, CJ1-H, CJ1M, or CS1D only) RSTB @RSTB !RSTB*1 !@RSTB*1	 **D**: Word address **N**: Bit number	RSTB(533) turns OFF the specified bit in the specified word when the execution condition is ON. Unlike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

3-3 Sequence Control Instructions

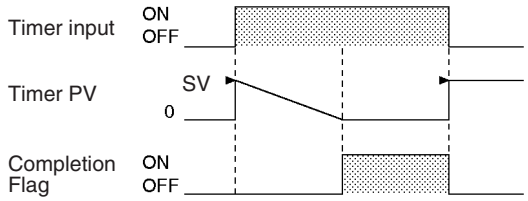
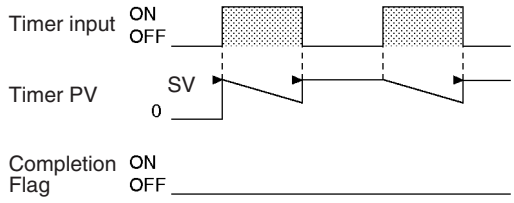
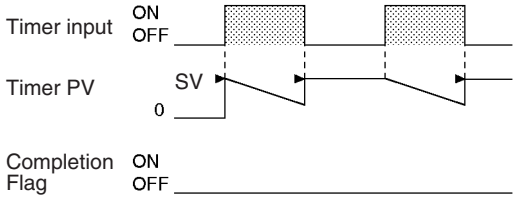
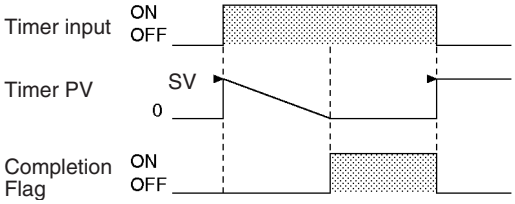
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
END END 001	END(001)	Indicates the end of a program. END(001) completes the execution of a program for that cycle. No instructions written after END(001) will be executed. Execution proceeds to the program with the next task 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
NO OPERATION NOP 000		This instruction has no function. (No processing is performed for NOP(000).)	Output Not required
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

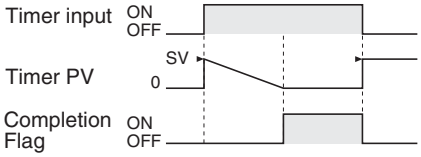
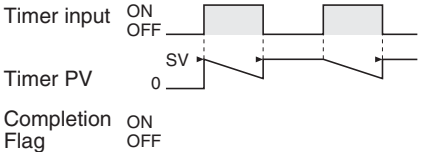
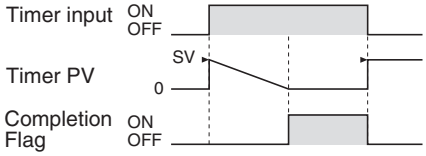
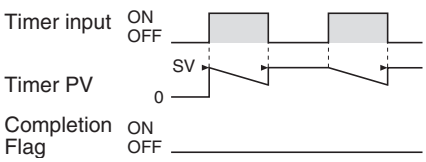
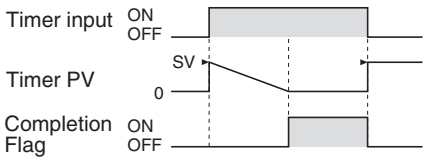
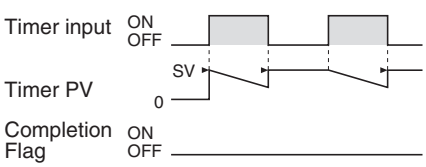
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
INTERLOCK CLEAR ILC 003	ILC(003)	All outputs between IL(002) and ILC(003) are interlocked when the execution condition for IL(002) is OFF. IL(002) and ILC(003) are normally used in pairs.	Output Not required
MULTI-INTERLOCK 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.	Output Required
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
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
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
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

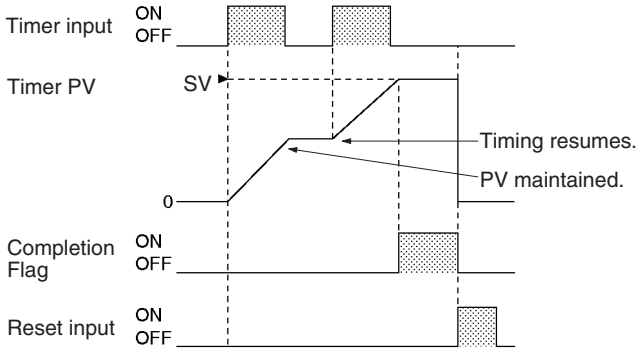
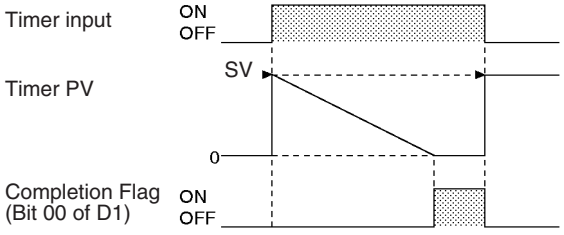
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
CONDITIONAL JUMP CJP 510	CJP(510) N **N**: Jump number	The operation of CJP(510) is the basically the opposite of JMP(004). When the execution condition for CJP(510) is ON, program execution jumps directly to the first JME(005) in the program with the same jump number. CJP(510) and JME(005) are used in pairs. 	Output Required
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
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
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

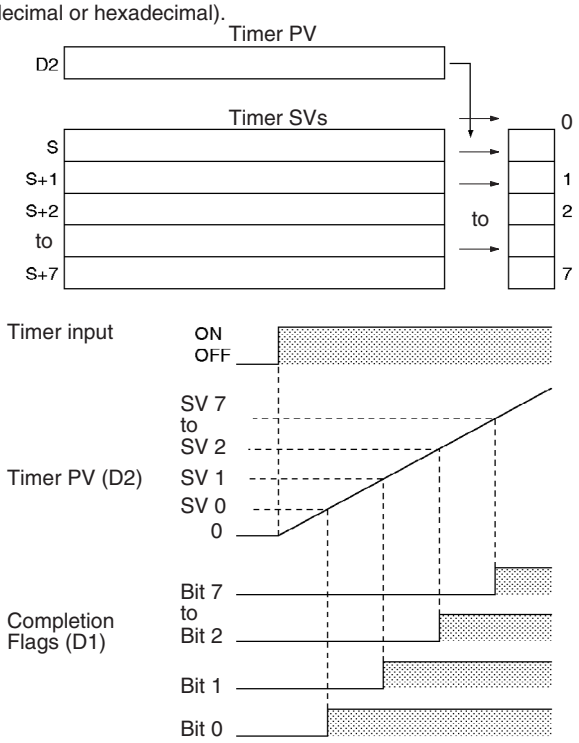
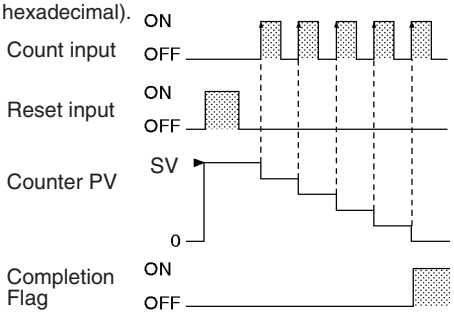
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
FOR-NEXT LOOPS FOR 512	FOR(512) N **N**: Number of loops	The instructions between FOR(512) and NEXT(513) are repeated a specified number of times. FOR(512) and NEXT(513) are used in pairs. 	Output Not required
BREAK LOOP BREAK 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
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

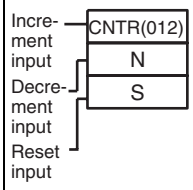
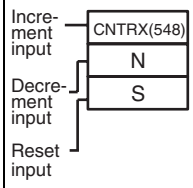
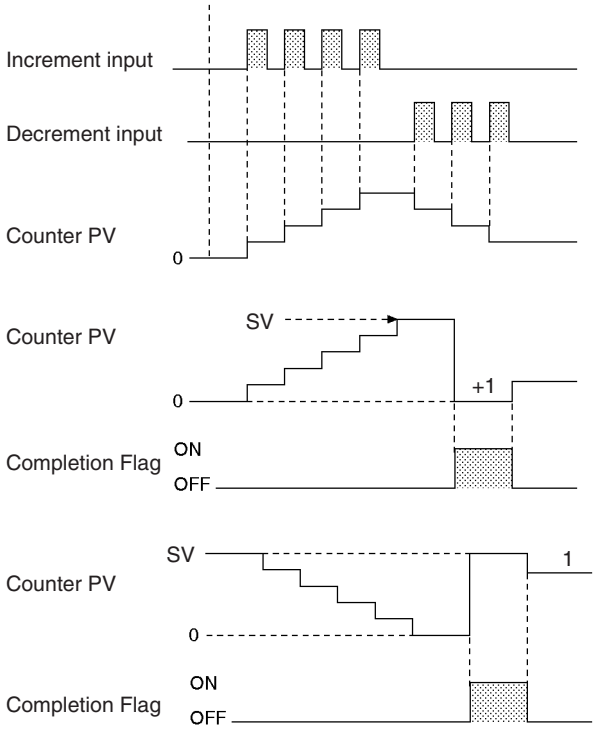
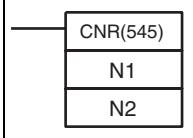
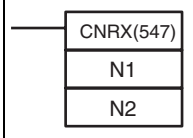
3-4 Timer and Counter Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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).  Timer Input Turned OFF before Completion Flag Turns ON 	Output Required
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).  Timer Input Turned OFF before Completion Flag Turns ON 	Output Required

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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). The timing charts for TMHH(540) and TMHHX(552) are the same as those given above for TIMH(015).  Timer Input Turned OFF before Completion Flag Turns ON 	Output Required
TENTH-MS TIMER **(CJ1-H-R only)** TIMU 541 (BCD) TIMUX 556 (BIN)	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 a decrementing timer with units of 0.1-ms. The setting range for the set value (SV) is 0 to 0.9999 s for BCD and 0 to 6.5535 s for binary (decimal or hexadecimal).  Timer Input Turned OFF before Completion Flag Turns ON 	Output Required
HUNDREDTH-MS TIMER **(CJ1-H-R only)** TMUH (BCD) TIMUX (BIN)	TMUH(544) N S **N**: Timer number **S**: Set value TMUHX(557) N S **N**: Timer number **S**: Set value	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 BCD and 0 to 0.65535 s for binary (decimal or hexadecimal).  Timer Input Turned OFF before Completion Flag Turns ON 	Output Required

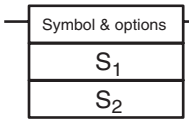
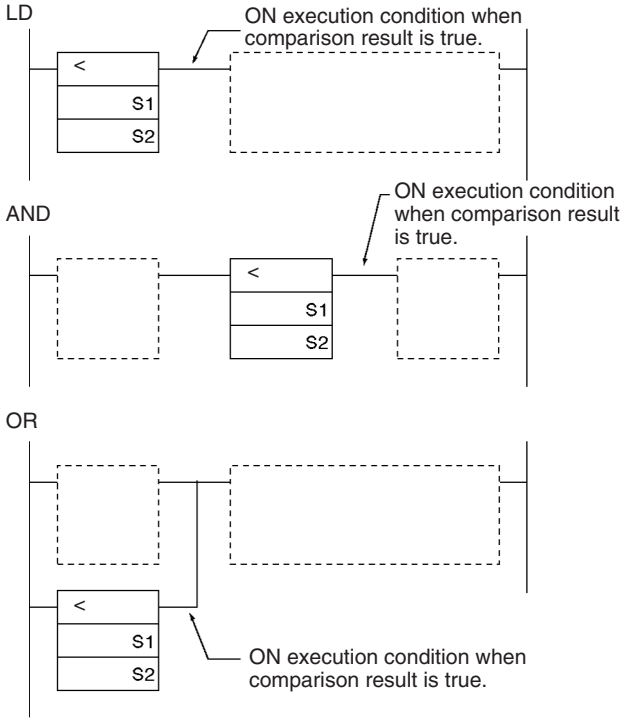
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
ACCUMULATIVE TIMER TTIM 087 (BCD) TTIMX 555 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	Timer input — TTIM(087) / N / S Reset input **N**: Timer number **S**: Set value Timer input — TTIMX(555) / N / S Reset input **N**: Timer 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
LONG TIMER TIML 542 (BCD) TIMLX 553 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	TIML(542) / D1 / D2 / S **D1**: Completion Flag **D2**: PV word **S**: SV word TIMLX(553) / D1 / D2 / S **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 4,971 days for binary (decimal or hexadecimal).  	Output Required

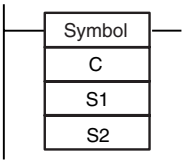
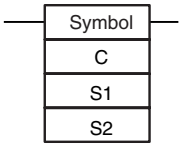
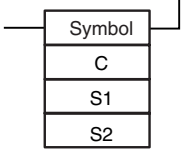
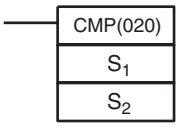
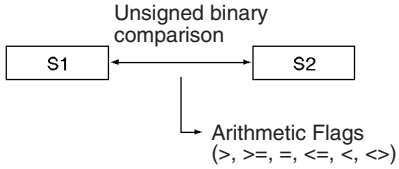
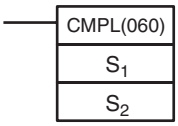
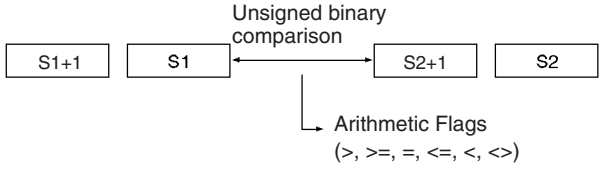
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
MULTI-OUTPUT TIMER MTIM 543 (BCD) MTIMX 554 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	MTIM(543) D1 D2 S **D1**: Completion Flags **D2**: PV word **S**: 1st SV word MTIMX(554) D1 D2 S **D1**: Completion Flags **D2**: PV word **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
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

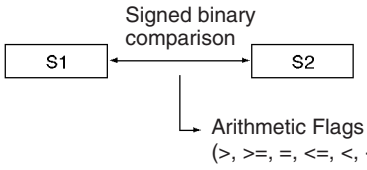
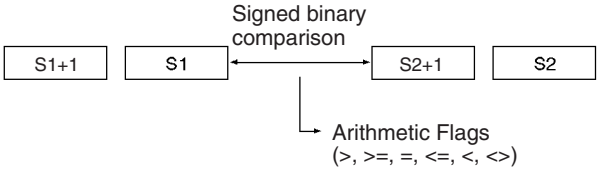
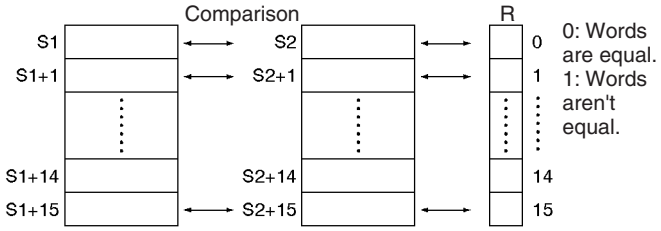
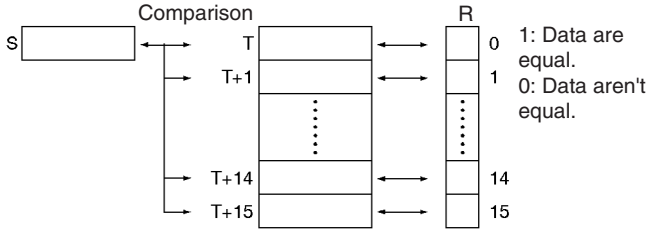
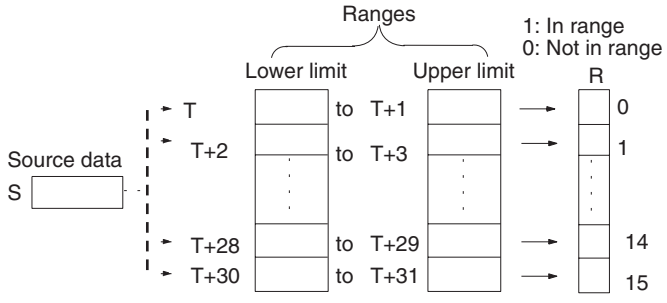
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
REVERSIBLE COUNTER CNTR 012 (BCD) CNTRX 548 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	 **N**: Counter number **S**: Set value  **N**: Counter number **S**: Set value	CNTR(012)/CNTRX(548) operates a reversible counter. 	Output Required
RESET TIMER/ COUNTER CNR @CNR 545 (BCD) CNRX @CNRX 547 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	 **N1**: 1st number in range **N2**: Last number in range  **N1**: 1st number in range **N2**: 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

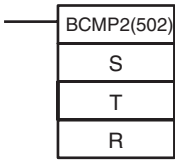
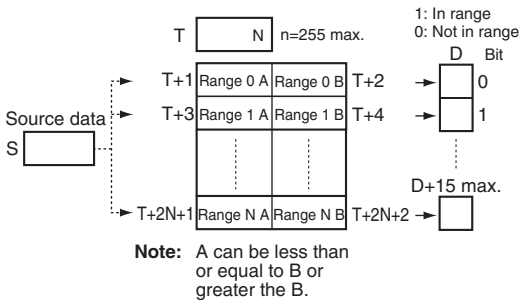
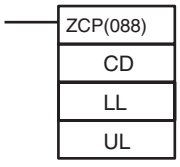
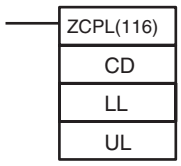
3-5 Comparison Instructions

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

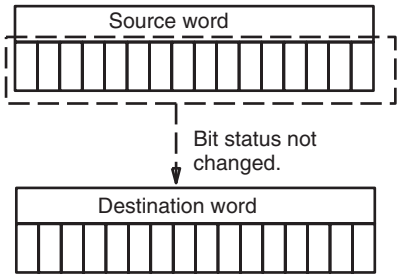
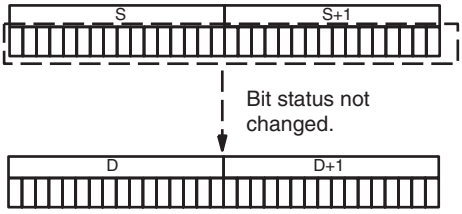
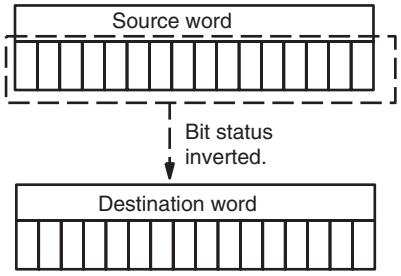
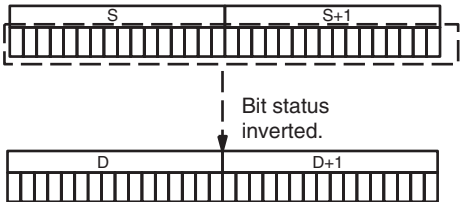
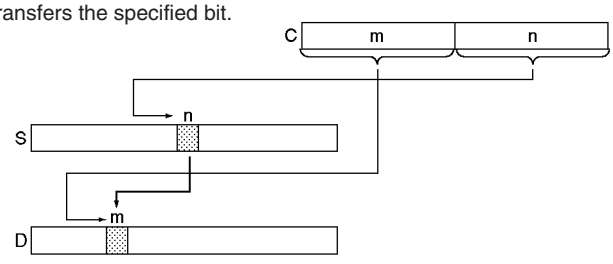
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
Symbol Comparison (Unsigned) LD, AND, OR + =, <>, <, <=, >, >= 300 (=) 305 (<>) 310 (<) 315 (<=) 320 (>) 325(>=)	 **S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (unsigned) compare two 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
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
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

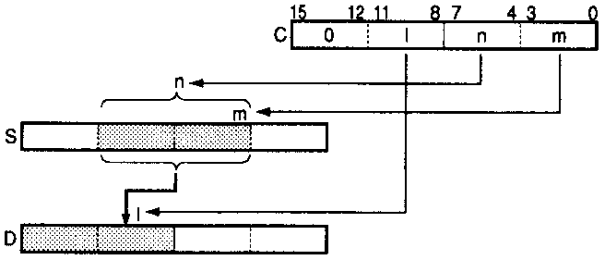
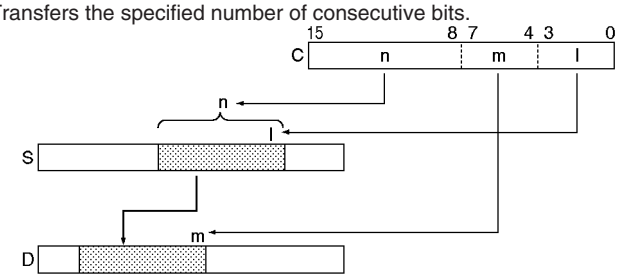
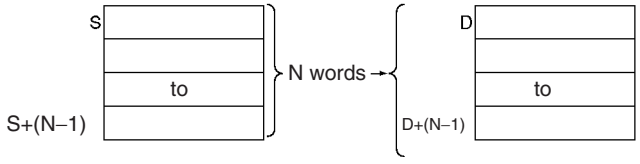
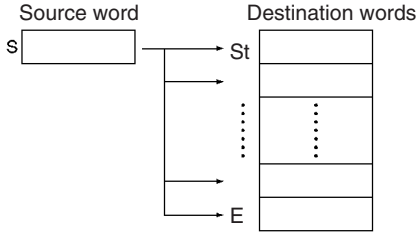
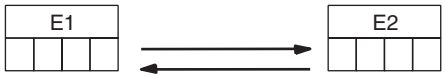
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
Symbol Comparison (Double-word, signed) LD, AND, OR + =, <>, <, <=, >, >= +SL 303 (=) 308 (<>) 313 (<) 318 (<=) 323 (>) 328 (>=)	**S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (double-word, 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
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
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
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

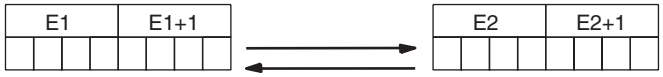
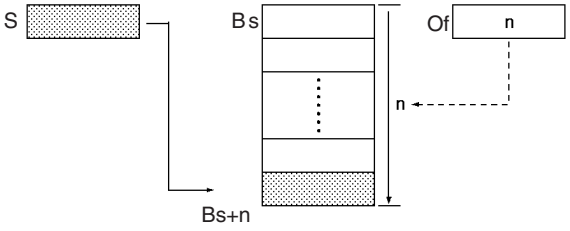
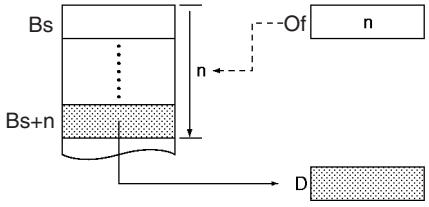
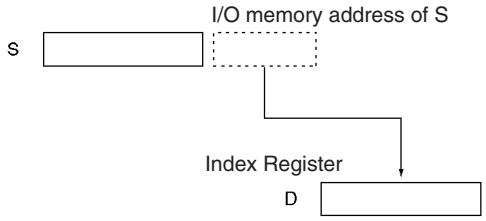
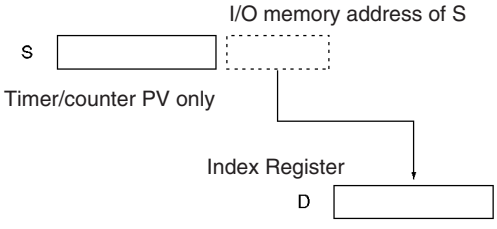
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SIGNED BINARY COMPARE CPS !CPS*1 114	CPS(114) S_1 S_2 **S1**: Comparison data 1 **S2**: Comparison data 2	Compares two signed binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area. 	Output Required
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
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
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
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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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)	 **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
AREA RANGE COMPARE ZCP @ZCP 088 (CS1-H, CJ1-H, CJ1M, or CS1D only)	 **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
DOUBLE AREA RANGE COMPARE ZCPL @ZCPL 116 (CS1-H, CJ1-H, CJ1M, or CS1D only)	 **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

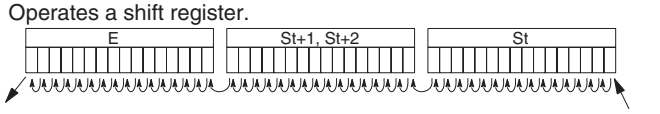
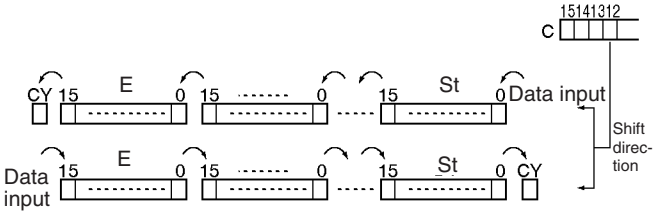
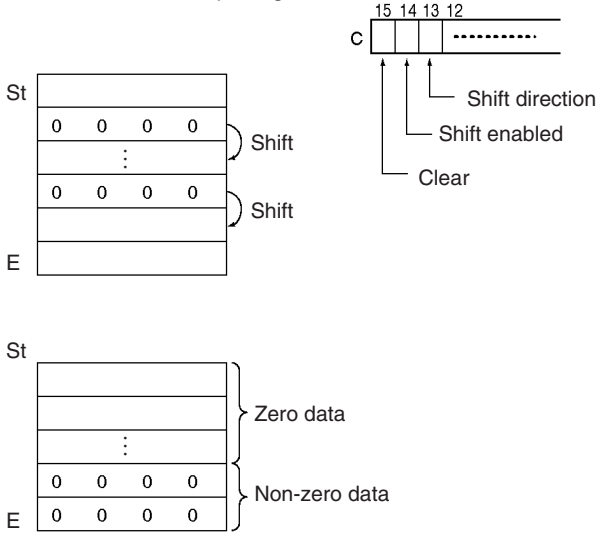
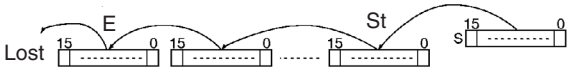
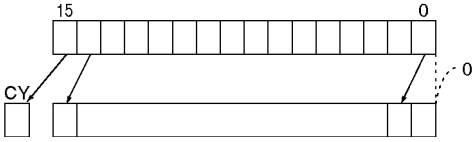
3-6 Data Movement Instructions

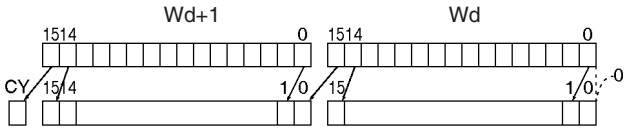
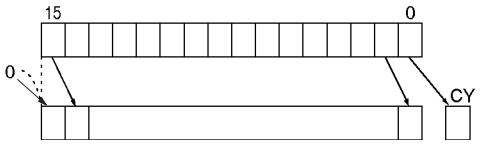
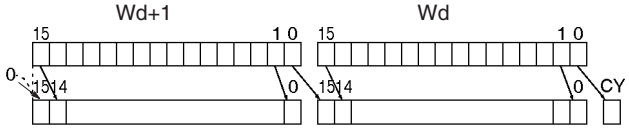
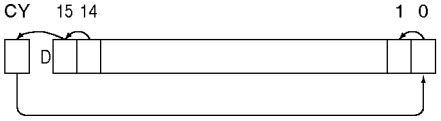
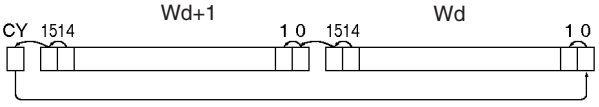
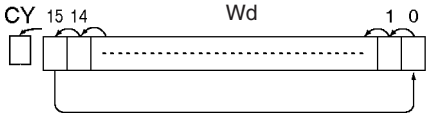
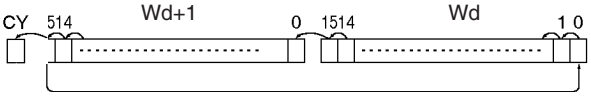
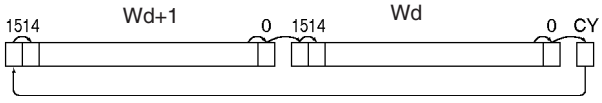
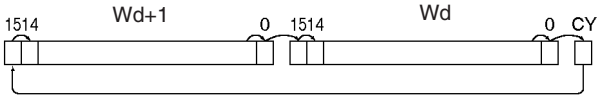
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
MOVE MOV @MOV !MOV !@MOV 021	MOV(021) S D **S**: Source **D**: Destination	Transfers a word of data to the specified word. 	Output Required
DOUBLE MOVE MOVL @MOVL 498	MOVL(498) S D **S**: 1st source word **D**: 1st destination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

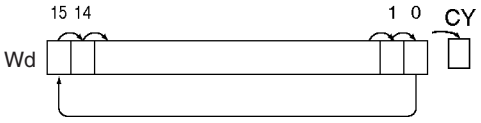
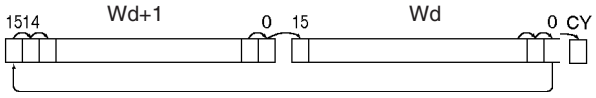
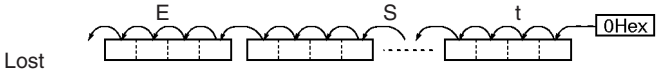
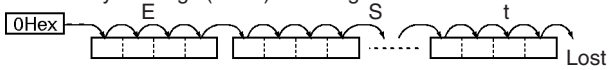
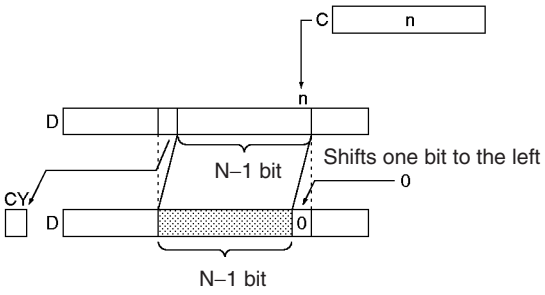
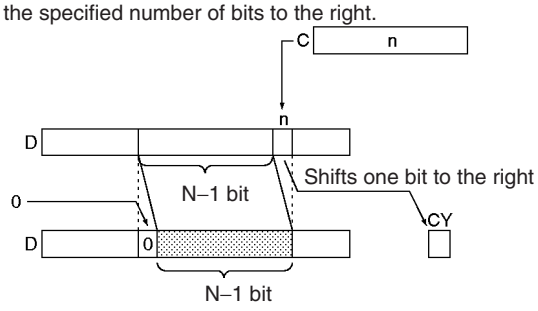
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
MOVE DIGIT MOVD @MOVD 083	MOVD(083) S C D **S**: Source word or data **C**: Control word **D**: Destination word	Transfers the specified digit or digits. (Each digit is made up of 4 bits.) 	Output Required
MULTIPLE BIT TRANSFER 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
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
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
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

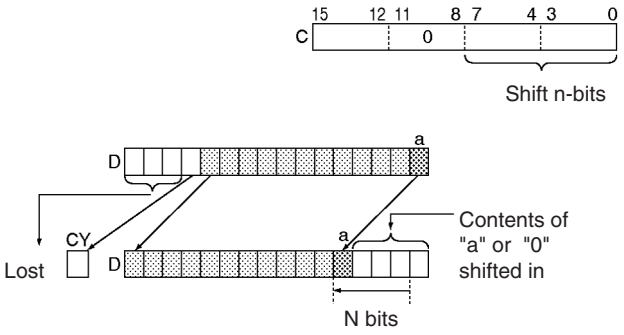
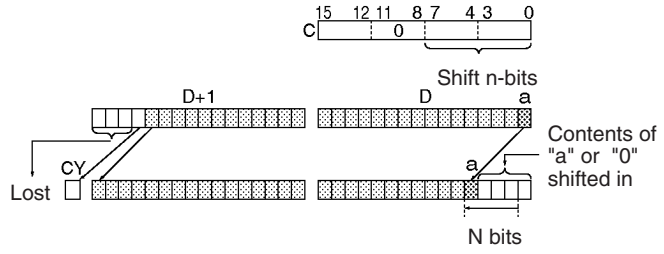
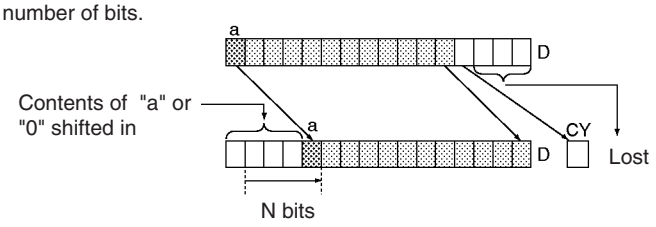
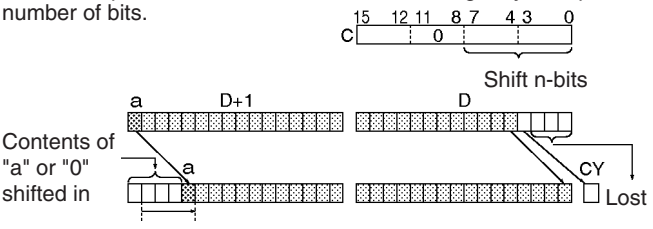
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DOUBLE DATA EXCHANGE XCGL @XCGL 562	XCGL(562) E1 E2 **E1**: 1st exchange word **E2**: Second exchange word	Exchanges the contents of a pair of consecutive words with another pair of consecutive words. 	Output Required
SINGLE 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
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
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
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

3-7 Data Shift Instructions

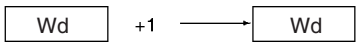
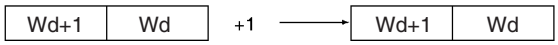
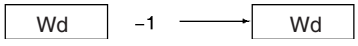
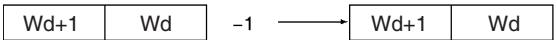
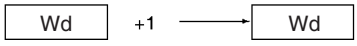
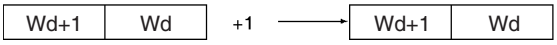
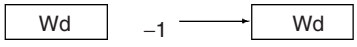
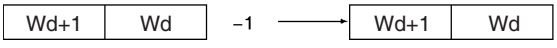
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SHIFT REGISTER SFT 010	Data input Shift input Reset input SFT(010) St E **St**: Starting word **E**: End word	Operates a shift register. 	Output Required
REVERSIBLE SHIFT REGISTER SFTR @SFTR 084	SFTR(084) C St E **C**: Control word **St**: Starting word **E**: End word	Creates a shift register that shifts data to either the right or the left. 	Output Required
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
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
ARITHMETIC SHIFT LEFT ASL @ASL 025	ASL(025) Wd **Wd**: Word	Shifts the contents of Wd one bit to the left. 	Output Required

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
ARITHMETIC SHIFT RIGHT ASR @ASR 026	ASR(026) Wd **Wd**: Word	Shifts the contents of Wd one bit to the right. 	Output Required
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
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
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
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
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
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
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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
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
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
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
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

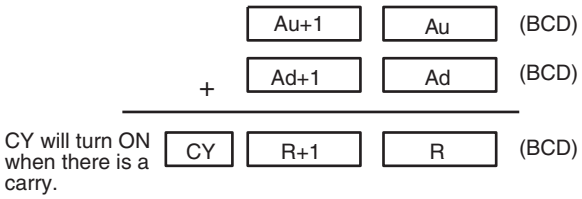
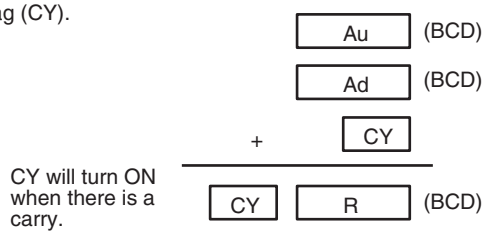
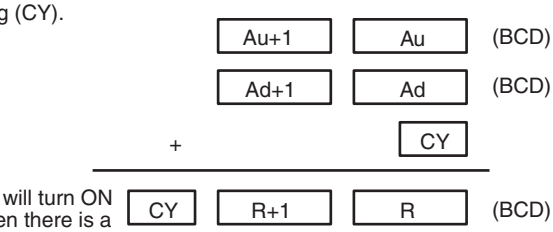
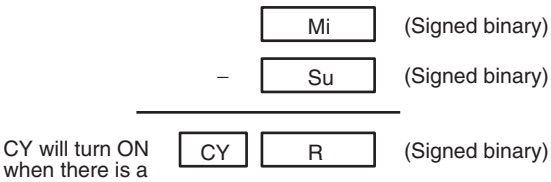
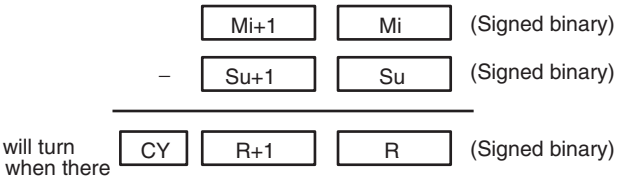
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SHIFT N-BITS LEFT NASL @NASL 580	NASL(580) D C **D**: Shift word **C**: Control word	Shifts the specified 16 bits of word data to the left by the specified number of bits. 	Output Required
DOUBLE SHIFT N-BITS LEFT NSLL @NSLL 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
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
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

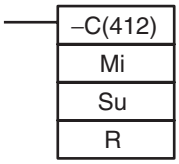
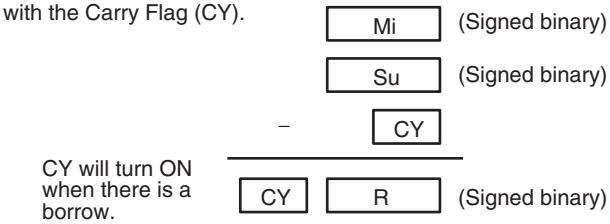
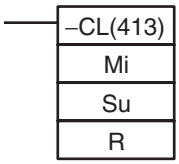
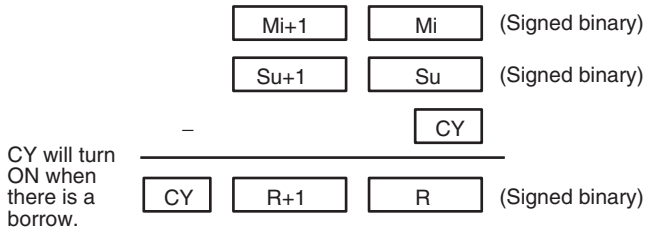
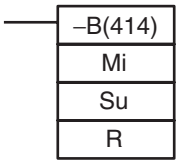
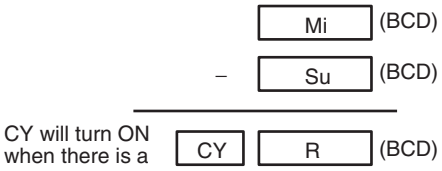
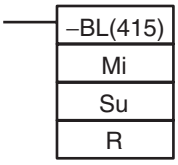
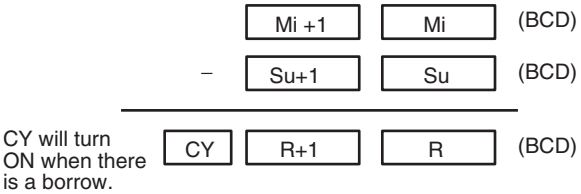
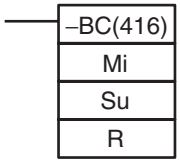
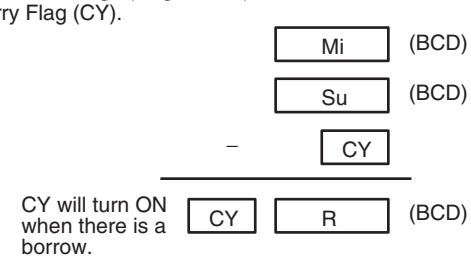
3-8 Increment/Decrement Instructions

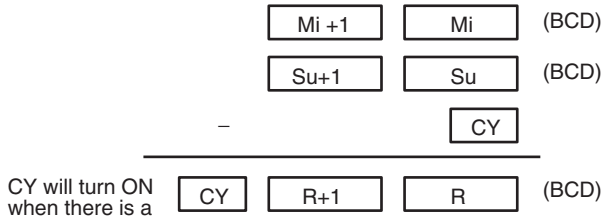
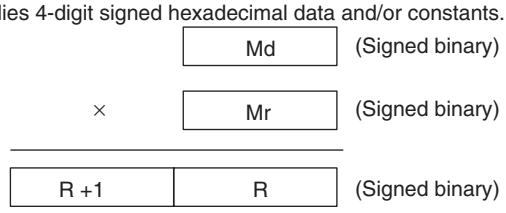
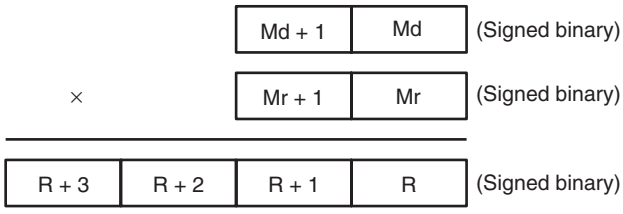
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
INCREMENT BINARY ++ @++ 590	++(590) Wd **Wd**: Word	Increments the 4-digit hexadecimal content of the specified word by 1. 	Output Required
DOUBLE INCREMENT BINARY ++L @++L 591	++L(591) Wd **Wd**: Word	Increments the 8-digit hexadecimal content of the specified words by 1. 	Output Required
DECREMENT BINARY – – @– – 592	– – (592) Wd **Wd**: Word	Decrements the 4-digit hexadecimal content of the specified word by 1. 	Output Required
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
INCREMENT BCD ++B @++B 594	++B(594) Wd **Wd**: Word	Increments the 4-digit BCD content of the specified word by 1. 	Output Required
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
DECREMENT BCD – –B @– –B 596	– – B(596) Wd **Wd**: Word	Decrements the 4-digit BCD content of the specified word by 1. 	Output Required
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

3-9 Symbol Math Instructions

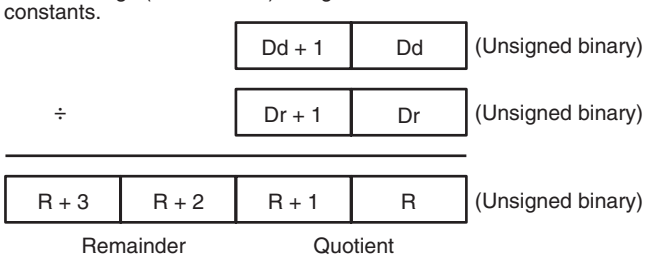
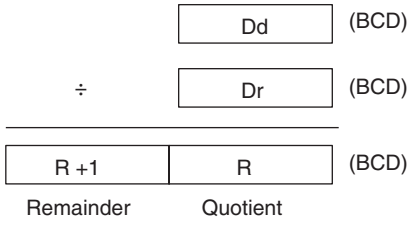
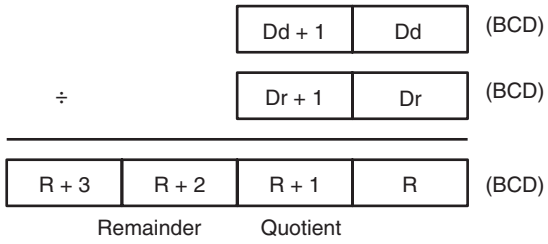
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SIGNED BINARY ADD WITHOUT CARRY + @+ 400	+(400) Au Ad R **Au**: Augend word **Ad**: Addend word **R**: Result word	Adds 4-digit (single-word) hexadecimal data and/or constants. Au (Signed binary) + Ad (Signed binary) CY will turn ON when there is a carry. CY R (Signed binary)	Output Required
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
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
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
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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
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
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
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

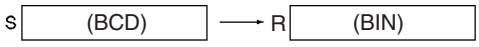
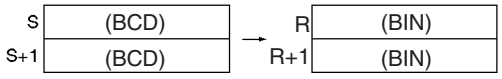
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SIGNED BINARY SUBTRACT WITH CARRY −C @−C 412	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY). 	Output Required
DOUBLE 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). 	Output Required
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. 	Output Required
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. 	Output Required
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). 	Output Required

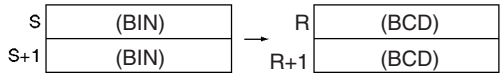
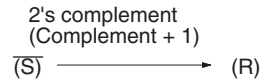
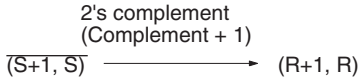
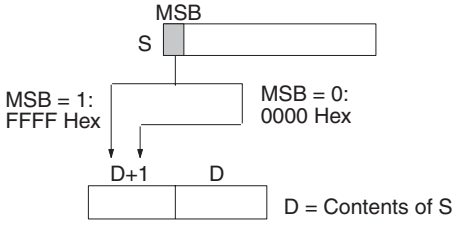
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DOUBLE BCD SUBTRACT WITH CARRY −BCL @−BCL 417	−BCL(417) Mi Su R **Mi**: 1st minuend word **Su**: 1st subtrahend word **R**: 1st result word	Subtracts 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY). 	Output Required
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
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
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
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

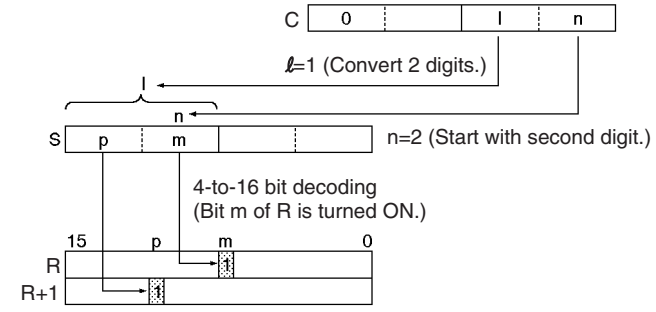
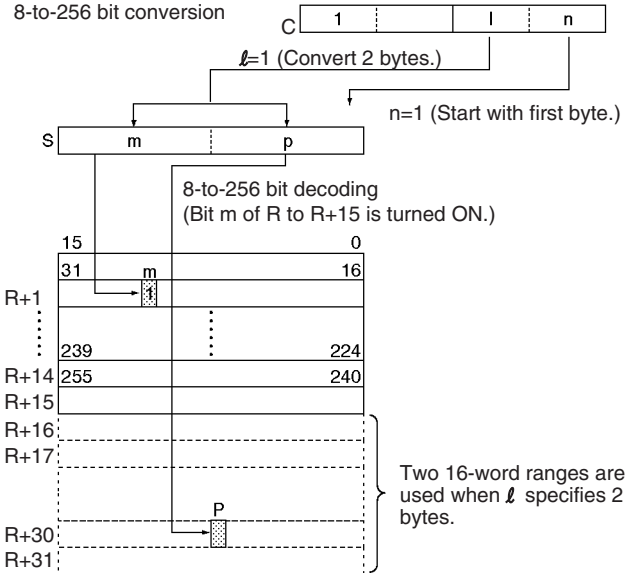
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
BCD MULTIPLY *B @*B 424	*B(424) Md Mr R **Md**: Multiplicand word **Mr**: Multiplier word **R**: Result word	Multiplies 4-digit (single-word) BCD data and/or constants. Md (BCD) × Mr (BCD) R +1 | R (BCD)	Output Required
DOUBLE 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. Md + 1 | Md (BCD) × Mr + 1 | Mr (BCD) R + 3 | R + 2 | R + 1 | R (BCD)	Output Required
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. Dd (Signed binary) ÷ Dr (Signed binary) R +1 | R (Signed binary) Remainder Quotient	Output Required
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. Dd + 1 | Dd (Signed binary) ÷ Dr + 1 | Dr (Signed binary) R + 3 | R + 2 | R + 1 | R (Signed binary) Remainder Quotient	Output Required
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. Dd (Unsigned binary) ÷ Dr (Unsigned binary) R +1 | R (Unsigned binary) Remainder Quotient	Output Required

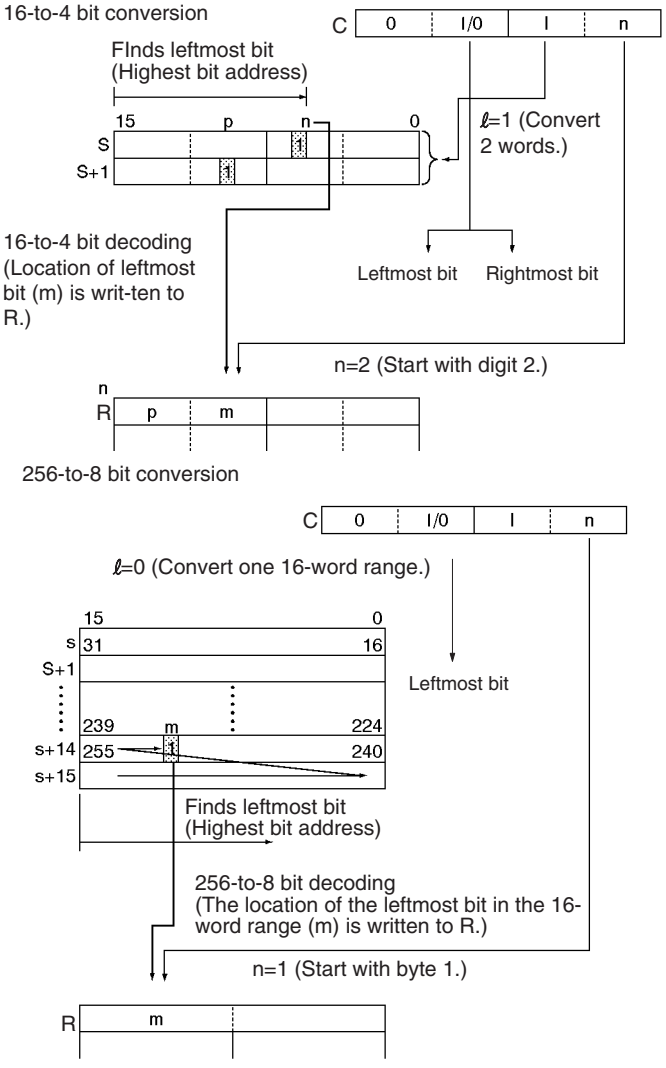
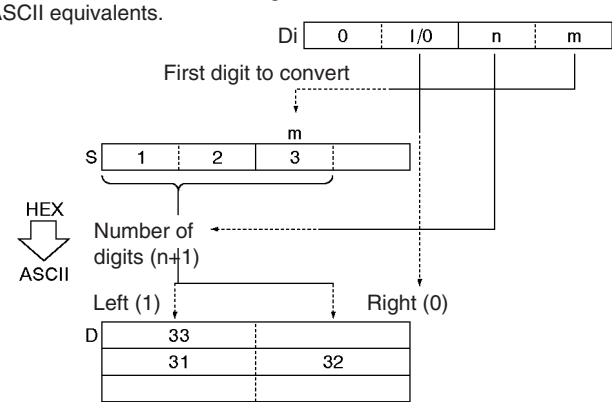
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DOUBLE UNSIGNED BINARY DIVIDE /UL @/UL 433	/UL(433) Dd Dr R **Dd**: 1st dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides 8-digit (double-word) unsigned hexadecimal data and/or constants. 	Output Required
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
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

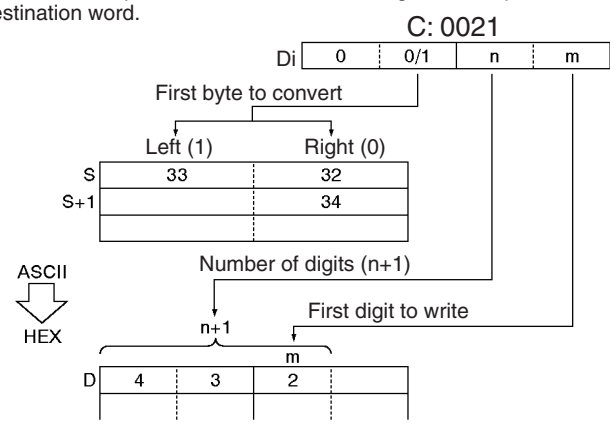
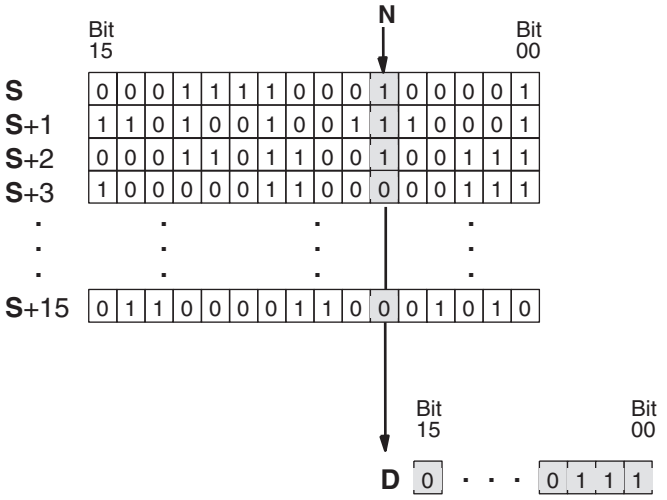
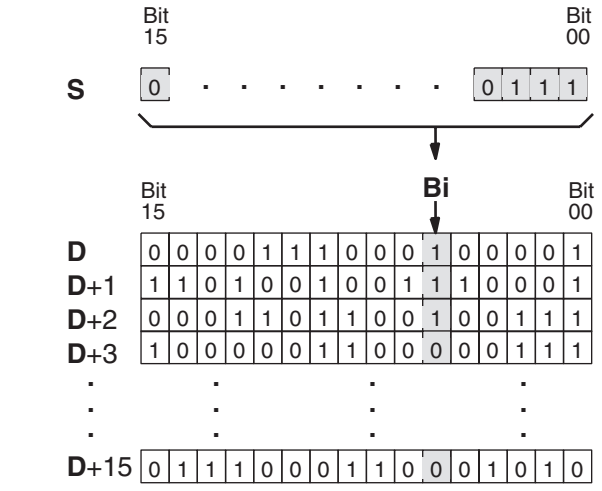
3-10 Conversion Instructions

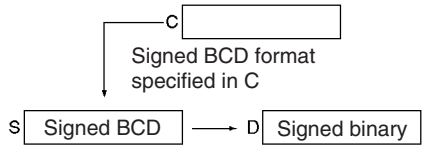
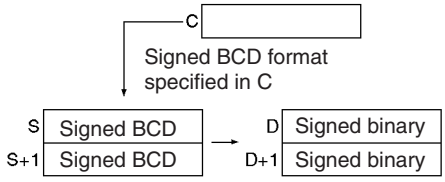
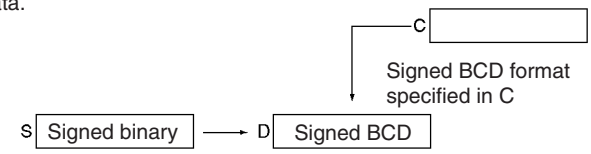
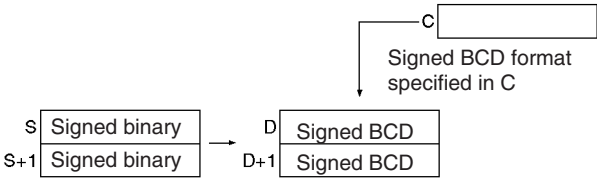
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
BCD-TO-BINARY BIN @BIN 023	BIN(023) S R **S**: Source word **R**: Result word	Converts BCD data to binary data. 	Output Required
DOUBLE BCD-TO-DOUBLE BINARY BINL @BINL 058	BINL(058) S R **S**: 1st source word **R**: 1st result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

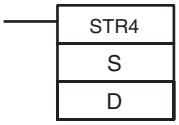
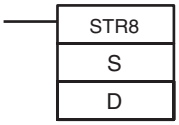
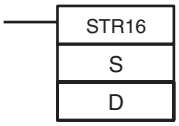
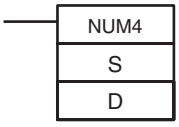
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
BINARY-TO-BCD BCD @BCD 024	BCD(024) S R **S**: Source word **R**: Result word	Converts a word of binary data to a word of BCD data. 	Output Required
DOUBLE BINARY-TO-DOUBLE BCD BCDL @BCDL 059	BCDL(059) S R **S**: 1st source word **R**: 1st result word	Converts 8-digit hexadecimal (32-bit binary) data to 8-digit BCD data. 	Output Required
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
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
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

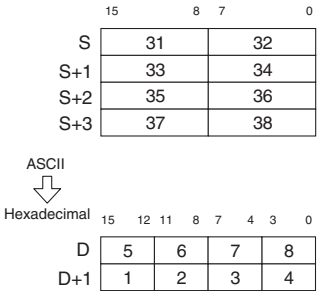
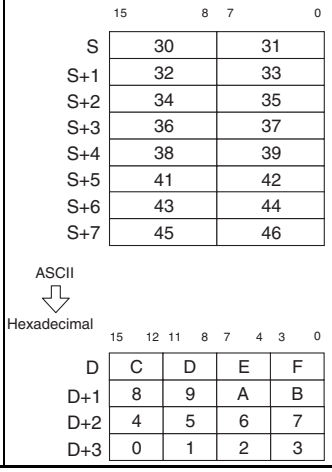
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DATA DECODER MLPX @MLPX 076	MLPX(076) S C R **S**: Source word **C**: Control word **R**: 1st result word	Reads the numerical value in the specified digit (or byte) in the source word, turns ON the corresponding bit in the 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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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 	Output Required
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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
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

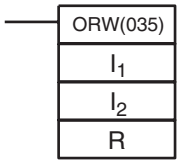
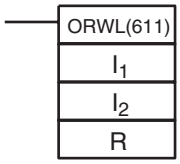
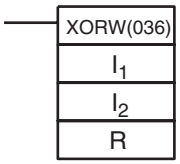
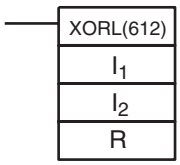
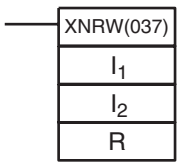
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SIGNED BCD-TO-BINARY BINS @BINS 470	BINS(470) C S D **C**: Control word **S**: Source word **D**: Destination word	Converts one word of signed BCD data to one word of signed binary data. 	Output Required
DOUBLE SIGNED BCD-TO-BINARY 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
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
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
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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
FOUR-DIGIT NUMBER TO ASCII STR4 @STR4 601	 **S**: Numeric **D**: ASCII text	Converts a 4-digit hexadecimal number (#0000 to #FFFF) to ASCII data (4 characters). 15 12 11 8 7 4 3 0 S: 1 2 3 4 Hexadecimal: #1234 ⇩ ASCII 15 8 7 0 D: 31 32 D+1: 33 34	Output Required
EIGHT-DIGIT NUMBER TO ASCII STR8 @STR8 602	 **S**: Numeric **D**: ASCII text	Converts an 8-digit hexadecimal number (#0000 0000 to #FFFF FFFF) to ASCII data (8 characters). 15 12 11 8 7 4 3 0 S: 5 6 7 8 S+1: 1 2 3 4 Hexadecimal: #12345678 ⇩ ASCII 15 8 7 0 D: 31 32 D+1: 33 34 D+2: 35 36 D+3: 37 38	Output Required
SIXTEEN-DIGIT NUMBER TO ASCII STR16 @STR16 603	 **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). 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	Output Required
ASCII TO FOUR-DIGIT NUMBER NUM4 @NUM4 604	 **S**: ASCII text **D**: Numeric	Converts 4 characters of ASCII data to a 4-digit hexadecimal number. 15 8 7 0 S: 31 32 S+1: 33 34 ASCII ⇩ Hexadecimal 15 12 11 8 7 4 3 0 D: 1 2 3 4	Output Required

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
ASCII TO EIGHT-DIGIT NUMBER NUM8 @NUM8 605	NUM8 S D **S**: ASCII text **D**: Numeric	Converts 8 characters of ASCII data to an 8-digit hexadecimal number. 	Output Required
ASCII TO SIX-TEEN-DIGIT-NUMBER NUM16 @NUM16 606	NUM16 S D **S**: ASCII text **D**: Numeric	Converts 16 characters of ASCII data to a 16-digit hexadecimal number. 	Output Required

3-11 Logic Instructions

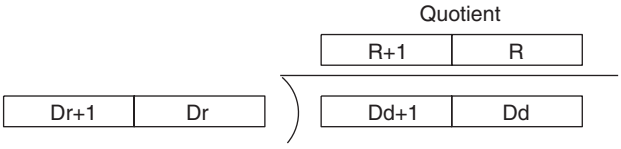
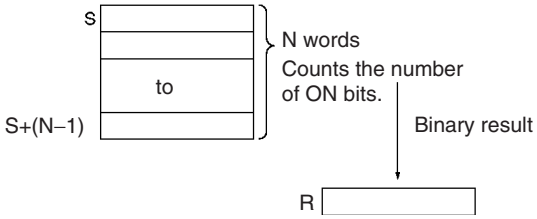
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
LOGICAL AND ANDW @ANDW 034	ANDW(034) I_1 I_2 R **I1**: Input 1 **I2**: Input 2 **R**: Result word	Takes the logical AND of corresponding bits in single words of word data and/or constants. **I_1. I_2 →R** **I_1** **I_2** **R** 1 1 1 1 0 0 0 1 0 0 0 0	Output Required
DOUBLE LOGICAL AND ANDL @ANDL 610	ANDL(610) I_1 I_2 R **I1**: Input 1 **I2**: Input 2 **R**: Result word	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) → (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	Output Required

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
LOGICAL OR ORW @ORW 035	 **I_1**: Input 1 **I_2**: Input 2 **R**: Result word	Takes the logical OR of corresponding bits in single words of word data and/or constants. **$I_1 + I_2 \rightarrow R$** I_1 I_2 R: 1 1 1; 1 0 1; 0 1 1; 0 0 0	Output Required
DOUBLE LOGICAL OR ORWL @ORWL 611	 **I_1**: Input 1 **I_2**: Input 2 **R**: Result word	Takes the logical OR of corresponding bits in double words of word data and/or constants. **$(I_1,I_1+1) + (I_2,I_2+1) \rightarrow (R, R+1)$** 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	Output Required
EXCLUSIVE OR XORW @XORW 036	 **I_1**: Input 1 **I_2**: Input 2 **R**: Result word	Takes the logical exclusive OR of corresponding bits in single words of word data and/or constants. **$I_1.I_2 + I_1.I_2 \rightarrow R$** I_1 I_2 R: 1 1 0; 1 0 1; 0 1 1; 0 0 0	Output Required
DOUBLE EXCLUSIVE OR XORL @XORL 612	 **I_1**: Input 1 **I_2**: Input 2 **R**: Result word	Takes the logical exclusive OR of corresponding bits in double words of word data and/or constants. **$(I_1,I_1+1).\overline{(I_2,I_2+1)} + \overline{(I_1,I_1+1)}.(I_2,I_2+1) \rightarrow (R,R+1)$** 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	Output Required
EXCLUSIVE NOR XNRW @XNRW 037	 **I_1**: Input 1 **I_2**: Input 2 **R**: Result word	Takes the logical exclusive NOR of corresponding single words of word data and/or constants. **$I_1.I_2 + I_1.I_2 \rightarrow R$** I_1 I_2 R: 1 1 1; 1 0 0; 0 1 0; 0 0 1	Output Required

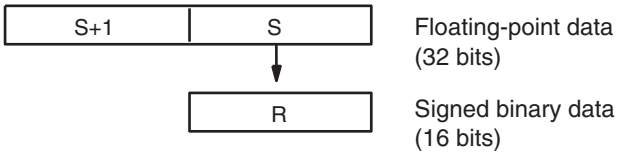
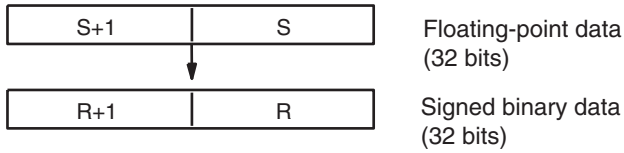
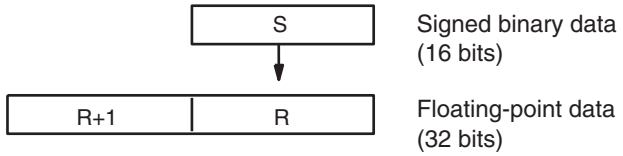
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DOUBLE EXCLUSIVE NOR XNRL @XNRL 613	XNRL(613) I_1 I_2 R **I_1**: Input 1 **I_2**: Input 2 **R**: 1st result word	Takes the logical exclusive NOR of corresponding bits in double words of word data and/or constants. **$(I_1.I_1+1). (I_2.I_2+1) + (\overline{I_1.I_1+1}). (\overline{I_2.I_2+1}) \rightarrow$ (R, R+1)** $I_1.I_1+1$ | $I_2.I_2+1$ | R, R+1 1 1 1 1 0 0 0 1 0 0 0 1	Output Required
COMPLEMENT COM @COM 029	COM(029) Wd **Wd**: Word	Turns OFF all ON bits and turns ON all OFF bits in Wd. $\overline{Wd} \rightarrow$ Wd: 1 → 0 and 0 → 1	Output Required
DOUBLE COMPLEMENT COML @COML 614	COML(614) Wd **Wd**: Word	Turns OFF all ON bits and turns ON all OFF bits in Wd and Wd+1. $\overline{(Wd+1, Wd)} \rightarrow$ (Wd+1, Wd)	Output Required

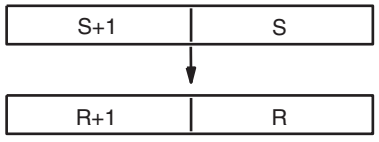
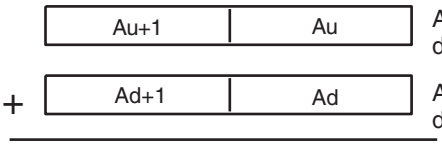
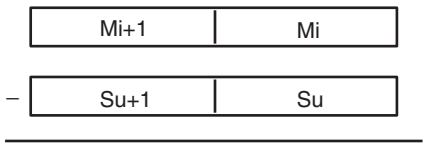
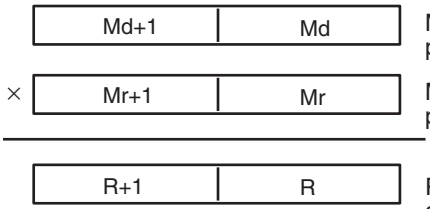
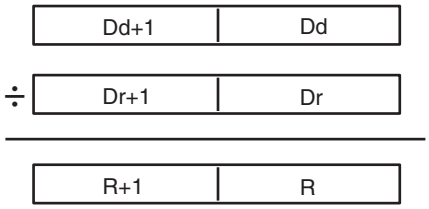
3-12 Special Math Instructions

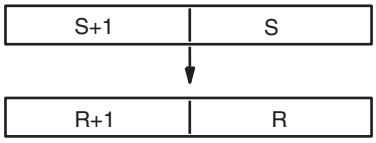
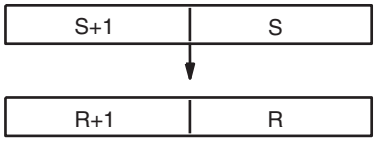
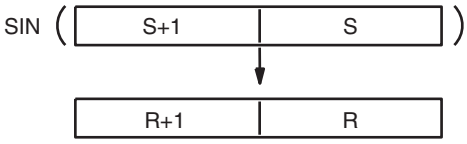
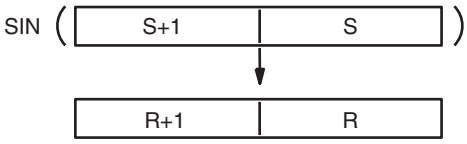
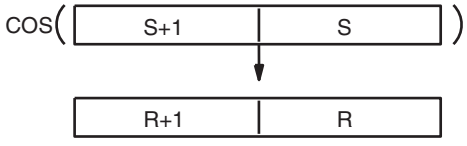
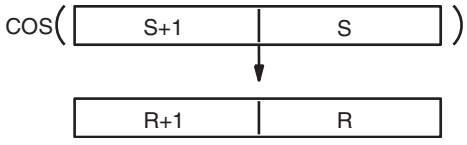
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
BINARY ROOT ROTB @ROTB 620	ROTB(620) S R **S**: 1st source word **R**: Result word	Computes the square root of the 32-bit binary content of the specified words and outputs the integer portion of the result to the specified result word. S+1 | S → R Binary data (32 bits) → Binary data (16 bits)	Output Required
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. S+1 | S → R BCD data (8 digits) → BCD data (4 digits)	Output Required
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

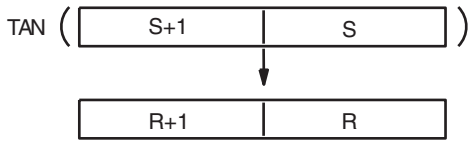
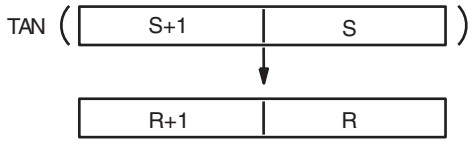
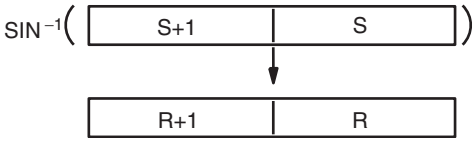
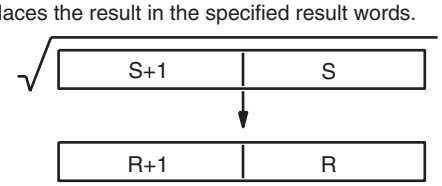
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
FLOATING POINT DIVIDE FDIV @FDIV 079	FDIV(079) Dd Dr R **Dd**: 1st dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides one 7-digit floating-point number by another. The floating-point numbers are expressed in scientific notation (7-digit mantissa and 1-digit exponent). 	Output Required
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

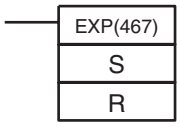
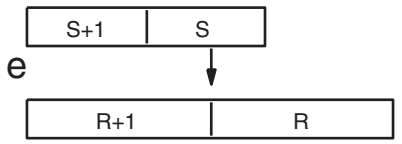
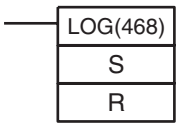
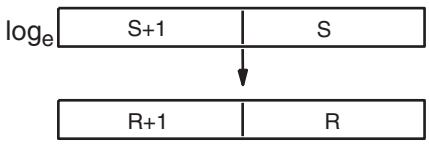
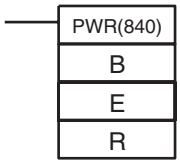
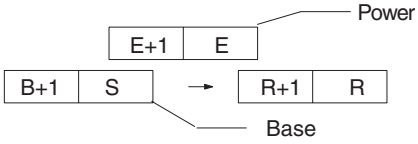
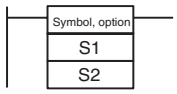
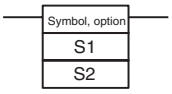
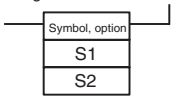
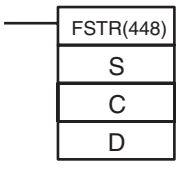
3-13 Floating-point Math Instructions

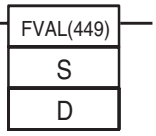
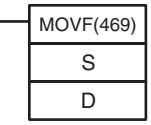
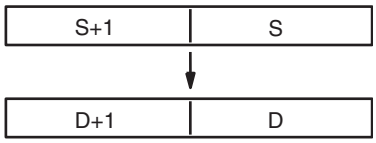
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
FLOATING TO 16-BIT FIX @FIX 450	FIX(450) S R **S**: 1st source word **R**: Result word	Converts a 32-bit floating-point value to 16-bit signed binary data and places the result in the specified result word. 	Output Required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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
FLOATING-POINT SUBTRACT −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
FLOATING-POINT MULTIPLY *F @*F 456	*F(456) Md Mr R **Md**: 1st Multiplicand word **Mr**: 1st Multiplier word **R**: 1st result word	Multiplies two 32-bit floating-point numbers and places the result in the specified result words.  Multiplicand (floating-point data, 32 bits) Multiplier (floating-point data, 32 bits) Result (floating-point data, 32 bits)	Output Required
FLOATING-POINT DIVIDE /F @/F 457	/F(457) Dd Dr R **Dd**: 1st Dividend word **Dr**: 1st Divisor word **R**: 1st result word	Divides one 32-bit floating-point number by another and places the result in the specified result words.  Dividend (floating-point data, 32 bits) Divisor (floating-point data, 32 bits) Result (floating-point data, 32 bits)	Output Required

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DEGREES TO RADIANS RAD @RAD 458	RAD(458) S R **S**: 1st source word **R**: 1st result word	Converts a 32-bit floating-point number from degrees to radians and places the result in the specified result words.  Source (degrees, 32-bit floating-point data) Result (radians, 32-bit floating-point data)	Output Required
RADIANS TO DEGREES DEG @DEG 459	DEG(459) S R **S**: 1st source word **R**: 1st result word	Converts a 32-bit floating-point number from radians to degrees and places the result in the specified result words.  Source (radians, 32-bit floating-point data) Result (degrees, 32-bit floating-point data)	Output Required
SINE SIN @SIN 460	SIN(460) S R **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
HIGH-SPEED SINE (CJ1-H-R CPU only) SINQ @SINQ 475	SINQ (475) S R **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
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. 	Output Required
HIGH-SPEED COSINE (CJ1-H-R CPU only) COS @COS 476	COS(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. 	Output Required

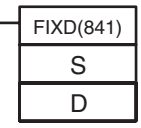
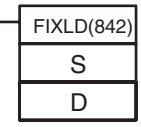
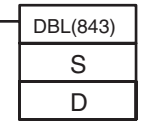
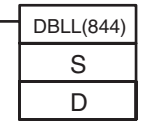
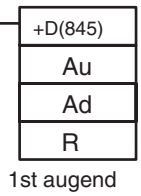
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
TANGENT TAN @TAN 462	TAN(462) S R **S**: 1st source word **R**: 1st result word	Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words. 	Output Required
HIGH-SPEED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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
EXPONENT EXP @EXP 467	 **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.  Source (32-bit floating-point data) Result (32-bit floating-point data)	Output Required
LOGARITHM LOG @LOG 468	 **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.  Source (32-bit floating-point data) Result (32-bit floating-point data)	Output Required
EXPONENTIAL POWER PWR @PWR 840	 **B**: 1st base word **E**: 1st exponent word **R**: 1st result word	Raises a 32-bit floating-point number to the power of another 32-bit floating-point number. 	Output Required
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
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

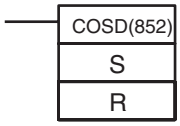
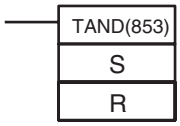
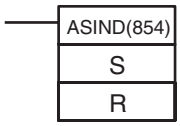
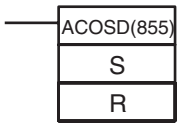
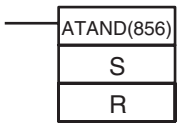
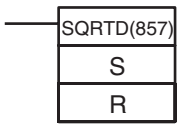
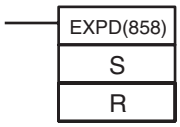
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
ASCII TO FLOATING-POINT (CS1-H, CJ1-H, CJ1M, or CS1D only) FVAL @FVAL 449	 **S:** Source word **D**: 1st destination word	Converts the specified text string (ASCII) representation of single-precision floating-point data (decimal-point or exponential format) to 32-bit single-precision floating-point data and outputs the result to the destination words.	Output required
MOVE FLOATING-POINT (SINGLE) (CJ1-H-R only) MOVF 469	 **S**: 1st source word **D**: 1st destination word	Transfers the specified 32-bit floating-point number to the specified destination words. 	Output required

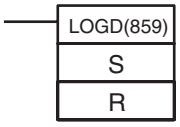
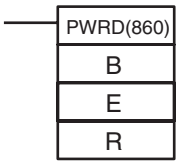
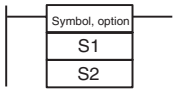
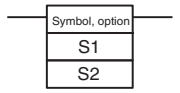
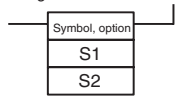
3-14 Double-precision Floating-point Instructions

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

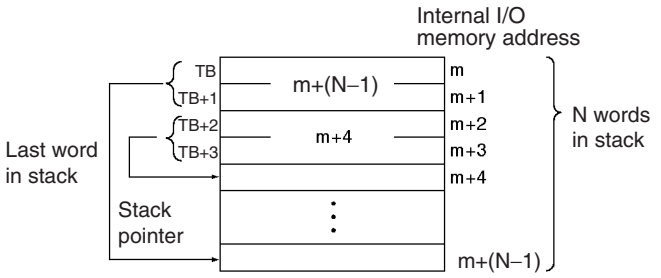
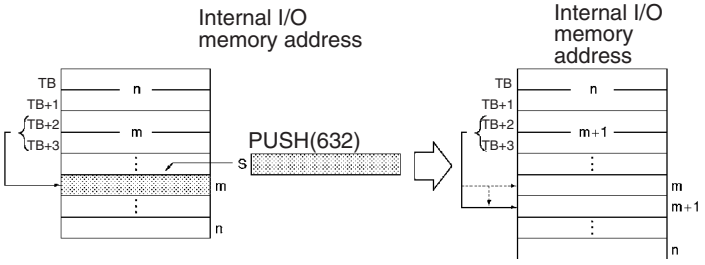
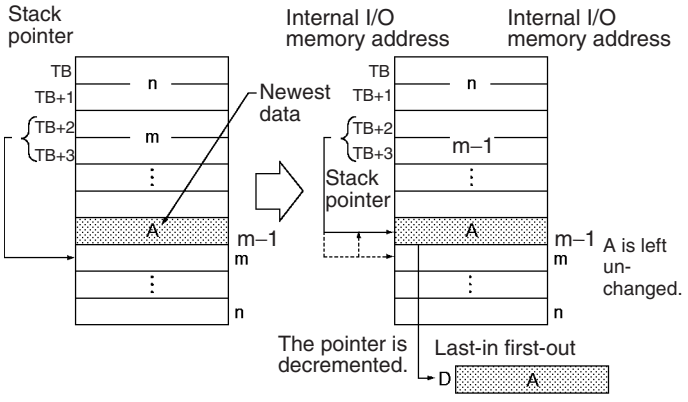
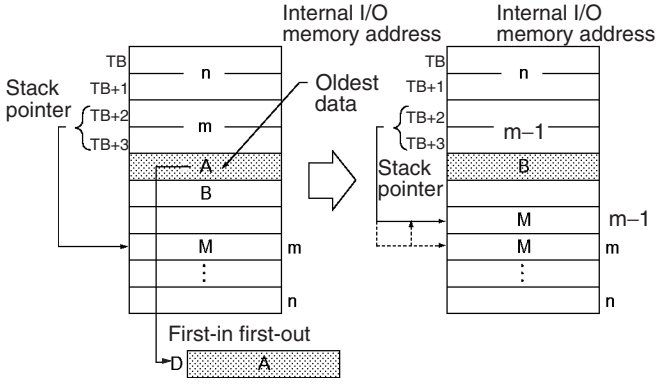
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DOUBLE FLOATING TO 16-BIT BINARY FIXD @FIXD 841	 **S:** 1st source word **D**: Destination word	Converts the specified double-precision floating-point data (64 bits) to 16-bit signed binary data and outputs the result to the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

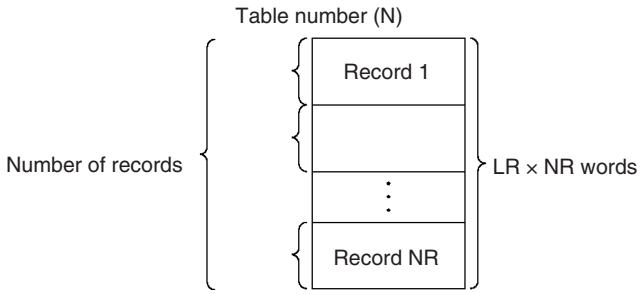
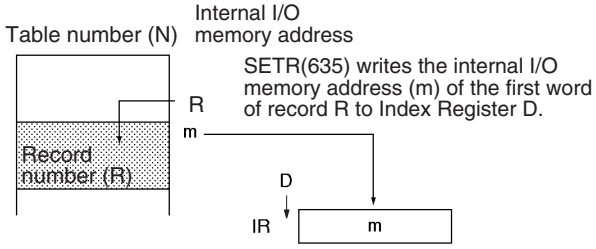
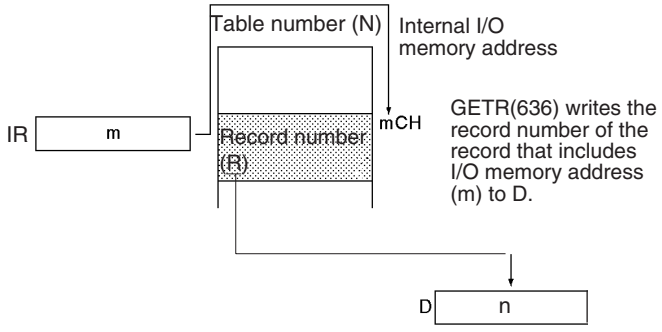
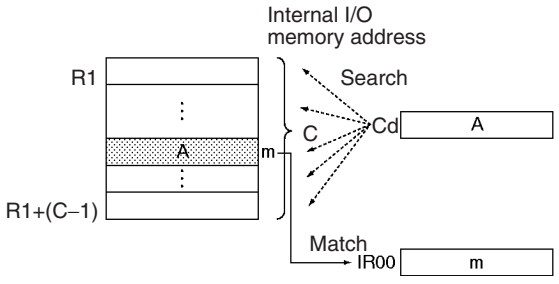
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DOUBLE FLOATING-POINT SUBTRACT −D @−D 846	−D(846) Mi Su R **Mi:** 1st minuend word **Su**: 1st subtrahend word **R**: 1st result word	Subtracts the specified double-precision floating-point values (64 bits each) and outputs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

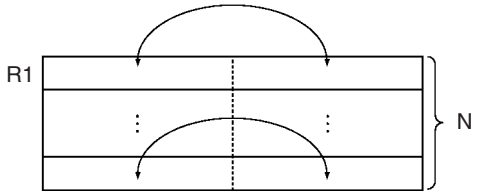
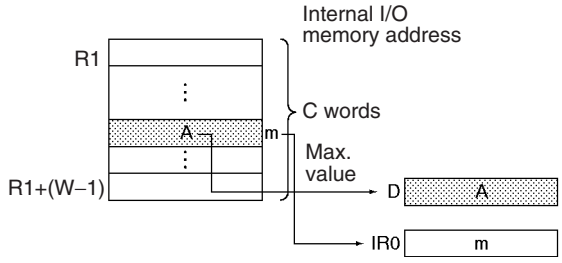
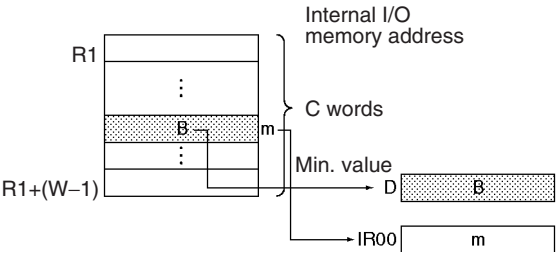
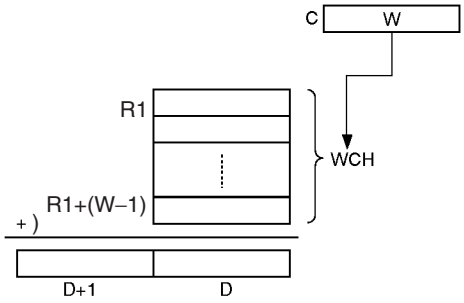
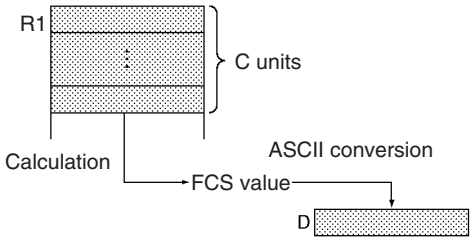
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DOUBLE COSINE COSD @COSD 852	 **S:** 1st source word **R**: 1st result word	Calculates the cosine of the angle (radians) in the specified double-precision floating-point data (64 bits) and outputs the result to the result words.	Output Required
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
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
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
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
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
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

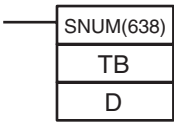
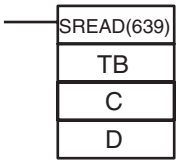
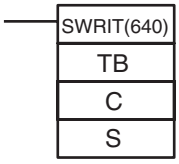
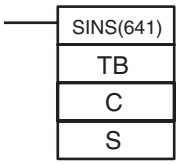
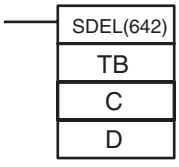
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DOUBLE LOGARITHM LOGD @LOGD 859	 **S:** 1st source word **R**: 1st result word	Calculates the natural (base e) logarithm of the specified double-precision floating-point data (64 bits) and outputs the result to the result words.	Output Required
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
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

3-15 Table Data Processing Instructions

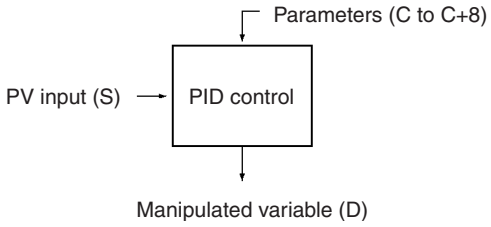
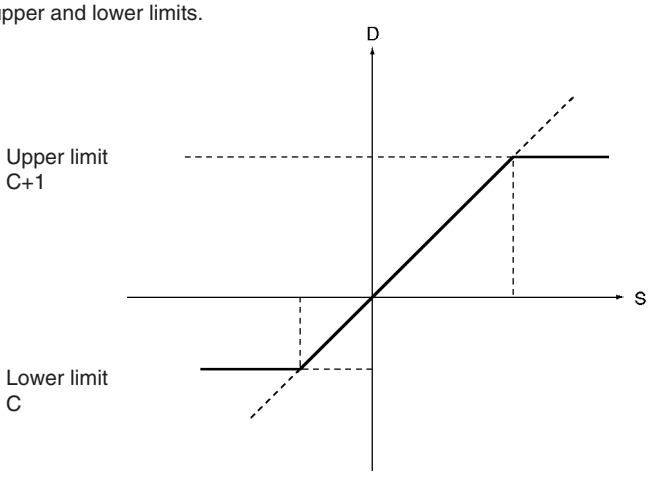
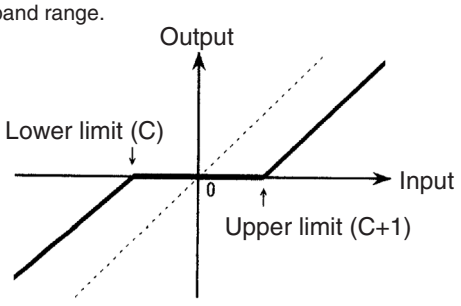
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SET STACK SSET @SSET 630	SSET(630) TB N **TB**: 1st stack address **N**: Number of words	Defines a stack of the specified length beginning at the specified word and initializes the words in the data region to all zeroes. 	Output Required
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
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
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

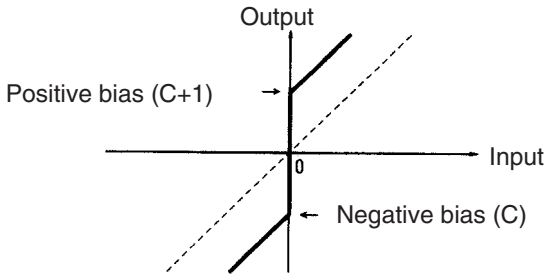
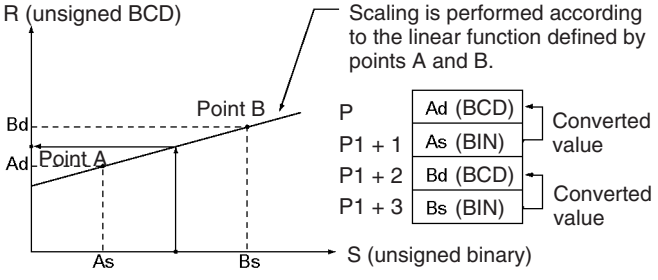
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DIMENSION RECORD TABLE DIM @DIM 631	DIM(631) N LR NR TB **N**: Table number **LR**: Length of each record **NR**: Number of records **TB**: 1st table word	Defines a record table by declaring the length of each record and the number of records. Up to 16 record tables can be defined. 	Output Required
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
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
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

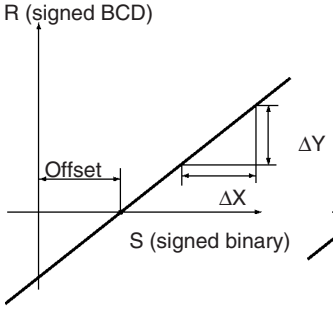
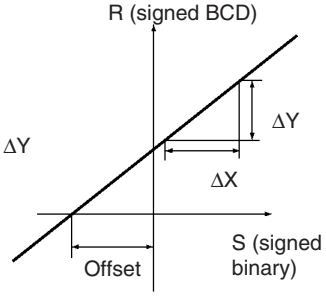
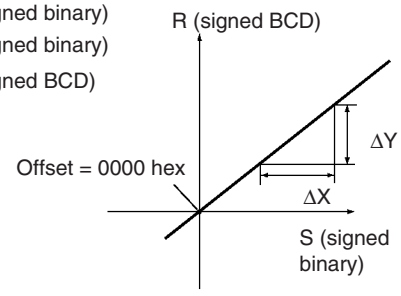
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SWAP BYTES SWAP @SWAP 637	SWAP(637) N R1 **N**: Number of words **R1**: 1st word in range	Switches the leftmost and rightmost bytes in all of the words in the range. Byte position is swapped. 	Output Required
FIND MAXIMUM 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
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
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
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

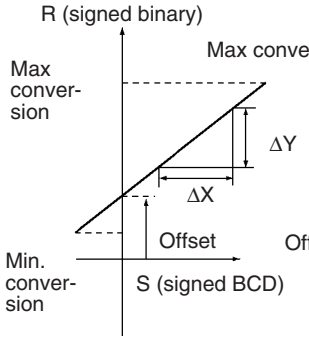
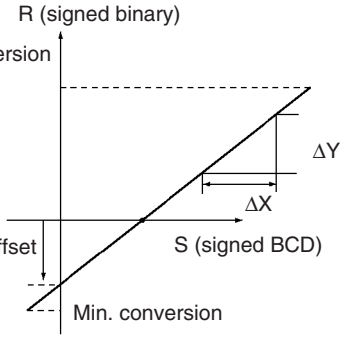
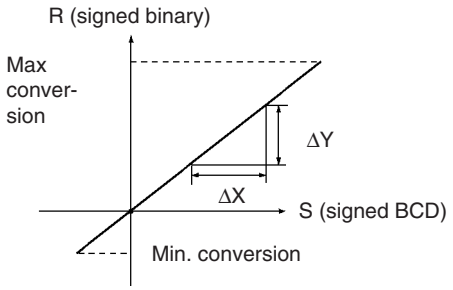
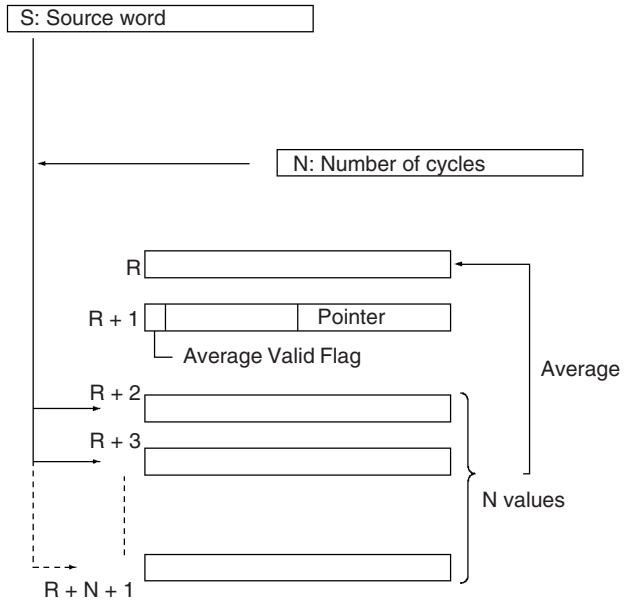
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
STACK SIZE READ (CS1-H, CJ1-H, CJ1M, or CS1D only) SNUM @SNUM 638	 **TB**: First stack address **D**: Destination word	Counts the amount of stack data (number of words) in the specified stack.	Output required
STACK DATA READ (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
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
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
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

3-16 Data Control Instructions

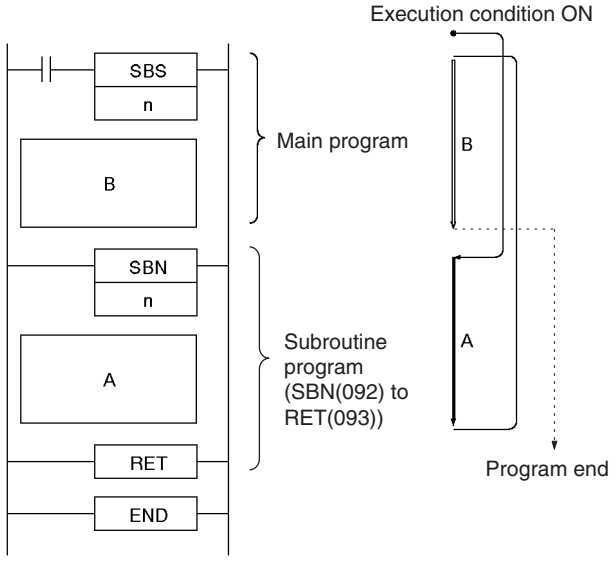
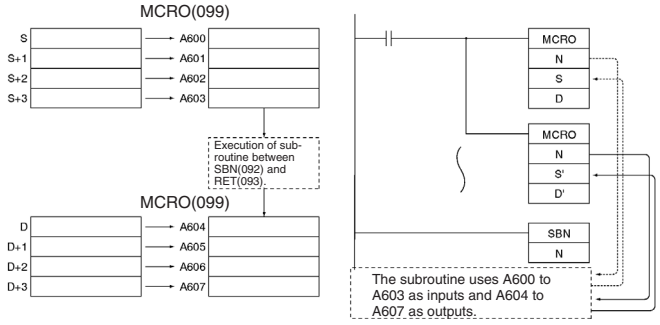
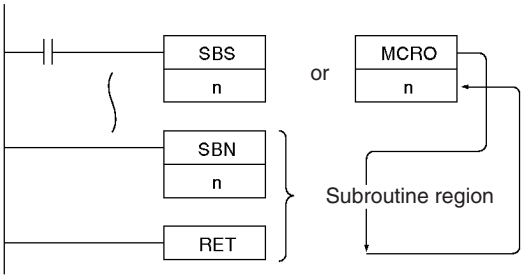
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
PID CONTROL PID 190	PID(190) S C D **S**: Input word **C**: 1st parameter word **D**: Output word	Executes PID control according to the specified parameters. 	Output Required
PID CONTROL WITH AUTOTUNING PIDAT 191 (CS1-H, CJ1-H, 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
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
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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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

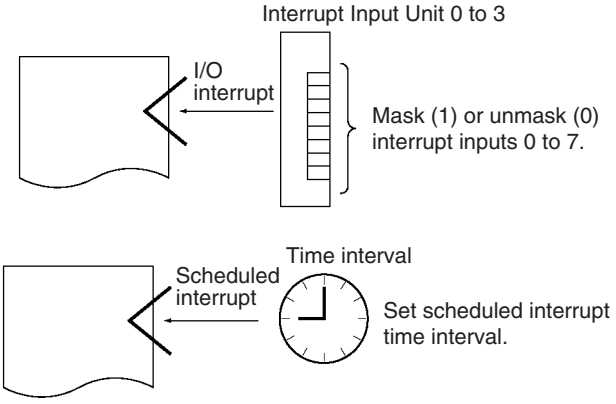
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SCALING 3 SCL3 @SCL3 487	SCL3(487) S P1 R **S**: Source word **P1**: 1st parameter word **R**: Result word	Converts signed BCD data into signed binary data according to the specified linear function. An offset can be input in defining the linear function. **Positive Offset**  **Negative Offset**  **Offset of 0000** 	Output Required
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

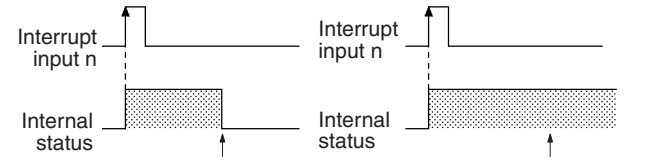
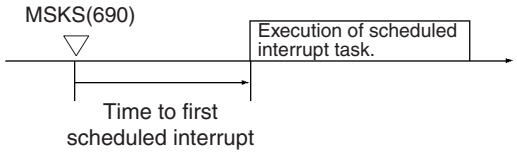
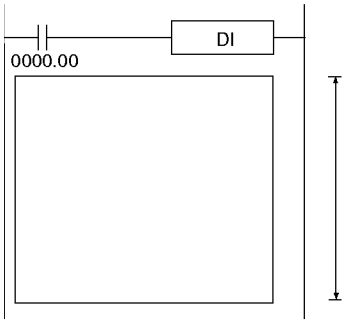
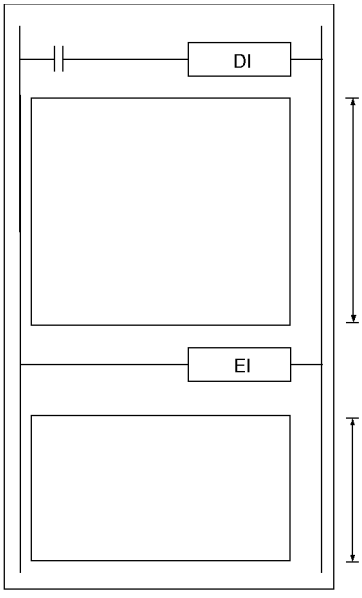
3-17 Subroutine Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
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
SUBROUTINE RETURN RET 093	RET(093)	Indicates the end of a subroutine program.	Output Not required

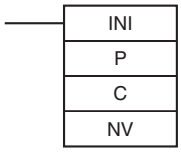
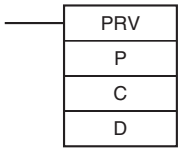
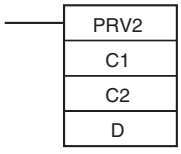
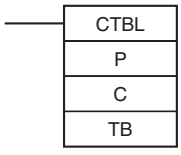
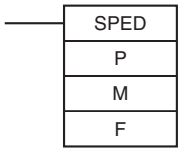
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
GLOBAL SUB-ROUTINE CALL (CS1-H, CJ1-H, CJ1M, or CS1D only) GSBS 750	GSBS(750) N **N**: Subroutine number	Calls the subroutine with the specified subroutine number and executes that program.	Output Not required
GLOBAL SUB-ROUTINE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

3-18 Interrupt Control Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SET INTERRUPT MASK (Not supported by CS1D CPU Units for Duplex-CPU Systems.) MSKS @MSKS 690	MSKS(690) N C **N**: Interrupt number **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
READ INTERRUPT MASK (Not supported by CS1D CPU Units for Duplex-CPU Systems.) MSKR @MSKR 692	MSKR(692) N D **N**: Interrupt number **D**: Destination word	Reads the current interrupt processing settings that were set with MSKS(690).	Output Required

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
CLEAR INTERRUPT (Not supported by CS1D CPU Units for Duplex-CPU Systems.) CLI @CLI 691	CLI(691) N C **N**: Interrupt number **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
DISABLE INTERRUPTS DI @DI 693	DI(693)	Disables execution of all interrupt tasks except the power OFF interrupt. 	Output Required
ENABLE INTERRUPTS EI 694	EI(694)	Enables execution of all interrupt tasks that were disabled with DI(693). 	Output Not required

3-19 High-speed Counter and Pulse Output Instructions (CJ1M-CPU21/22/23 Only)

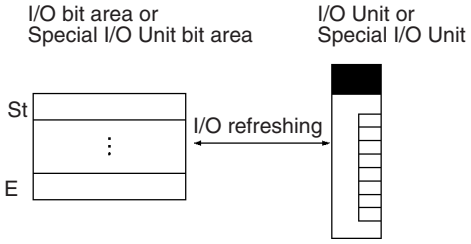
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
MODE CONTROL INI @INI 880	 **P:** Port specifier **C:** Control data **NV:** 1st word with new PV	INI(880) is used to start and stop target value comparison, to change the present value (PV) of a high-speed counter, to change the PV of an interrupt input (counter mode), to change the PV of a pulse output, or to stop pulse output.	Output Required
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
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
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
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

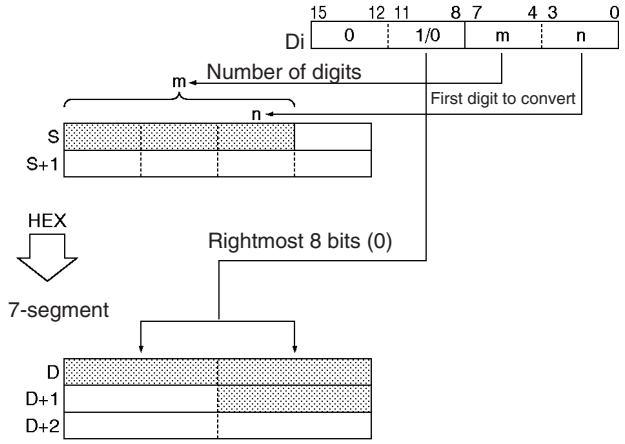
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SET PULSES PULS @PULS 886	PULS P T N **P:** Port specifier **T:** Pulse type **N:** Number of pulses	PULS(886) is used to set the number of pulses for pulse output.	Output Required
PULSE OUTPUT PLS2 @PLS2 887	PLS2 P M S F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

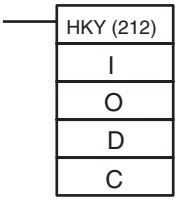
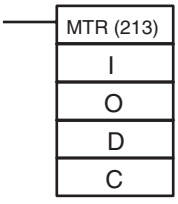
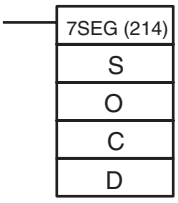
3-20 Step Instructions

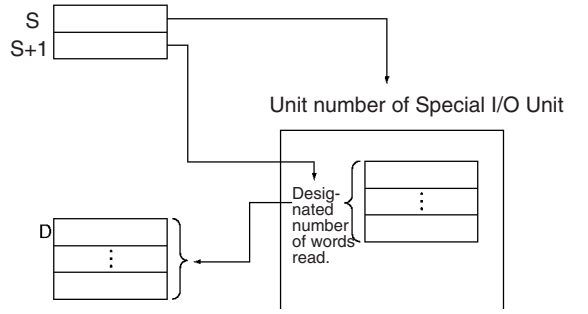
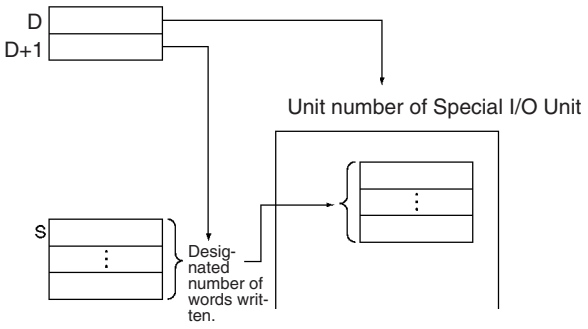
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
STEP DEFINE STEP 008	STEP(008) B **B**: Bit	STEP(008) functions in following 2 ways, depending on its position and whether or not a control bit has been specified. (1)Starts a specific step. (2)Ends the step programming area (i.e., step execution).	Output Required
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

3-21 Basic I/O Unit Instructions

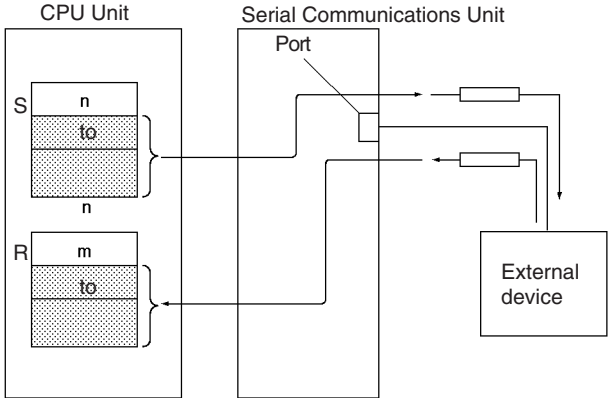
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
I/O REFRESH IORF @IORF 097	IORF(097) St E **St**: Starting word **E**: End word	Refreshes the specified I/O words. 	Output Required
SPECIAL I/O UNIT I/O REFRESH (CJ1-H-R only) FIORF @FIORF 225	FIORF(225) N **N**: Unit number	Immediately refreshes the I/O words allocated to the Special I/O Unit with the specified unit number.	Output Required
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 words allocated to the CPU Bus Unit with the specified unit number.	Output required

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
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

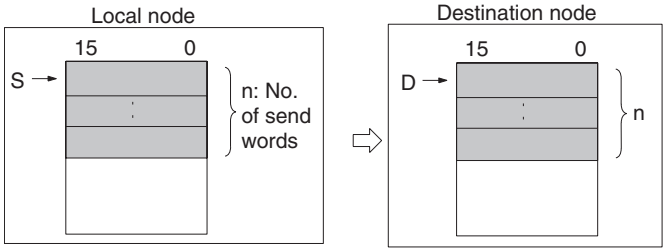
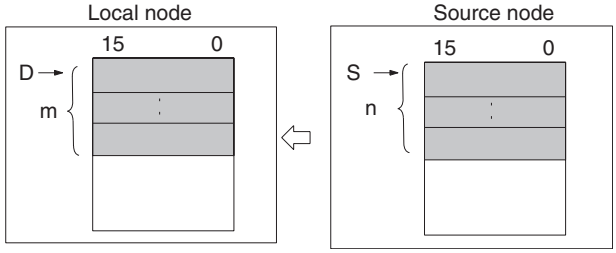
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
INTELLIGENT I/O READ IORD @IORD 222	IORD(222) C S D **C**: Control data **S**: Transfer source and number of words **D**: Transfer destination and number of words	Reads the contents of the memory area for the Special I/O Unit 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
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

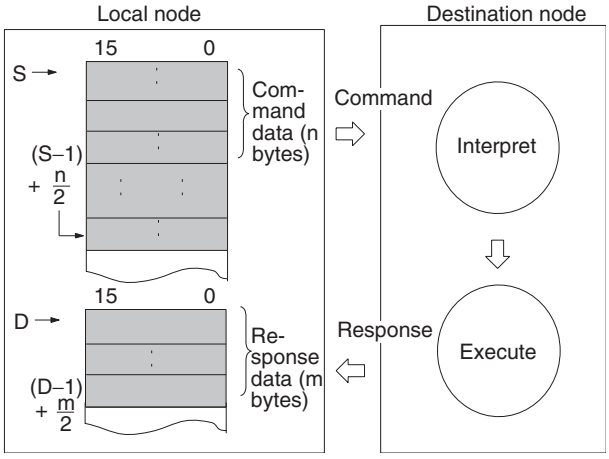
3-22 Serial Communications Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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 without conversion from the RS-232C port built into the CPU Unit (no-protocol mode) or the serial port of a Serial Communications Board or Unit with unit version 1.2 or later (no-protocol mode) according to the start code and end code specified for no-protocol mode in the PLC Setup.	Output Required
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 starting with the specified first word from the RS-232C port built into the CPU Unit (no-protocol mode) or the serial port of a Serial Communications Board or Unit with unit version 1.2 or later (no-protocol mode) according to the start code and end code specified for no-protocol mode in the PLC Setup.	Output Required
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 without conversion from the serial port of a Serial Communications Unit with 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 Area.	Output Required

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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 0100 hex)	Reads the specified number of bytes of data starting with the specified first word from the serial port of a Serial Communications Unit with 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.	
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

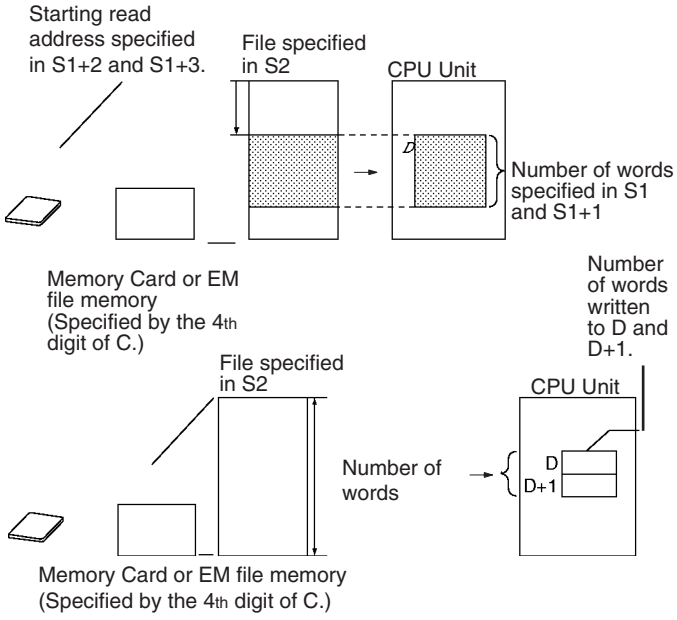
3-23 Network Instructions

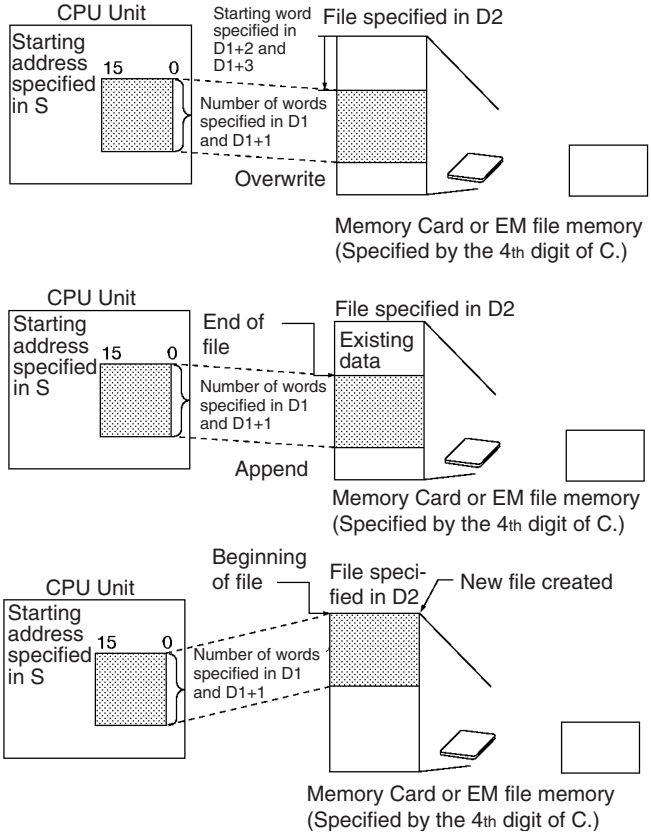
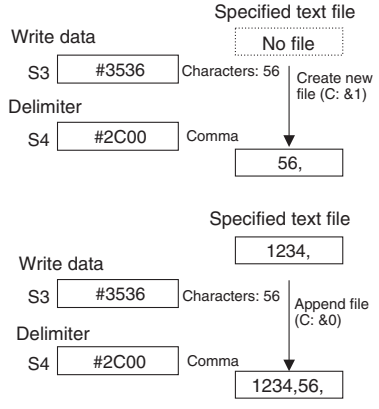
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
NETWORK SEND SEND @SEND 090	SEND(090) S D C **S**: 1st source word **D**: 1st destination word **C**: 1st control word	Transmits data to a node in the network. 	Output Required
NETWORK RECEIVE RECV @RECV 098	RECV(098) S D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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
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
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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
EXPLICIT WORD READ ECHRD 723 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	ECHRD (723) S D C **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
EXPLICIT WORD WRITE ECHWR 724 (CS/CJ-series CPU Unit Ver. 2.0 or later only)	ECHWR (724) S D C **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

3-24 File Memory Instructions

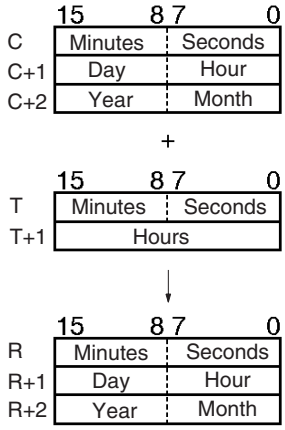
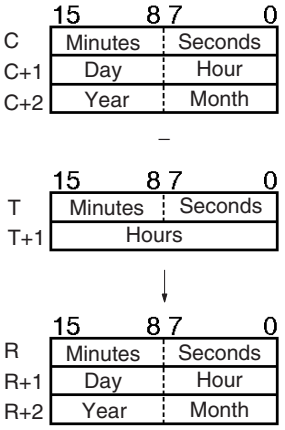
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
READ DATA FILE FREAD @FREAD 700	FREAD(700) C S1 S2 D **C**: Control word **S1**: 1st source word **S2**: Filename **D**: 1st destination word	Reads the specified data or amount of data from the specified data file in file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

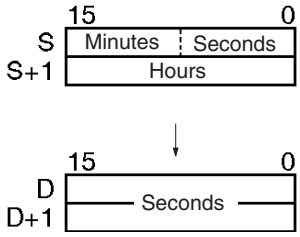
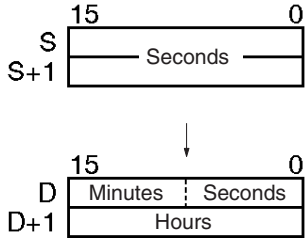
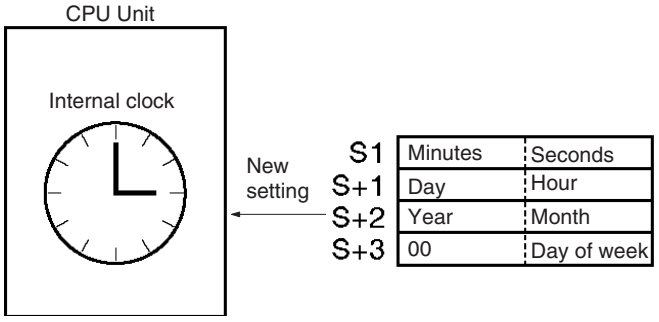
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
WRITE DATA FILE FWRIT @FWRIT 701	FWRIT(701) C D1 D2 S **C**: Control word **D1**: 1st destination word **D2**: Filename **S**: 1st source word	Overwrites or appends data in the specified data file in file memory with the specified 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
WRITE TEXT FILE TWRIT @TWRIT 704	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

3-25 Display Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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

3-26 Clock Instructions

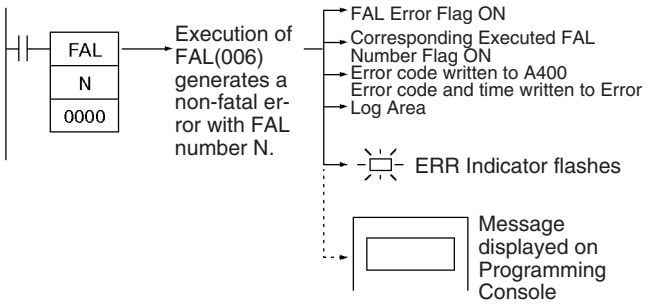
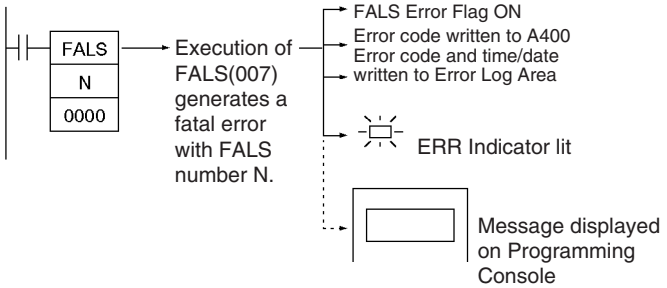
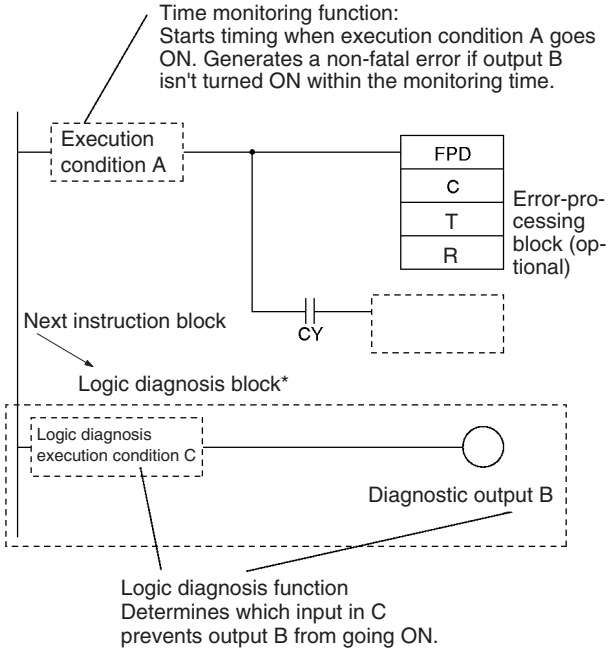
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
CALENDAR ADD CADD @CADD 730	CADD(730) C T R **C**: 1st calendar word **T**: 1st time word **R**: 1st result word	Adds time to the calendar data in the specified words. 	Output Required
CALENDAR SUBTRACT CSUB @CSUB 731	CSUB(731)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

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
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

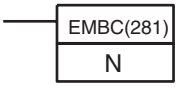
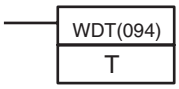
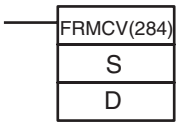
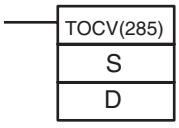
3-27 Debugging Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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

3-28 Failure Diagnosis Instructions

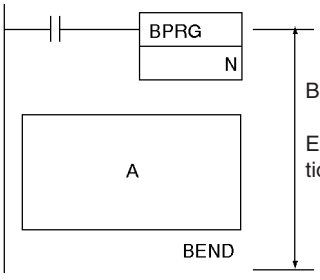
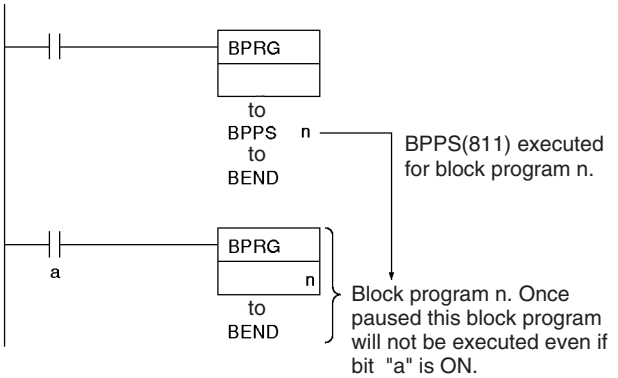
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
FAILURE ALARM FAL @FAL 006	FAL(006) N S **N**: FAL number **S**: 1st message word or error code to generate	Generates or clears user-defined non-fatal errors. Non-fatal errors do not stop PC operation. Also generates non-fatal errors with the system. 	Output Required
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
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

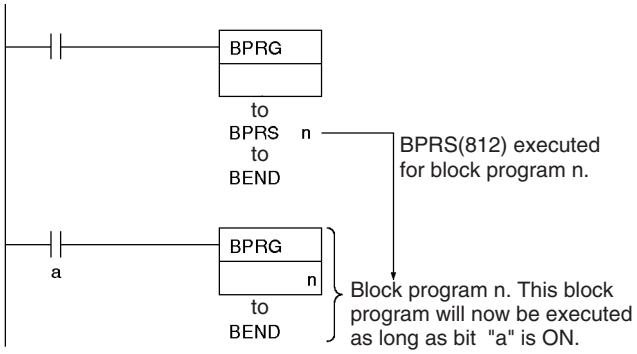
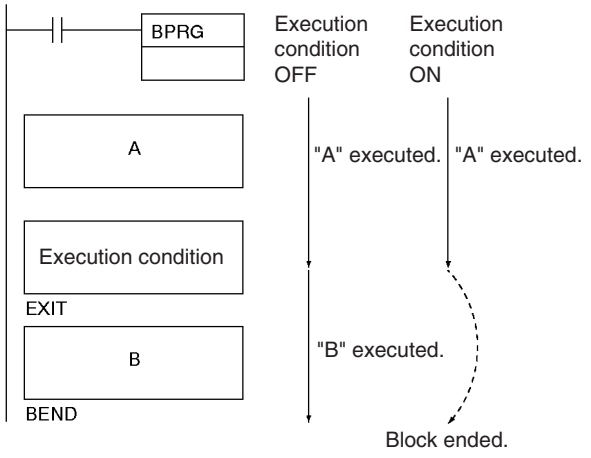
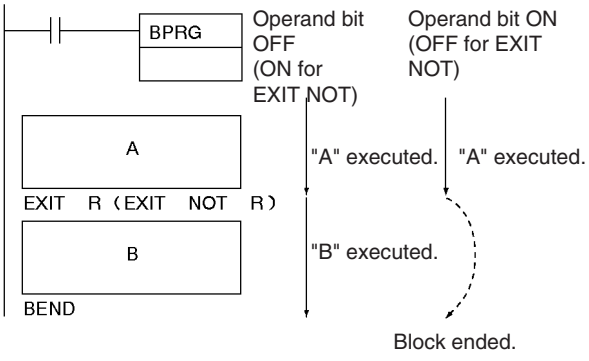
3-29 Other Instructions

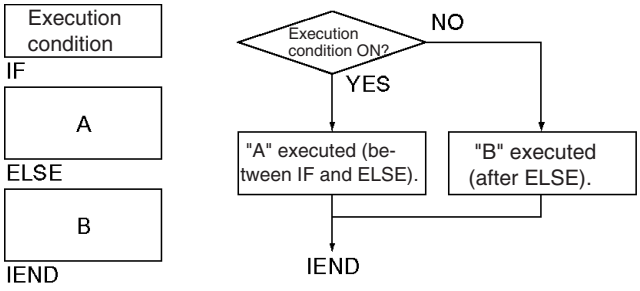
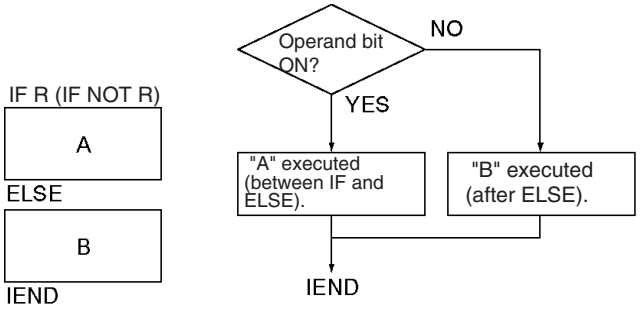
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
SET CARRY STC @STC 040		Sets the Carry Flag (CY).	Output Required
CLEAR CARRY CLC @CLC 041		Turns OFF the Carry Flag (CY).	Output Required
SELECT EM BANK EMBC @EMBC 281	 **N**: EM bank number	Changes the current EM bank.	Output Required
EXTEND MAXIMUM CYCLE TIME WDT @WDT 094	 **T**: Timer setting	Extends the maximum cycle time, but only for the cycle in which this instruction is executed.	Output Required
SAVE CONDITION FLAGS (CS1-H, CJ1-H, CJ1M, or CS1D only) CCS @CCS 282		Saves the status of the condition flags.	Output Required
LOAD CONDITION FLAGS (CS1-H, CJ1-H, CJ1M, or CS1D only) CCL @CCL 283		Reads the status of the condition flags that was saved.	Output Required
CONVERT ADDRESS FROM CV (CS1-H, CJ1-H, CJ1M, or CS1D only) FRMCV @FRMCV 284	 **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
CONVERT ADDRESS TO CV (CS1-H, CJ1-H, CJ1M, or CS1D only) TOCV @TOCV 285	 **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

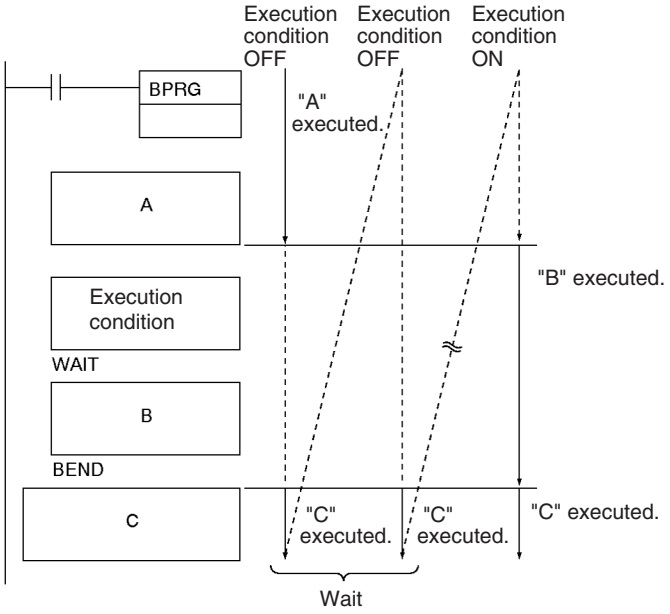
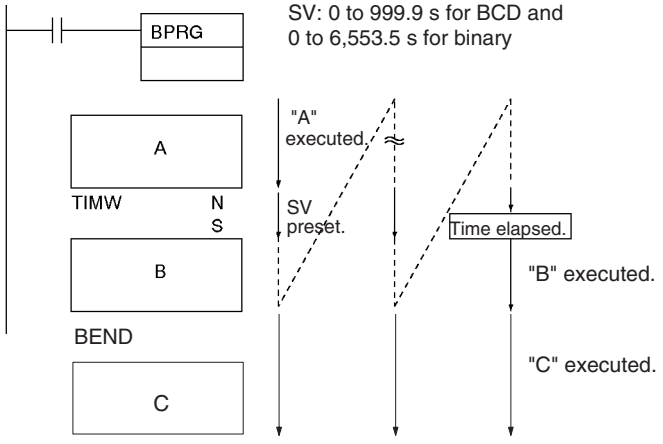
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
DISABLE PERIPHERAL SERVICING (CS1D CPU Units for Single-CPU Systems, CS1-H, CJ1-H, or CJ1M only) IOSP @IOSP 287	IOSP(287)	Disables peripheral servicing during program execution in one of the Parallel Processing Modes or Peripheral Servicing Priority Mode.	Output Required
ENABLE PERIPHERAL SERVICING (CS1D CPU Unit for Single-CPUs 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

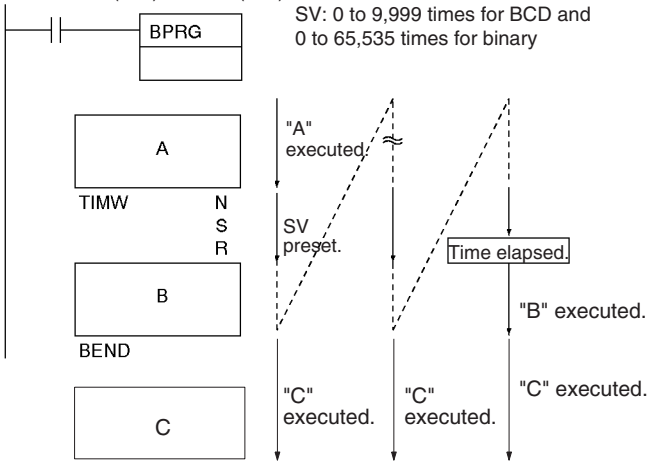
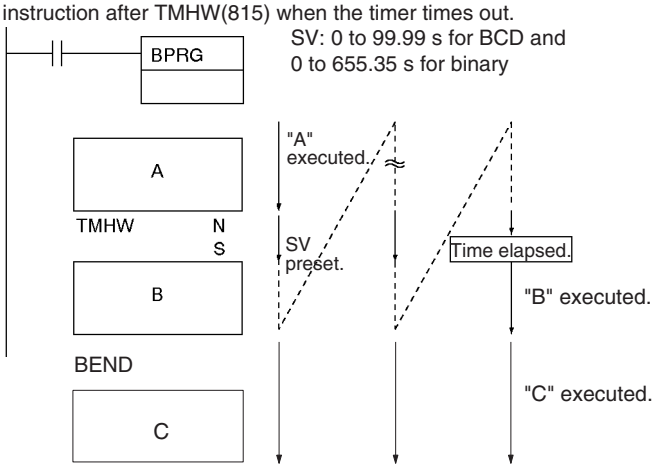
3-30 Block Programming Instructions

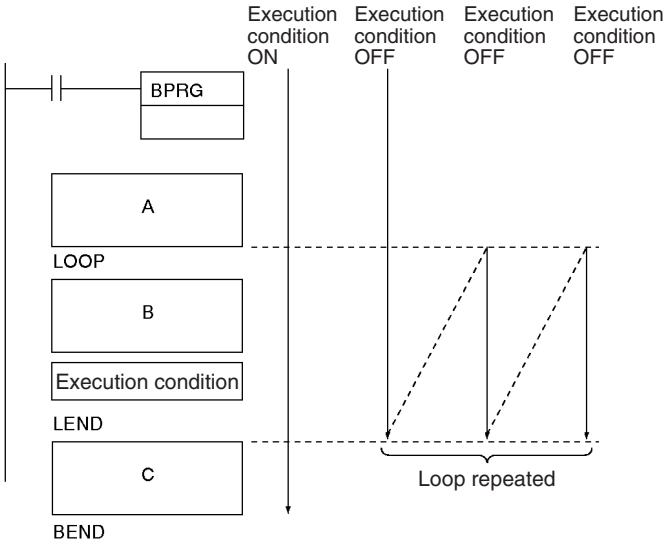
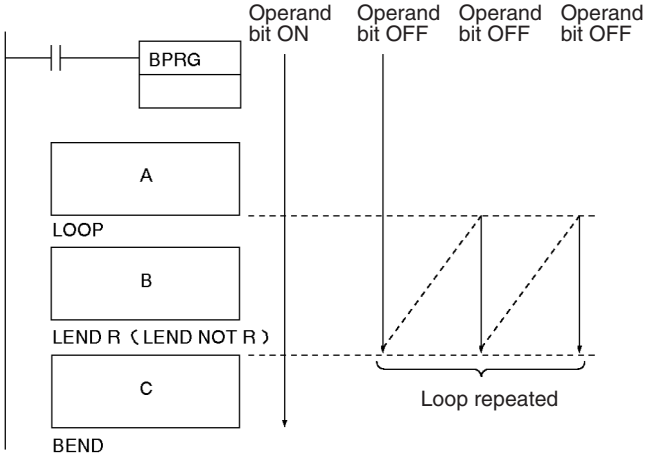
Instruction Mnemonic Code	Symbol/Operand	FunctionS	Location Execution condition
BLOCK PROGRAM BEGIN BPRG 096	BPRG(096) N **N**: Block program number	Define a block programming area. For every BPRG(096) there must be a corresponding BEND(801). 	Output Required
BLOCK PROGRAM END BEND 801		Define a block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

Instruction Mnemonic Code	Symbol/Operand	FunctionS	Location Execution condition
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
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
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
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

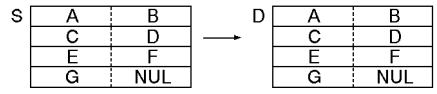
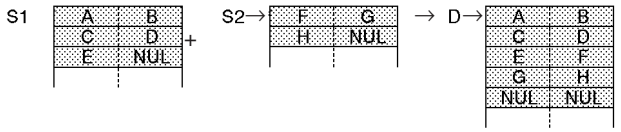
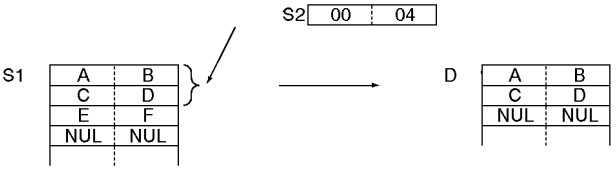
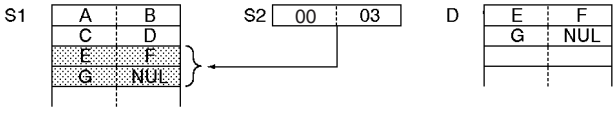
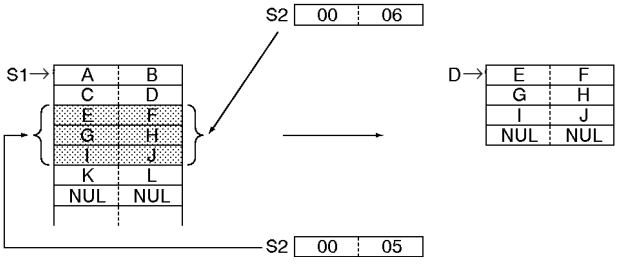
Instruction Mnemonic Code	Symbol/Operand	FunctionS	Location Execution condition
CONDITIONAL BLOCK BRANCHING IF 802	IF (802)	If the execution condition is ON, the instructions between IF(802) and ELSE(803) will be executed and if the execution condition is OFF, the instructions between ELSE(803) and IEND(804) will be executed. 	Block program Required
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
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
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
CONDITIONAL BLOCK BRANCHING END IEND 804	---	If the operand bit is OFF, only the instructions after IEND(804) will be executed.	Block program Required

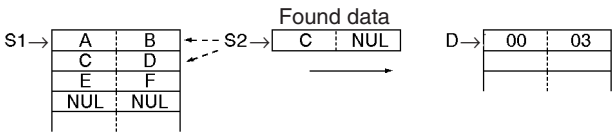
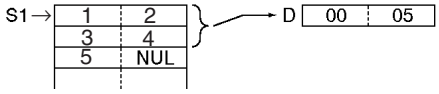
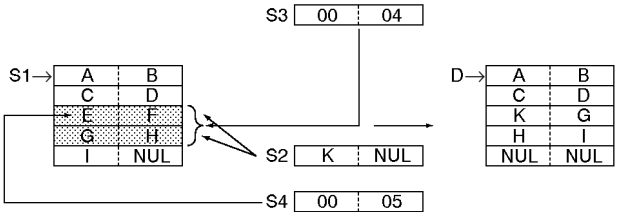
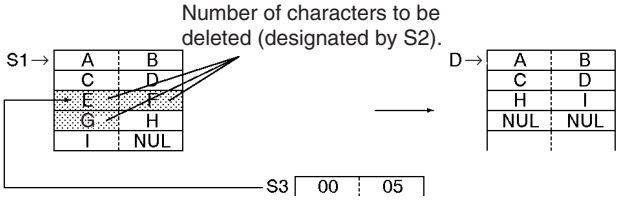
Instruction Mnemonic Code	Symbol/Operand	FunctionS	Location Execution condition
ONE CYCLE AND WAIT WAIT 805	WAIT(805)	If the execution condition is ON for WAIT(805), the rest of the instruction in the block program will be skipped. 	Block program Required
ONE CYCLE AND WAIT WAIT 805	WAIT(805) B **B**: 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
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
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

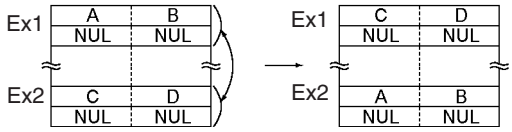
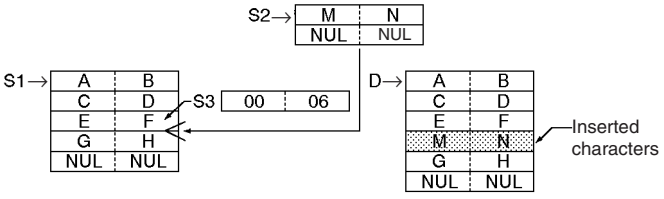
Instruction Mnemonic Code	Symbol/Operand	FunctionS	Location Execution condition
COUNTER WAIT CNTW 814 (BCD) CNTWX 817 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	CNTW(814) N SV **N**: Counter number **SV**: Set value **I**: Count input CNTWX(817) N SV **N**: Counter number **SV**: Set value **I**: Count 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(817) when the counter counts out. SV: 0 to 9,999 times for BCD and 0 to 65,535 times for binary 	Block program Required
HIGH-SPEED TIMER WAIT TMHW 815 (BCD) TMHWX 818 (Binary) (CS1-H, CJ1-H, CJ1M, or CS1D only)	TMHW(815) N SV **N**: Timer number **SV**: Set value TMHW(818) 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) when the timer times out. SV: 0 to 99.99 s for BCD and 0 to 655.35 s for binary 	Block program Required

Instruction Mnemonic Code	Symbol/Operand	FunctionS	Location Execution condition
LOOP LOOP 809	---	LOOP(809) designates the beginning of the loop program. 	Block program Required
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
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
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

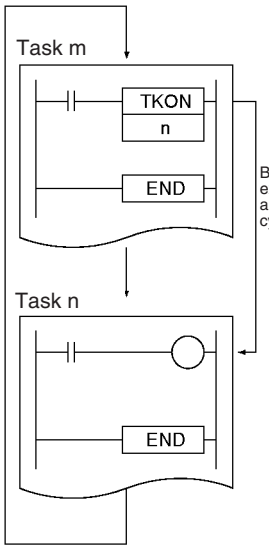
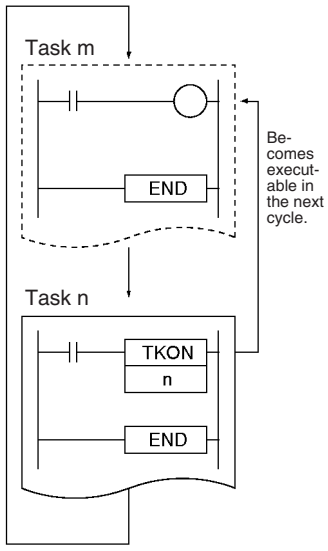
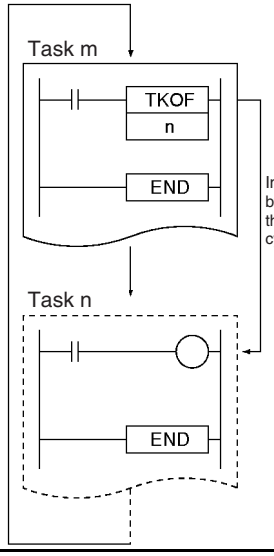
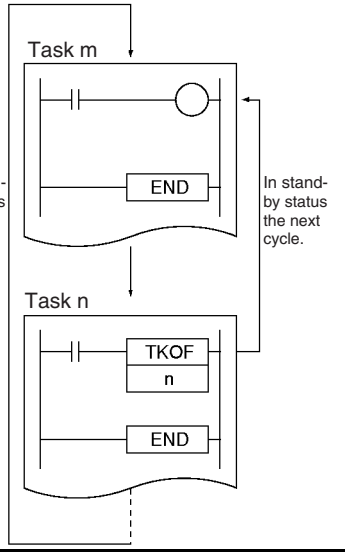
3-31 Text String Processing Instructions

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

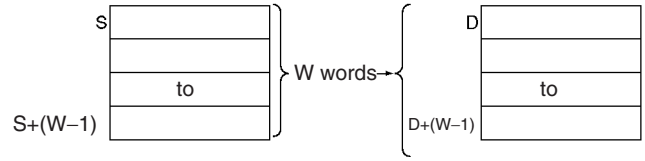
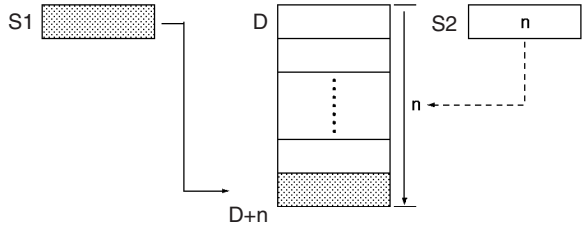
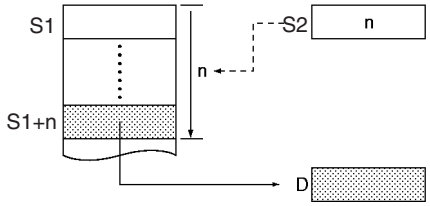
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
FIND IN STRING FIND @FIND$ 660	FIND$(660) S1 S2 D **S1**: Source text string first word **S2**: Found text string first word **D**: First destination word	Finds a designated text string from within a text string. 	Output Required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

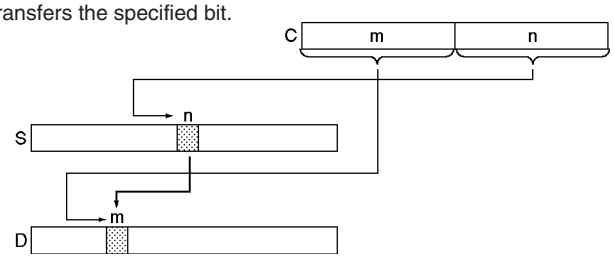
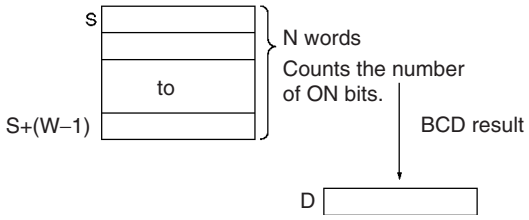
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
EXCHANGE STRING XCHG$ @XCHG$ 665	XCHG$(665) Ex1 Ex2 **Ex1**: 1st exchange word 1 **Ex2**: 1st exchange word 2	Replaces a designated text string with another designated text string. 	Output Required
CLEAR STRING CLR$ @CLR$ 666	CLR$(666) S **S**: Text string first word	Clears an entire text string with NUL (00 hex). 	Output Required
INSERT INTO STRING INS$ @INS$ 657	INS$(657) S1 S2 S3 D **S1**: Base text string first word **S2**: Inserted text string first word **S3**: Beginning position **D**: First destination word	Deletes a designated text string from the middle of a text string. 	Output Required
String Comparison LD, AND, OR + =$, <>$, <$, <=$, >$, >=$ 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

3-32 Task Control Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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

3-33 Model Conversion Instructions (CPU Unit Ver. 3.0 or Later Only)

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
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
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
DATA COLLECT COLLC @COLLC 567	COLLC(567) Bs Of D **Bs**: Source base address **Of**: Offset (BCD) **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 (Stack Read Operation).	Output Required

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
MOVE BIT MOVBC @MOVBC 568	MOVBC(568) S C D **S**: Source word or data **C**: Control word (BCD) **D**: Destination word	Transfers the specified bit. 	Output Required
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

3-34 Special Function Block Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition
GET VARIABLE ID GETID @GETID 286	GETID(286) S D1 D2 **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

SECTION 4
Tasks

This section describes the operation of tasks.

4-1 Task Features

4-1-1 Overview

CS/CJ-series control operations can be divided by functions, controlled devices, processes, developers, or any other criteria and each operation can be programmed in a separate unit called a "task." Using tasks provides the following advantages:

1,2,3...

1. Programs can be developed simultaneously by several people.

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

2. Programs can be standardized in modules.

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

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

3. Improved overall response.

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

4. Easy revision and debugging.

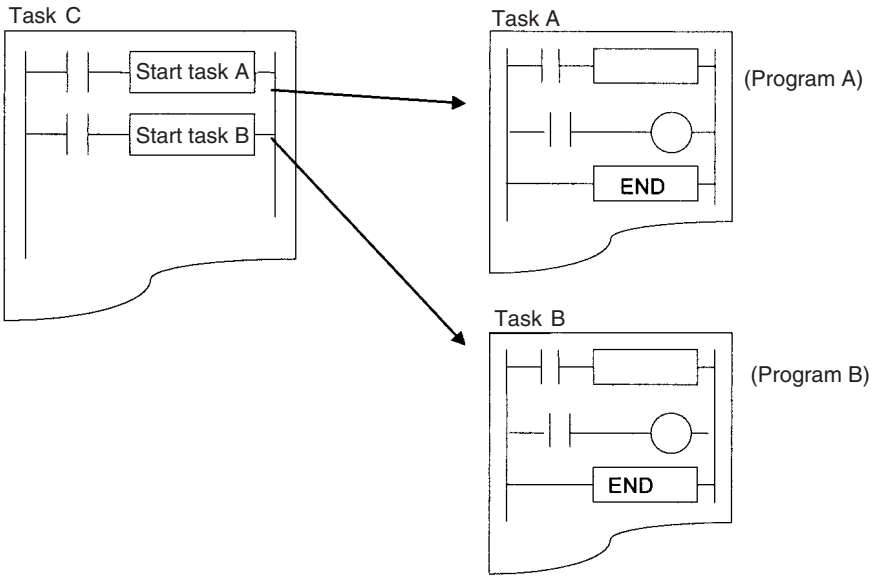
   - Debugging is much more efficient because tasks can be developed separately by several people, and then revised and debugged by individual task.
   - Maintenance is simple because only the task that needs revising will be changed in order to make specification or other changes.
   - Debugging is more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

5. Easy to switch programs.

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

6. Easily understood user programs.

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



4-1-2 Tasks and Programs

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

Note

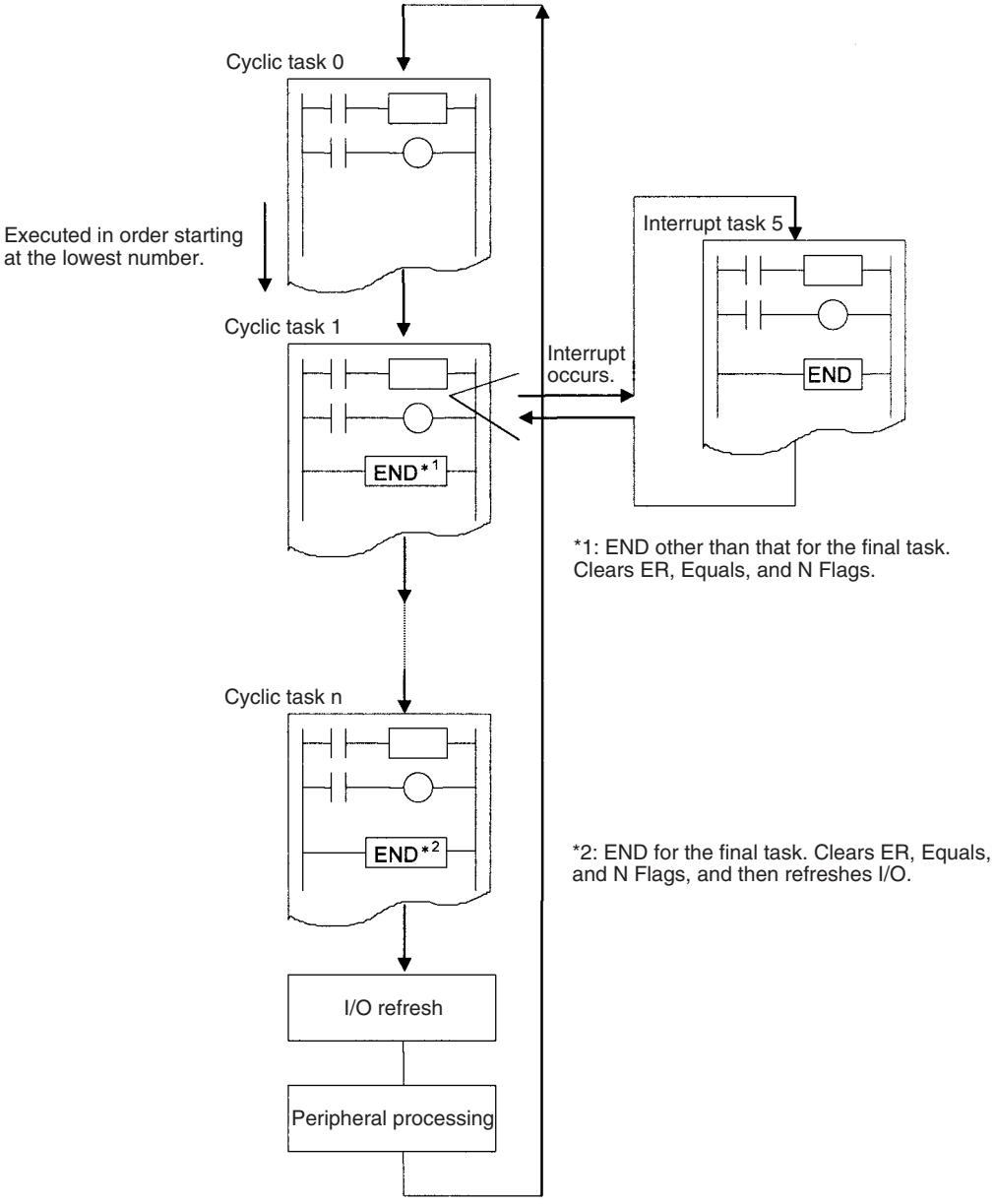
1. Up to 32 cyclic tasks and 256 interrupt tasks for a maximum total of 288 tasks can be created. Each task has its own unique number ranging from 0 to 31 for cyclic tasks and 0 to 255 for interrupt tasks.
2. Interrupt task (interrupt task numbers 0 to 255) can be executed as cyclic tasks by starting them with TKON. These are called “extra cyclic tasks.” If extra cyclic tasks are used, then the total number of cyclic tasks that can be used is 288.
   The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support this function.
3. CJ1□-CPU□□ CPU Units do not currently support I/O interrupt tasks and external interrupt tasks. The maximum number of tasks for a CJ1 CPU Unit is thus 35, i.e., 32 cyclic tasks and 3 interrupt tasks. The total number of programs that can be created and managed is also 35.

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

4-1-3 Basic CPU Unit Operation

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

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support this function.

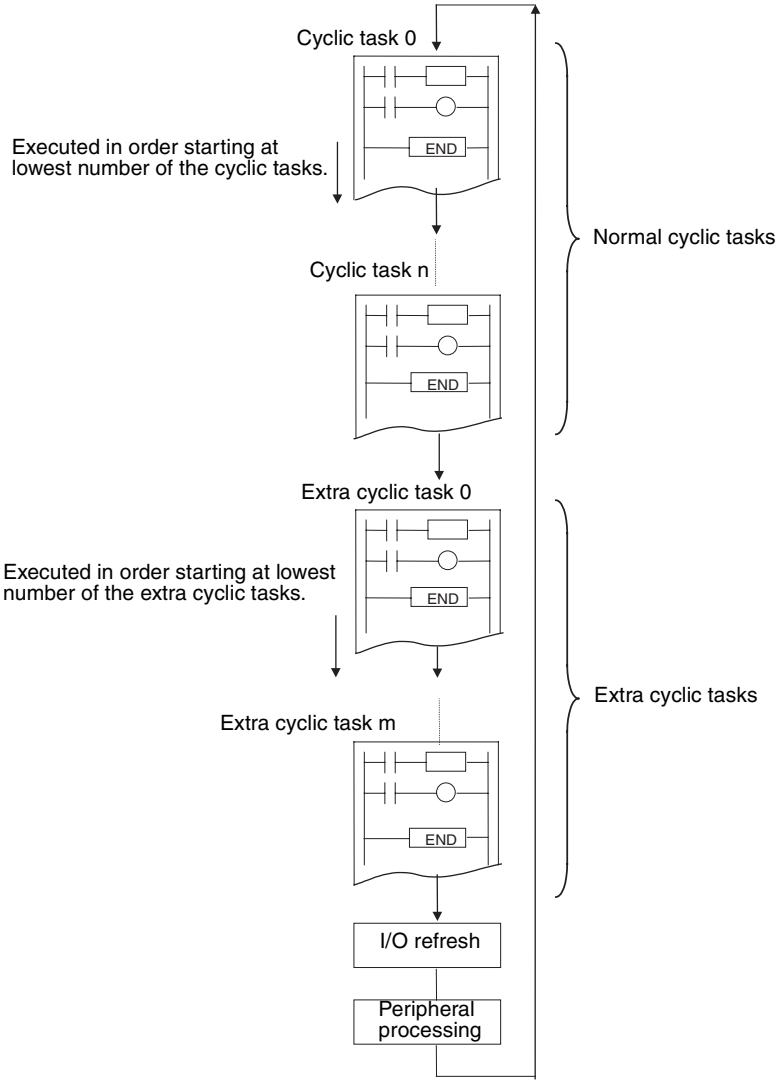


Note All Condition Flags (ER, CY, Equals, AER, etc.) and instruction conditions (interlock ON, etc.) will be cleared at the beginning of a task. Therefore Condition Flags cannot be read nor can INTERLOCK/INTERLOCK CLEAR (IL/ILC) instructions, JUMP/JUMP END (JMP/JME) instructions, or SUBROUTINE CALL/SUBROUTINE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

With a CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit, interrupt task can be executed as cyclic tasks by starting them with TKON. These are called “extra

cyclic tasks." Extra cyclic tasks (interrupt task numbers 0 to 255) are executed starting at the lowest task number after execution of the normal cyclic task (celiac task numbers 0 to 31) has been completed.

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support this function.



4-1-4 Types of Tasks

Tasks are broadly classified as either cyclic tasks or interrupt tasks. Interrupt tasks are further divided into power OFF, scheduled, I/O (CS Series only), and external interrupt tasks (CS Series only). Interrupt tasks can also be executed as extra cyclic tasks.

Note With the CS1-H, CJ1-H, CJ1M, or CS1D Units, interrupt task can be executed as cyclic tasks by starting them with TKON. These are called “extra cyclic tasks.”

Cyclic Tasks

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

If the extra cyclic tasks are used, the CPU Unit supports up to 288 tasks.

Interrupt Tasks

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

The interrupt tasks can also be used as extra cyclic tasks.

Note The CS1D (for Duplex-CPU Systems), CS1G/H-CPU□□ (-V1), and CJ1□-CPU□□ CPU Units do not support this function.

The built-in interrupt inputs and high-speed counter inputs on a CJ1M CPU Unit can be used to activate interrupt tasks. Refer to the *CJ Series Built-in I/O Operation Manual* for details.

Power OFF Interrupt Task

The power OFF interrupt task will be executed if CPU Unit power is shut OFF. Only one power OFF interrupt task can be programmed (Interrupt task number: 1).

Note The power OFF interrupt task must execute before the following time elapses or the task will be forced to quit.

10 ms – (Power OFF detection delay time)

The power OFF detection delay time is set in the PLC Setup.

Scheduled Interrupt Tasks

A scheduled interrupt task will be executed at a fixed interval based on the internal timer of the CPU Unit. The maximum number of scheduled interrupt tasks is 2 (Interrupt task numbers: 2 and 3).

Note The SET INTERRUPT MASK (MSKS(690)) instruction is used to set the interrupt for a scheduled interrupt task. Interrupt times can be set in 10-ms or 1.0-ms increments in the PLC Setup.
With CJ1-H-R/CJ1M CPU Units, the interrupt times can be set in 0.1-ms increments.

I/O Interrupt Tasks

An I/O interrupt task will be executed if an Interrupt Input Unit input turns ON. The maximum number of I/O interrupt tasks is 32 (Interrupt task numbers: 100 to 131). The Interrupt Input Unit must be mounted to the CPU Rack. For CJ1-H CPU Units, the Unit must be connected as one of the five Units next to the CPU Unit (slots 0 to 4). For CJ1M CPU Units, the Unit must be connected as

one of the three Units next to the CPU Unit (slots 0 to 2). I/O Interrupt Units mounted elsewhere cannot be used to request execution of I/O interrupt tasks.

I/O interrupts are not supported by CJ1 CPU Units.

External Interrupt Tasks

An external interrupt task will be executed when requested by an Special I/O Unit, CPU Bus Unit, or Inner Board (CS Series only) user program. Special I/O Units and CPU Bus Units, however, must be mounted to the CPU Rack. The Special I/O Unit or CPU Bus Unit must be mounted to the CPU Rack. For CJ1-H CPU Units, the Unit must be connected as one of the five Units next to the CPU Unit (slots 0 to 4). For CJ1M CPU Units, the Unit must be connected as one of the three Units next to the CPU Unit (slots 0 to 2). Units mounted elsewhere cannot be used to generate external interrupts.

The maximum number of external interrupt tasks is 256 (Interrupt task numbers: 0 to 255). If an external interrupt task has the same number as a power OFF, scheduled, or I/O interrupt task, the interrupt task will be executed for either condition (the two conditions will operate with OR logic) but basically task numbers should not be duplicated.

I/O interrupts are not supported by CJ1 CPU Units.

Note

1. If another interrupt task is being executed when a scheduled, I/O, or external interrupt occurs, then these interrupt tasks will not be executed until the interrupt task that is currently being executed has been completed. If multiple interrupts occur simultaneously, then interrupt tasks will be executed sequentially starting at the lowest interrupt task number.
2. The CJ1 CPU Units do not support I/O interrupt and external interrupt tasks.

Extra Cyclic Tasks

An interrupt tasks can be executed every cycle, just like the normal cyclic tasks. Extra cyclic tasks (interrupt task numbers 0 to 255) are executed starting at the lowest task number after execution of the normal cyclic task (cyclic task numbers 0 to 31) has been completed. The maximum number of extra cyclic tasks is 256 (Interrupt task numbers: 0 to 255). Cycle interrupt tasks, however, are different from normal cyclic tasks in that they are started with the TKON(820)instruction.

If an extra cyclic task has the same number as a power OFF, scheduled, or I/O interrupt task, the interrupt task will be executed for either condition (the two conditions will operate with OR logic). Do not use interrupt tasks both as normal interrupt tasks and as extra cyclic tasks.

Note

1. TKON(820) and TKOF(821) can be input and executed in an extra cyclic task, but they will not be executed when the task operates as an interrupt task.
2. The differences between normal cyclic tasks and extra cyclic tasks are listed in the following table.

Item	Extra cyclic tasks	Normal cyclic tasks
Activating at startup	Setting is not possible.	Set from CX-Programmer
Task Flags	Not supported.	Supported. (Cyclic task numbers 00 to 31 correspond to Task Flags TK00 to TK31.)
Initial Task Execution Flag (A20015) and Task Start Flag (A20014)	Not supported.	Supported.

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

3. The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support this function.

4-1-5 Task Execution Conditions and Settings

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

Task		No.	Execution condition	Related Setting
Cyclic tasks		0 to 31	Executed once each cycle if READY (set to start initially or started with the TKON(820)instruction) when the right to execute is obtained.	None
Interrupt tasks	Power OFF interrupt task	Interrupt task 1	Executes when CPU Unit power shuts OFF.	• Power OFF interrupt enabled in PLC Setup.
	Scheduled interrupt tasks 0 and 1	Interrupt tasks 2 and 3	Executes once every time the preset period elapses according to the internal timer of CPU Unit.	• The scheduled interrupt time is set (0 to 9999) through the SET INTERRUPT MASK instruction (MSKS). • Scheduled interrupt unit (10 ms or 1.0 ms) is set in PLC Setup. (See note 3.)
	I/O interrupt tasks 00 to 31	Interrupt tasks 100 to 131	Executes when an input on an Interrupt Input Unit on the CPU Rack turns ON.	• Masks for designated inputs are canceled through the SET INTERRUPT MASK instruction (MSKS).
	External interrupt tasks 0 to 255	Interrupt tasks 0 to 255	Executes when requested by a user program in a Special I/O Unit or CPU Bus Unit on the CPU Rack or by a user program in an Inner Board (CS Series only).	None (always enabled)
Extra cyclic tasks		Interrupt tasks 0 to 255	Executed once each cycle if READY (started with the TKON(820)instruction) when the right to execute is obtained.	None (always enabled)

Note

1. The Interrupt Input Unit must be mounted to the CPU Rack. For CJ1-H CPU Units, the Unit must be connected as one of the five Units next to the CPU Unit (slots 0 to 4). For CJ1M CPU Units, the Unit must be connected as one of the three Units next to the CPU Unit (slots 0 to 2). I/O Interrupt Units mounted elsewhere cannot be used to request execution of I/O interrupt tasks
2. The Special I/O Unit or CPU Bus Unit must be mounted to the CPU Rack. For CJ1-H CPU Units, the Unit must be connected as one of the five Units next to the CPU Unit (slots 0 to 4). For CJ1M CPU Units, the Unit must be connected as one of the three Units next to the CPU Unit (slots 0 to 2). Units mounted elsewhere cannot be used to generate external interrupts.
3. The CJ1-H-R and CJ1M CPU Units can use a 0.1-ms time units setting.

4. The number of cyclic tasks and interrupt tasks are limited when the memory clear operation is performed with a Programming Console.
   - Only cyclic task 0 can be created.
     Cyclic tasks 1 to 31 cannot be created with a Programming Console, but these tasks can be edited if they were already created with CX-Programmer.
   - Only interrupt tasks 1, 2, 3, and 100 through 131 (CS Series only) can be created.
     Interrupt tasks 0, 4 through 99, and 132 through 255 cannot be created with a Programming Console (except that 140 through 143 can be created for CJ1M CPU Units), but these tasks can be edited if they were already created with CX-Programmer.

4-1-6 Cyclic Task Status

The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support this function

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

Disabled Status (INI)

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

READY Status

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

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

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

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

RUN Status

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

Standby Status

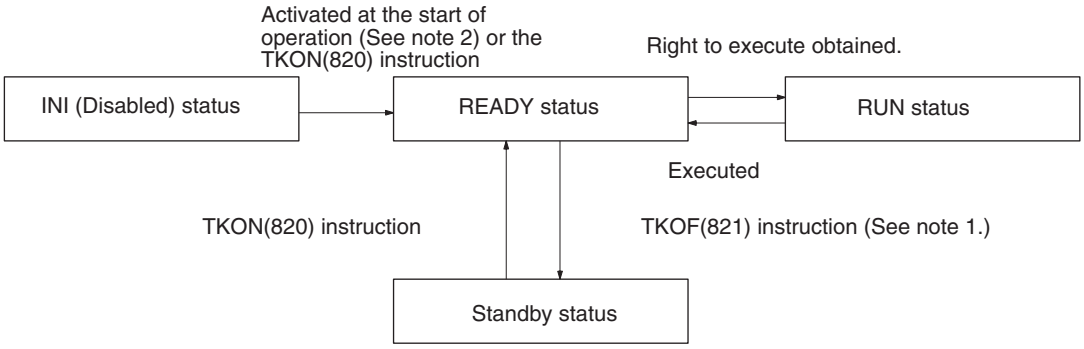
A TASK OFF (TKOF(821)) instruction can be used to change a cyclic task from Disabled status to Standby status.

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

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

Note This function is supported by CX-Programmer version 4.0 or higher.

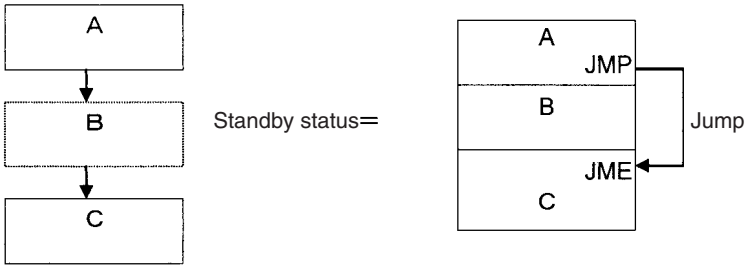
4-1-7 Status Transitions



Note

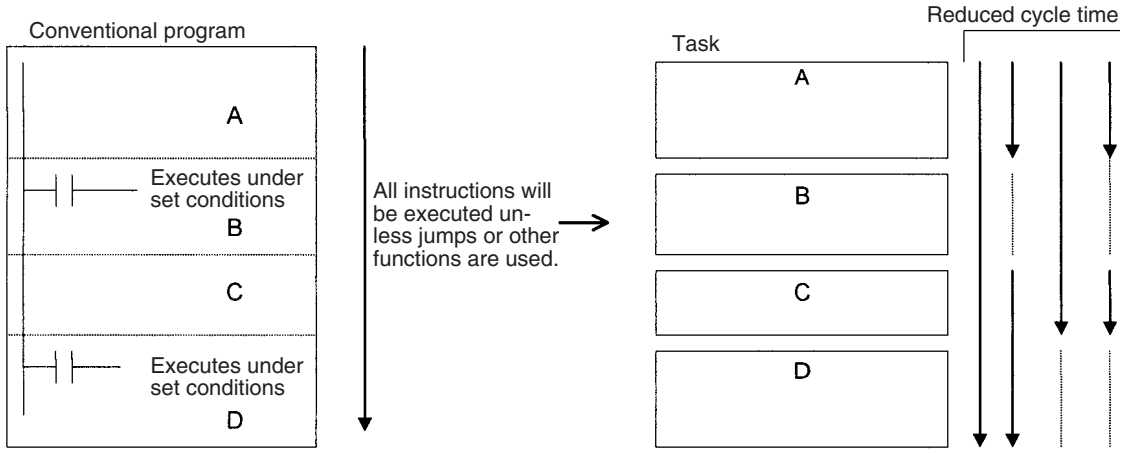
1. A task in RUN status will be put into Standby status by the TKOF(821) instruction even when the TKOF(821) instruction is executed within that task.
2. Activation at the start of operation is possible for normal cyclic tasks only. It is not possible for extra cyclic tasks.

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



Instructions will not be executed in Standby status, so instruction execution time will not be increased. Programming that does not need to be executed all

the time can be made into tasks and assigned Standby status to reduce cycle time.



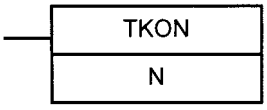
Note Standby status simply means that a task will be skipped during task execution. Changing to Standby status will not end the program.

4-2 Using Tasks

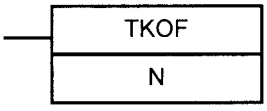
4-2-1 TASK ON and TASK OFF

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

Note Extra cyclic tasks are supported only by CS1-H, CJ1-H, CJ1M, or CS1D CPU Units.



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



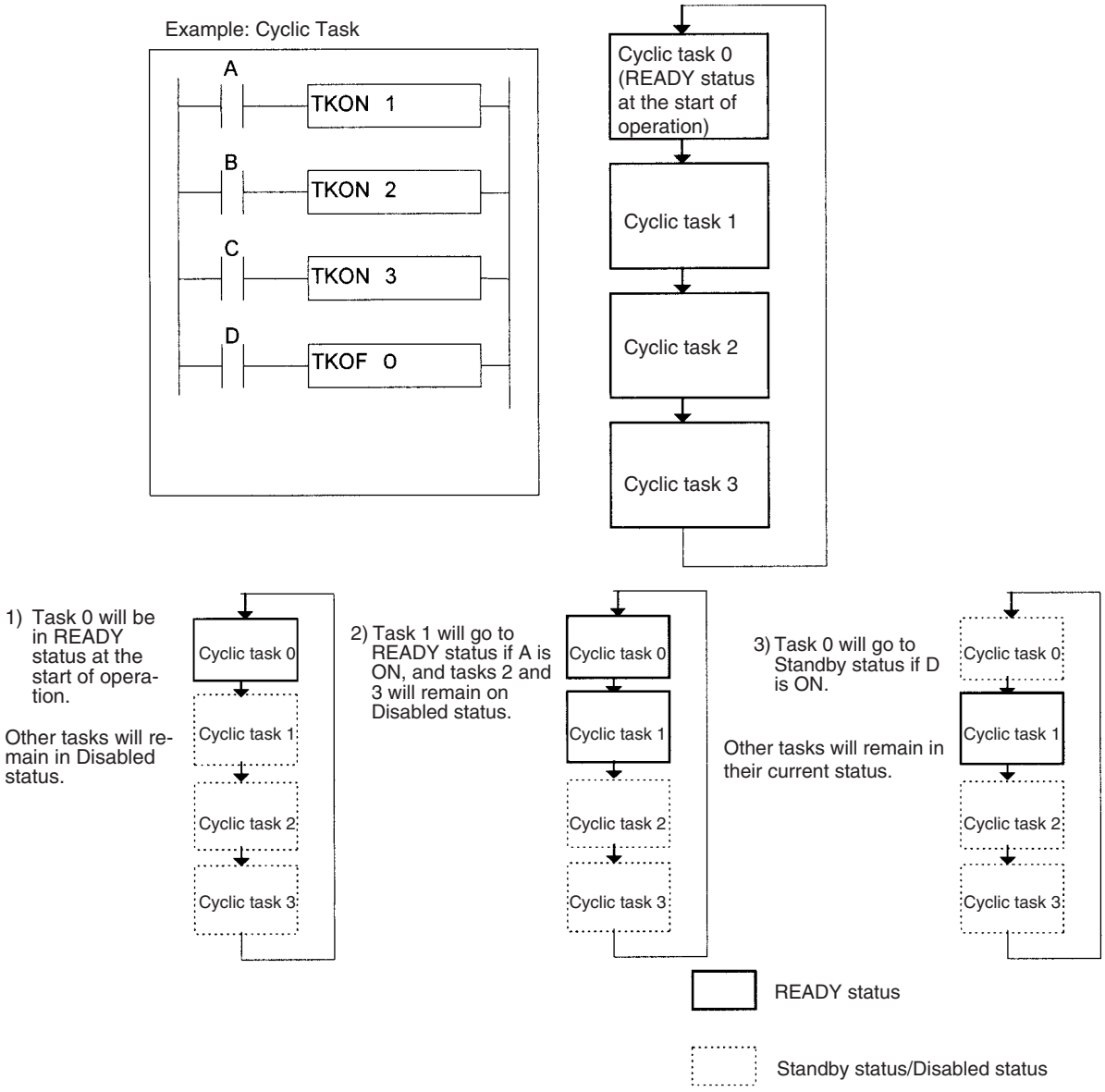
N: Task No. A task will go to Standby status when the execution condition is ON, and the corresponding Task Flag will turn OFF.

Note: Task Flags do not work for extra cyclic tasks.

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

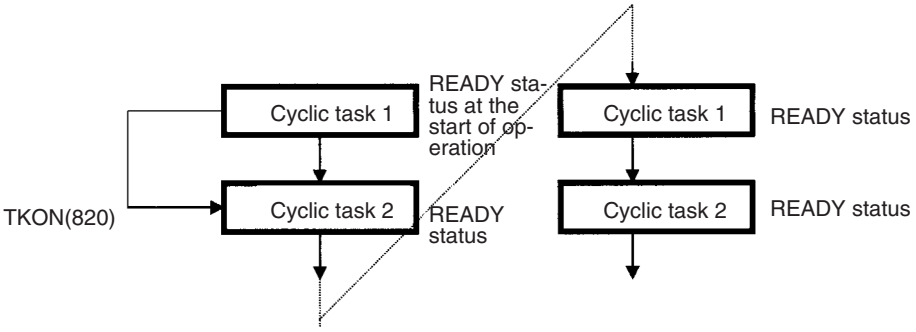
The TASK ON and TASK OFF instructions can be used only with cyclic tasks and not with interrupt tasks.

Note At least one cyclic task must be in READY status in each cycle. If there is not cyclic task in READY status, the Task Error Flag (A29512) will turn ON, and the CPU Unit will stop running.

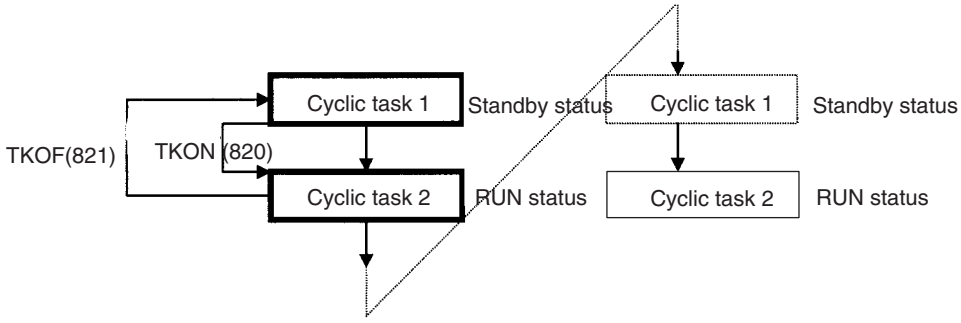


Tasks and the Execution Cycle

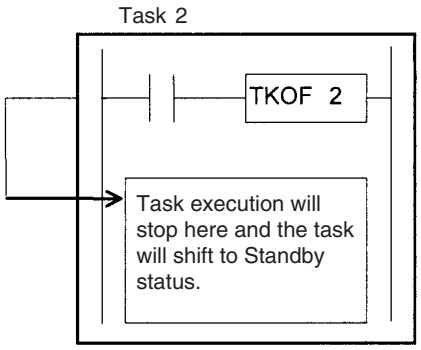
A cyclic task (including an extra cyclic task) that is in READY status will maintain that status in subsequent cycles.



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



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



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

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

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

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

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

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

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

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

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

Relationship of Tasks to I/O Memory

There are two different ways to use Index Registers (IR) and Data Registers (DR): 1) Independently by task or 2) Shared by all task (supported by CS1-H, CJ1-H, CJ1M, or CS1D CPU Units only).

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

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

- Other words and bits in I/O Memory are shared by all tasks. CIO 001000 for example is the same bit for both cyclic task 1 and cyclic task 2. Therefore, be very careful in programming any time I/O memory areas other than the IR and DR Areas are used because values changed with one task will be used by other tasks.

I/O memory	Relationship to tasks
CIO, Auxiliary, Data Memory and all other memory areas except the IR and DR Areas. (See note 1.)	Shared with other tasks.
Index registers (IR) and data registers (DR) (See note 2.)	Used separately for each task.

Note

1. The current EM bank is also shared by tasks. Therefore if the current EM bank number is changed with cyclic task 1 for example, the new current EM bank number will be valid for cyclic task 2 as well.
2. IR and DR values are not set when interrupt tasks (including extra cyclic tasks) are started. If IR and DR are used in an interrupt task, these values must be set by the MOVR/MOVRW (MOVE TO REGISTER and MOVE TIMER/COUNTER PV TO REGISTER) instructions within the interrupt task. After the interrupt task has been executed, IR and DR will return to their values prior to the interrupt automatically.

Relationship of Tasks to Timer Operation

Timer present values for TIM, TIMX, TIMH, TIMHX, TMHH, TMHHX, TIMU, TIMUX (See note.), TMUH, TMUHX (See note.), TIMW, TIMWX, TMHW, and TMHWX programmed for timer numbers 0000 to 2047 will be updated even if the task is switched or if the task containing the timer is changed to Standby status or back to READY status.

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

Note These instructions are supported by the CJ1-H-R CPU Units only.

- The present values for timers programmed with timer numbers 2048 to 4098 will be maintained when the task is in Standby status.

Relationship of Tasks to Condition Flags

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

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support CCS(173) and CCL(172).

Note When the status of Condition Flags is monitored from a Programming Console, the Programming Console will show the flags' status at the end of the cycle, i.e., their status at the end of the last task in the cycle.

4-2-2 Task Instruction Limitations

Instructions Required in the Same Task

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

Mnemonic	Instruction
JMP/JME	JUMP/JUMP END
CJP/JME	CONDITIONAL JUMP/JUMP END
CJPN/JME	CONDITIONAL JUMP NOT/CONDITIONAL JUMP END
JMP0/JME0	MULTIPLE JUMP/JUMP END
FOR/NEXT	FOR/NEXT
IL/ILC	INTERLOCK/INTERLOCK CLEAR
SBS/SBN/RET	SUBROUTINE CALL/SUBROUTINE ENTRY/SUBROUTINE RETURN Note: If the global subroutine instructions (GSBS(750), GSBN(751), and GRET(752)) are used, the subroutine can also be called from another task.
MCRO/SBN/RET	MACRO/SUBROUTINE ENTRY/SUBROUTINE RETURN
BPRG/BEND	BLOCK PROGRAM BEGIN/BLOCK PROGRAM END
STEP/STEP	STEP DEFINE

Instructions Not Allowed in Interrupt Tasks

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

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

The operation of the following instructions is unpredictable in an interrupt task: HUNDRED-MS TIMER: TIM and TIMX(550), TEN-MS TIMER: TIMH(015) and TIMHX(551), ONE-MS TIMER: TMHH(540) and TMHHX(552), TENTH-MS TIMER: TIMU(541) and TIMUX(556) (See note.), HUNDREDTH-MS TIMER TIMUH(544) and TIMUHX(557) (See note.), ACCUMULATIVE TIMER: TTIM(087) and TTIMX(555), MULTIPLE OUTPUT TIMER: MTIM(543) and MTIMX(554), LONG TIMER: TIML(542) and TIMLX(553), TIMER WAIT: TIMW(813) and TIMWX(816), HIGH-SPEED TIMER WAIT: TMHW(815) and TMHWX(817), PID CONTROL: PID(190), FAILURE POINT DETECTION: FPD(269), and CHANGE SERIAL PORT SETUP: STUP(237).

Note These instructions are supported by the CJ1-H-R CPU Units only.

The following instructions cannot be used in the power OFF interrupt task (they will not be executed even if they are used and the Error Flag will **not** turn ON):
READ DATA FILE: FREAD(700), WRITE DATA FILE: FWRIT(701), WRITE TEXT FILE(TWRIT(704)), NETWORK SEND: SEND(090), NETWORK RECEIVE: RECV(098), DELIVER COMMAND: CMND(490), TRANSMIT:

TXD(236), RECEIVE: RXD(235), and PROTOCOL MACRO: PMCR(260), EXPLICIT MESSAGE SEND: EXPLT(270), EXPLICIT GET ATTRIBUTE: EGATR(271), EXPLICIT SET ATTRIBUTE: ESATR(272), EXPLICIT WORD READ: ECHRD(273), EXPLICIT WORD WRITE: ECHWR(274), TRANSMIT VIA SERIAL COMMUNICATIONS UNIT: TXDU(256), and RECEIVE VIA SERIAL COMMUNICATIONS UNIT: RXDU(255).

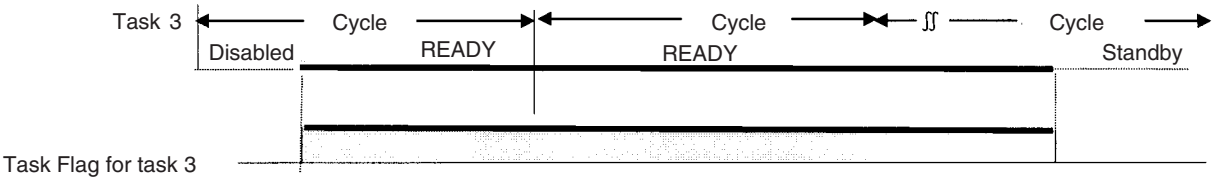
4-2-3 Flags Related to Tasks

Flags Related to Cyclic Tasks

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

Task Flags (TK00 to TK31)

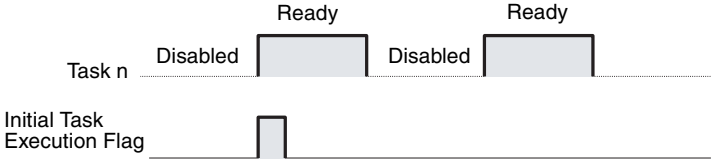
A Task Flag is turned ON when a cyclic task in READY status and is turned OFF when the task is in Disabled (INI) or in Standby (WAIT) status. Task numbers 00 to 31 correspond to Task Flags TK00 to TK31.



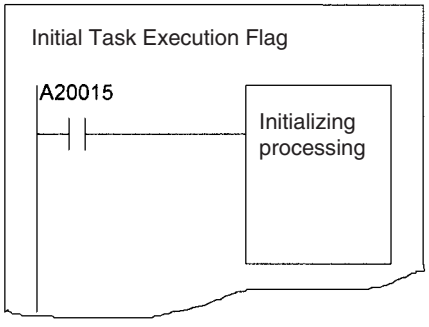
Note Task Flags are used only with cyclic tasks and not with interrupt tasks. With an interrupt task, A44115 will turn ON if an interrupt task executes after the start of operation, and the number of the interrupt task that required for maximum processing time will be stored in two-digit hexadecimal in A44100 to A44107.

Initial Task Execution Flag (A20015)

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



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

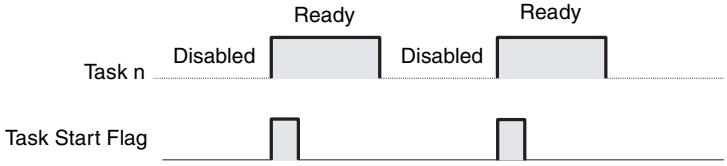


Note Even though a Standby cyclic task is shifted back to READY status through the TKON(820) instruction, this is not considered an initial execution and the Initial Task Execution Flag (20015) will not turn ON. The Initial Task Execution

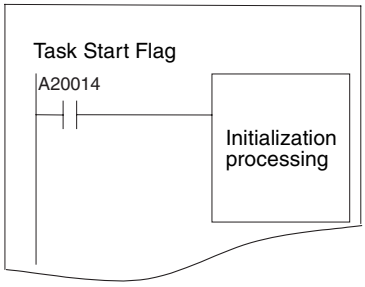
Flag (20015) will also not turn ON if a cyclic task is shifted from Disabled to RUN status or if it is put in Standby status by another task through the TKOF(821) instruction before the right to execute actually is obtained.

Task Start Flag (A20014, CS1-H, CJ1-H, CJ1M, or CS1D CPU Units only)

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



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



Flags Related to All Tasks

Task Error Flag (A29512)

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

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

Task Number when Program Stopped (A294)

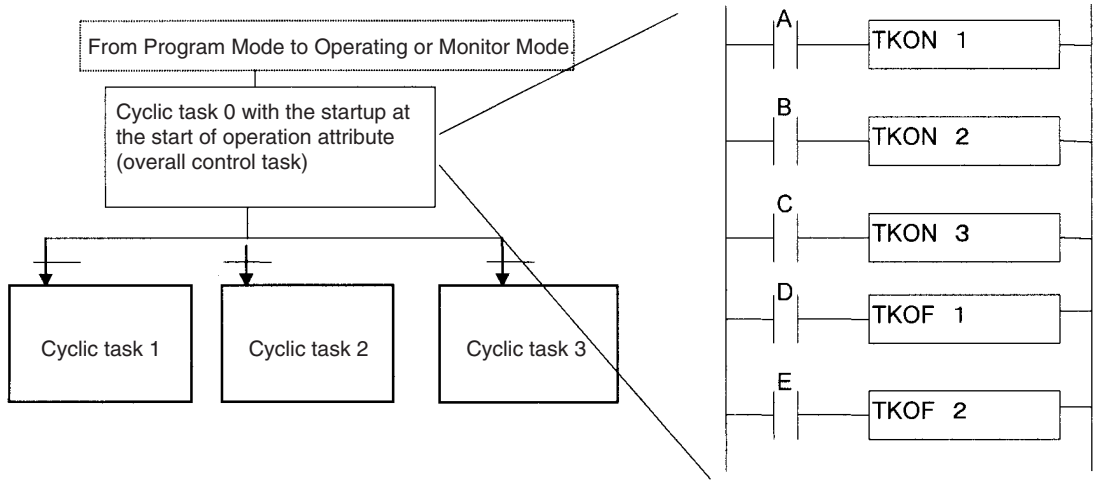
The type of task and the current task number when a task stops execution due to a program error will be stored as follows:

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

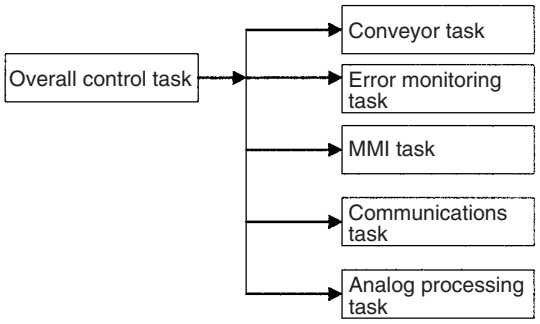
This information makes it easier to determine where the fatal error occurred, and it will be cleared when the fatal error is cleared. The program address where task operation stopped is stored in A298 (rightmost bits of the program address) and in A299 (leftmost bits of the program address).

Examples of Tasks

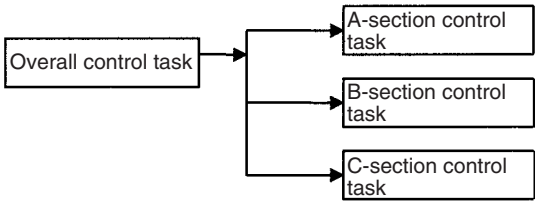
An overall control task that is set to go to READY status at the start of operation is generally used to control READY/Standby status for all other cyclic tasks (including extra cyclic tasks). Of course, any cyclic task can control the READY/Standby status of any other cyclic task as required by the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



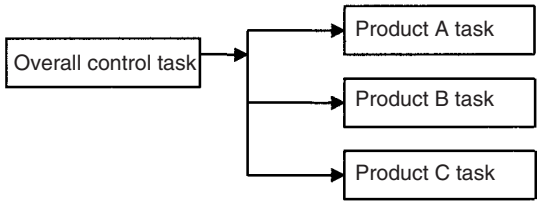
Tasks Separated by Function



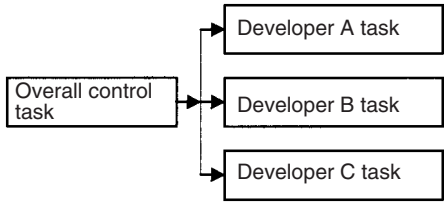
Tasks Separated by Controlled Section



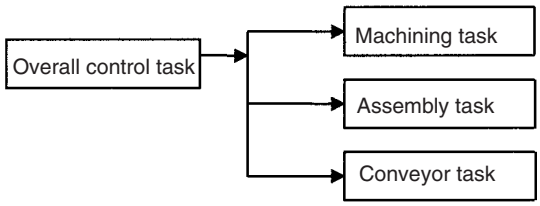
Tasks Separated by Product



Tasks Separated by Developer



Tasks Separated by Process



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

4-2-4 Designing Tasks

We recommend the following guidelines for designing tasks.

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

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

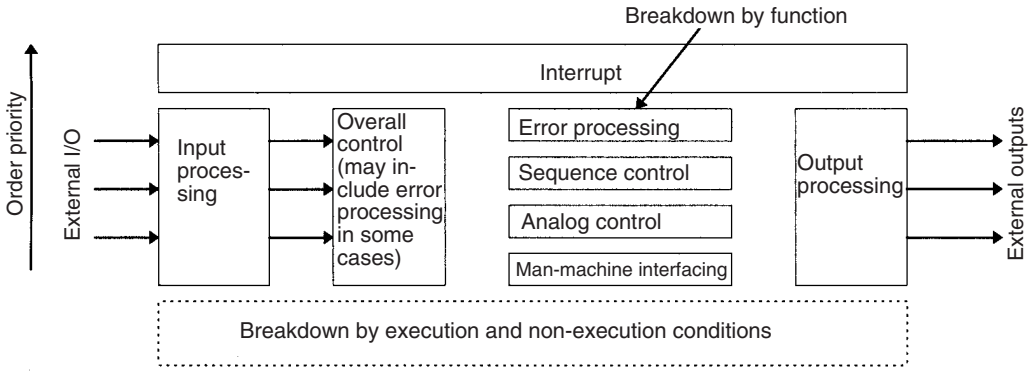
b) Summarize the presence or absence of external I/O.

c) Summarize functions.

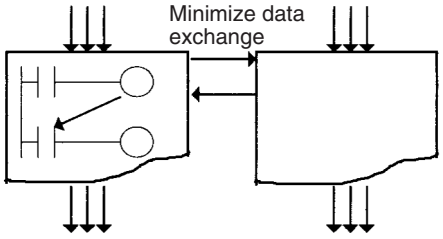
Keep data exchanged between tasks for sequence control, analog control, man-machine interfacing, error processing and other processes to an absolute minimum in order to maintain a high degree of autonomy.

d) Summarize execution in order of priority.

Separate processing into cyclic and interrupt tasks.



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



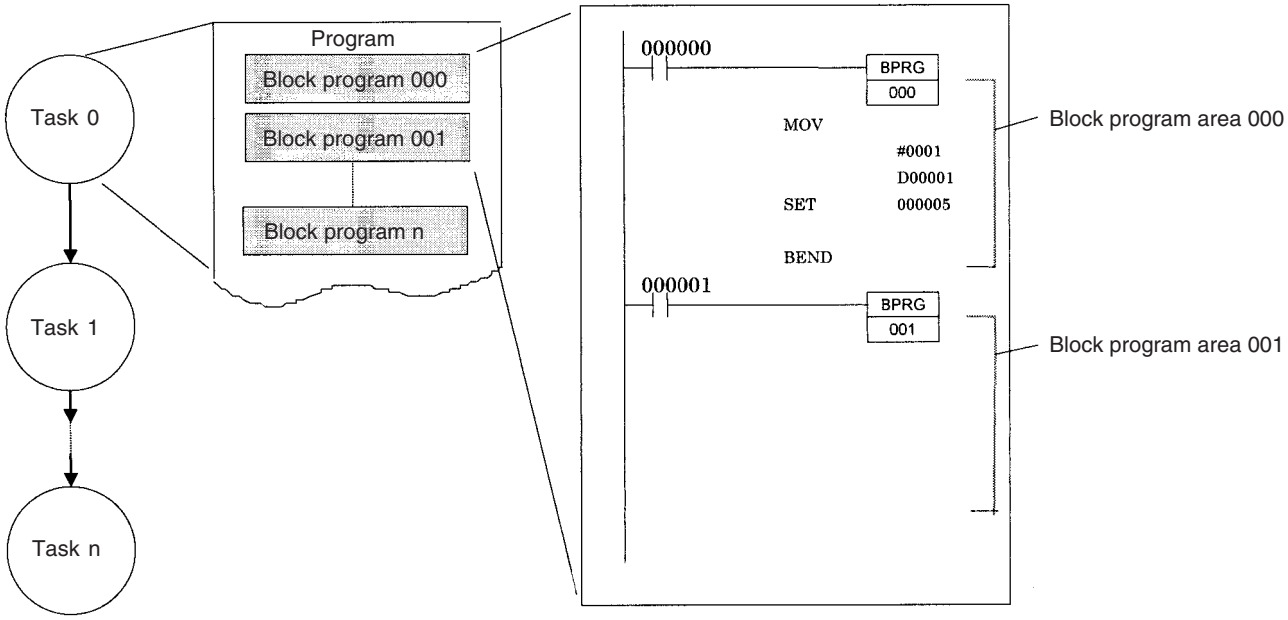
3. Generally, use an overall control task to control the READY/Standby status of the other tasks.
4. Allocate the lowest numbers to tasks with the highest priority.
Example: Allocate a lower number to the control task than to processing tasks.
5. Allocate lower numbers to high-priority interrupt tasks.
6. A task in READY status will be executed in subsequent cycles as long as the task itself or another task does not shift it to Standby status. Be sure to insert a TKOF(821) (TASK OFF) instruction for other tasks if processing is to be branched between tasks.
7. Use the Initial Task Execution Flag (A20015) or the Task Start Flag (A20014) in the execution condition to execution instructions to initialize tasks. The Initial Task Execution Flag will be ON during the first execution of each task. The Task Start Flag each time a task enters READY status.

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

Relationship of Tasks to Block Programs

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

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



4-2-5 Global Subroutines

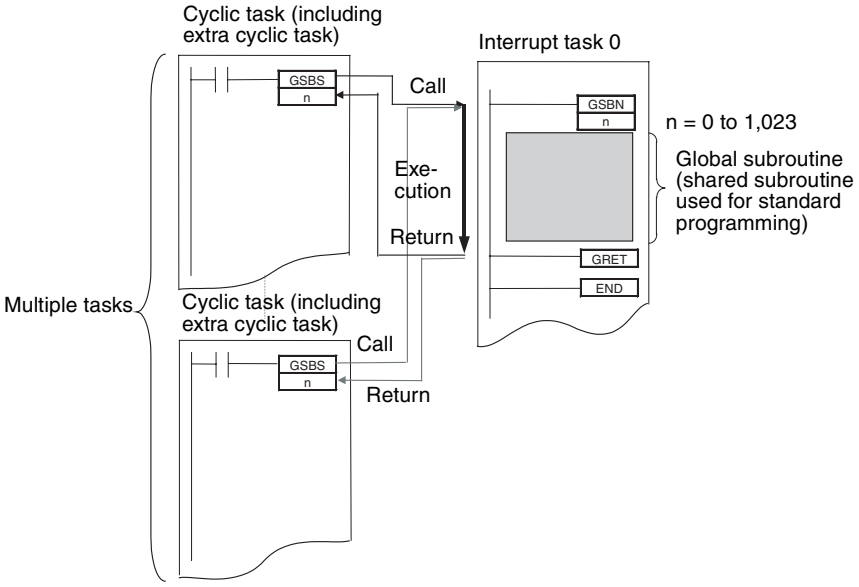
Global subroutines can be called from more than one task. With regular subroutine instructions, it is not possible to call a subroutine in one task from a different task.

With the CS1 or CJ1 CPU Units, global subroutines cannot be used, i.e., a subroutine in one task cannot be called from other tasks. With the CS1-H, CJ1-H, CJ1M, or CS1D CPU Units, however, global subroutines can be created in interrupt task number 0, and these subroutines can be called from any cyclic task (including extra cyclic tasks).

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support this function.

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

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



4-3 Interrupt Tasks

4-3-1 Types of Interrupt Tasks

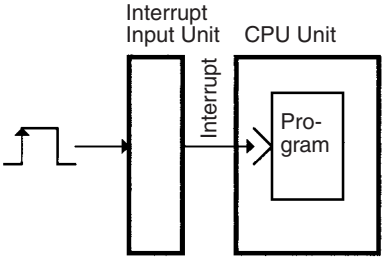
Interrupt tasks can be executed at any time in the cycle if any of the following conditions are in effect.

The built-in interrupt inputs and high-speed counter inputs on a CJ1M CPU Unit can be used to activate interrupt tasks. Refer to the *CJ Series Built-in I/O Operation Manual* for details.

Note The CS1D CPU Units for Duplex-CPU Systems do not support interrupts. With the CS1D CPU Units, interrupt tasks can be used only as extra cyclic tasks.

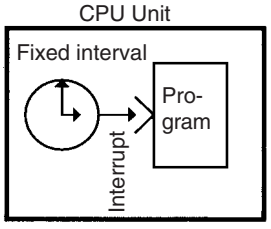
I/O Interrupts (CS Series Only)

The I/O interrupt task will be executed when input to the Interrupt Input Unit is ON.



Scheduled Interrupts

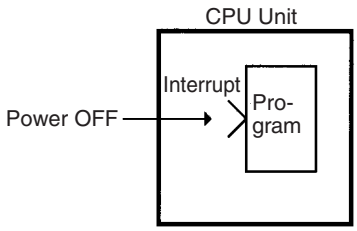
A scheduled interrupt task will be executed at fixed intervals.



Power OFF Interrupt

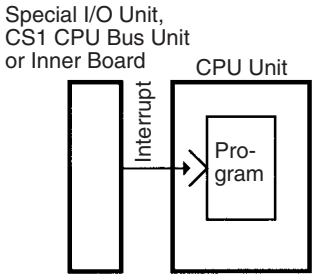
The power OFF interrupt task will be executed when power is turned OFF.

Note The execution time for the power OFF task must be less than 10 ms – (Power OFF delay detection time).



External Interrupts (CS Series Only)

An external interrupt task will be executed when an interrupt is requested by an Special I/O Unit, CPU Bus Unit, or Inner Board (CS Series only). The Special I/O Unit or CJ Bus Unit, however, must be on the CPU Rack to request execution of an external interrupt task.



List of Interrupt Tasks

Type	Task No.	Execution condition	Setting procedure	Number of interrupts	Application examples
I/O Interrupts 00 to 31	100 to 131	Input from the Interrupt Input Unit ON on the CPU Rack (See note 1.)	Use the MSKS (SET INTERRUPT MASK) instruction to assign inputs from Interrupt Input Units on the CPU Rack.	32 points	Increasing response speed to specific inputs
Scheduled Interrupts 0 and 1	2 and 3	Scheduled (fixed intervals)	Use the MSKS (SET INTERRUPT MASK) instruction to set the interrupt interval. See Scheduled Interrupt Time Units in PLC Setup.	2 points	Monitoring operating status at fixed intervals
Power OFF Interrupt	1	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.	1 point	Executing emergency processing when power shuts OFF.
External Interrupts 0 to 255	0 to 255	When requested by an Special I/O Unit or CPU Bus Unit on the CPU Rack or by an Inner Board (CS Series only) (See note 2.)	None (always valid)	256 points	Performing processing required by Special I/O Units, CPU Bus Units, and the Inner Board.

Note

1. The Interrupt Input Unit must be mounted to the CPU Rack. For CJ1-H CPU Units, the Unit must be connected as one of the five Units next to the CPU Unit (slots 0 to 4). For CJ1M CPU Units, the Unit must be connected as one of the three Units next to the CPU Unit (slots 0 to 2). I/O Interrupt Units mounted elsewhere cannot be used to request execution of I/O interrupt tasks
2. The Special I/O Unit or CPU Bus Unit must be mounted to the CPU Rack. For CJ1-H CPU Units, the Unit must be connected as one of the five Units

next to the CPU Unit (slots 0 to 4). For CJ1M CPU Units, the Unit must be connected as one of the three Units next to the CPU Unit (slots 0 to 2). Units mounted elsewhere cannot be used to generate external interrupts.

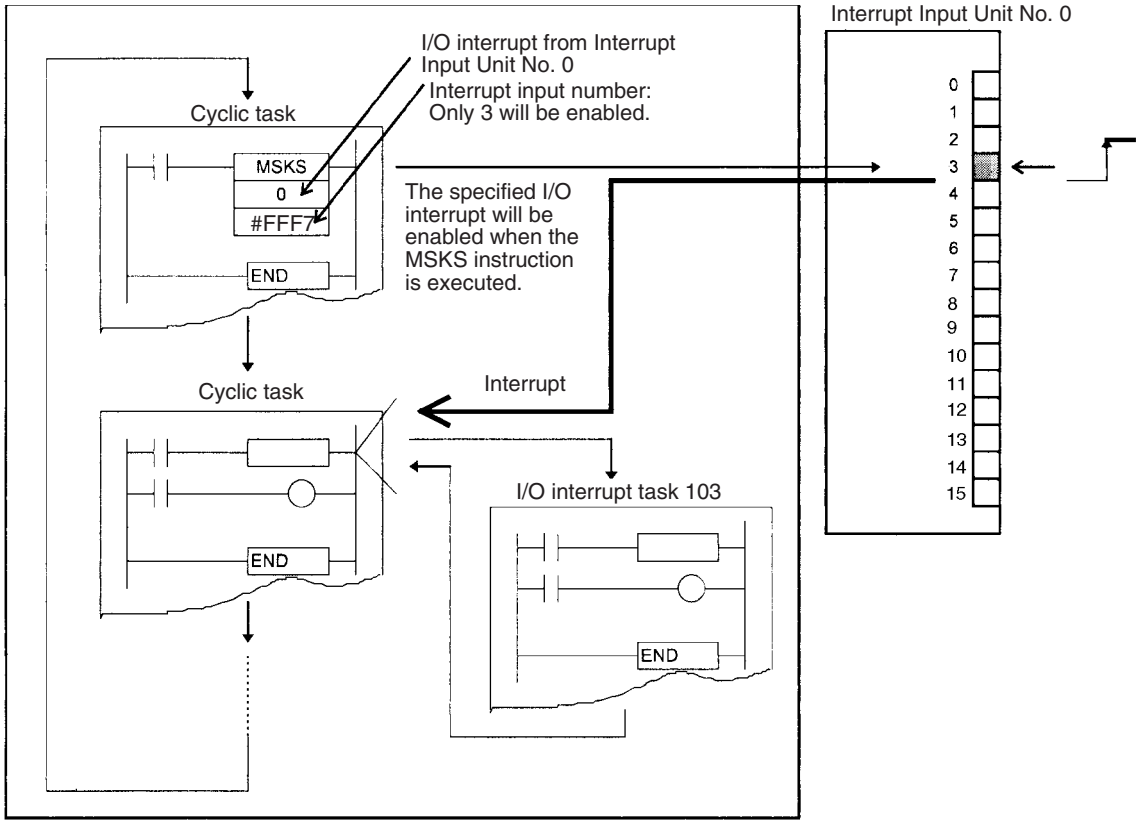
3. CJ1 CPU Units for Duplex-CPU Systems do not support I/O interrupt and external interrupt tasks.
4. The CS1D CPU Units for Duplex-CPU Systems do not support interrupts. With the CS1D CPU Units, interrupt tasks can be used only as extra cyclic tasks, i.e., no other type of interrupt task can be used.

I/O Interrupt Tasks: Tasks 100 to 131

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

Example: The following example shows execution I/O interrupt task 103 when interrupt input No. 3 of Interrupt Input Unit No. 0 (the leftmost of the two Units 0 and 1) is ON.

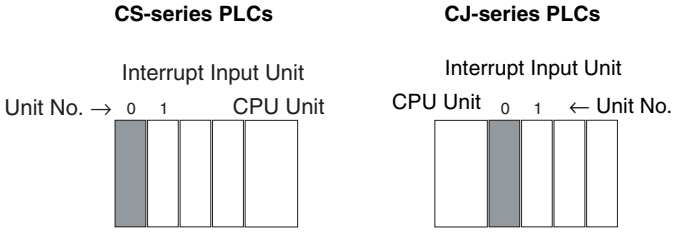
Note Do not enable unneeded I/O interrupt tasks. If the interrupt input is triggered by noise and there isn't a corresponding interrupt task, a fatal error (task error) will cause the program to stop.



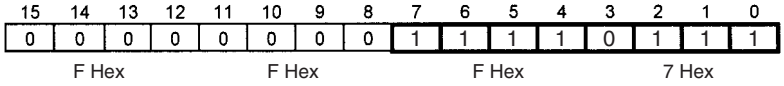
Interrupt Input Unit Numbers, Input Numbers, and I/O Interrupt Task Numbers

Interrupt Input Unit No. (See note.)	Input No.	I/O interrupt task
0	0 to 15	**100 to 115**
1	0 to 15	**116 to 131**

Note For CS-series PLCs, Interrupt Input Unit numbers are in order from 0 to 1 starting on the left side of the CPU Rack. For CJ-series PLCs, Interrupt Input Unit numbers are in order from 0 to 1 starting from the CPU Unit.



Operand S (the Second Operand) of MSKS: The bits of FFF7 Hex correspond to the interrupt inputs of the Interrupt Input Unit. Interrupt input numbers 0 to 15 correspond to bits 0 to 15.



Scheduled Interrupt Tasks: Tasks 2 and 3

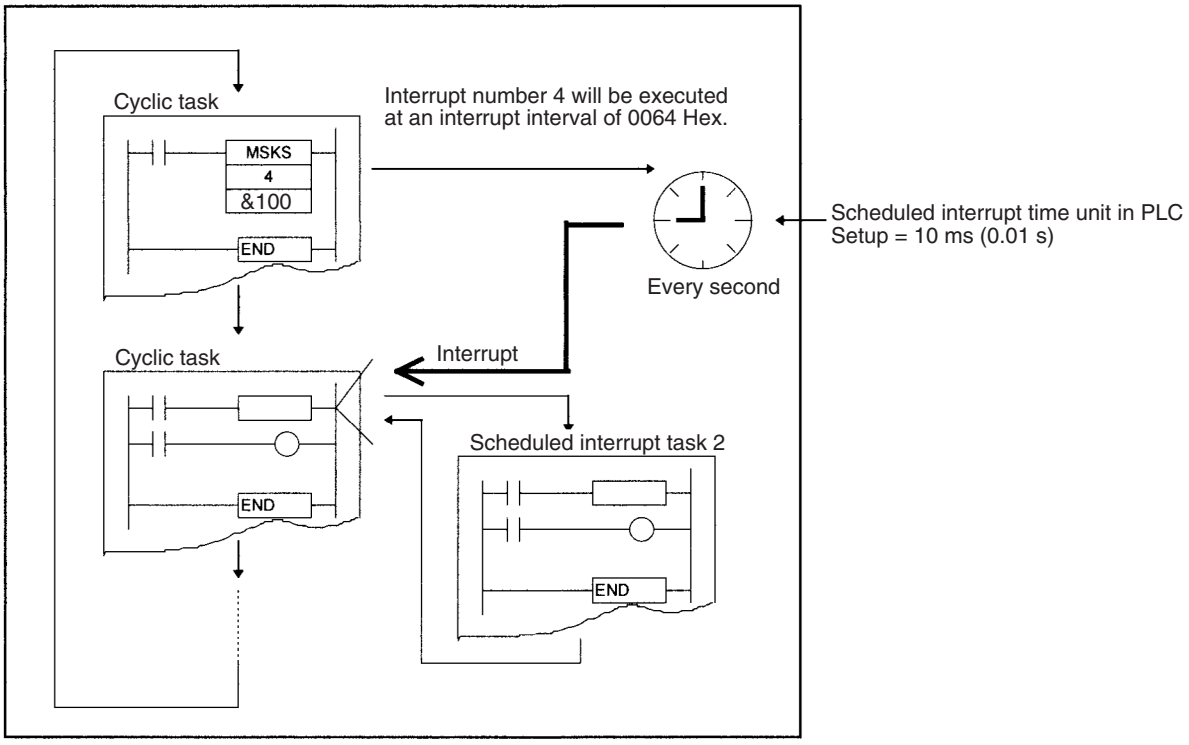
Scheduled interrupt tasks are disabled in the default PLC Setup at the start of cyclic task execution. Perform the following steps to enable scheduled interrupt tasks.

1,2,3...

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

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

Example: The following examples shows executed scheduled interrupt task 2 every second.



Interrupt Numbers and Scheduled Interrupt Task Number

Interrupt No.	Scheduled interrupt task
4	2
5	3

PLC Setup Settings

Programming Console address	Name	Description	Settings	Default setting
Word +195, bits 0 to 3	Scheduled interrupt time units	Sets time unit for scheduled interrupts to execute interrupt tasks at fixed intervals.	00 Hex: 10 ms 01 Hex: 1.0 ms 02 Hex: 0.1 ms (CJ1-H-R/CJ1M CPU Units only)	00 Hex

Power OFF Interrupt Task: Task 1

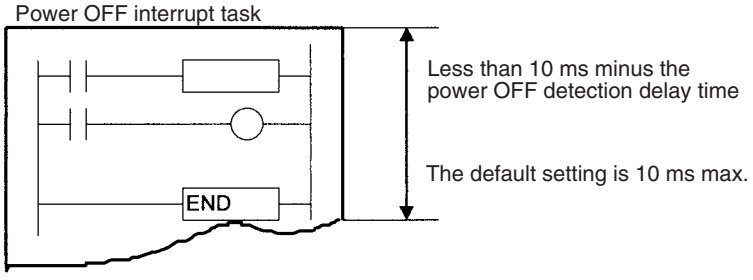
The power OFF interrupt task is disabled in the default PLC Setup at the start of cyclic task execution.

The power OFF interrupt task can be enabled in the PLC Setup.

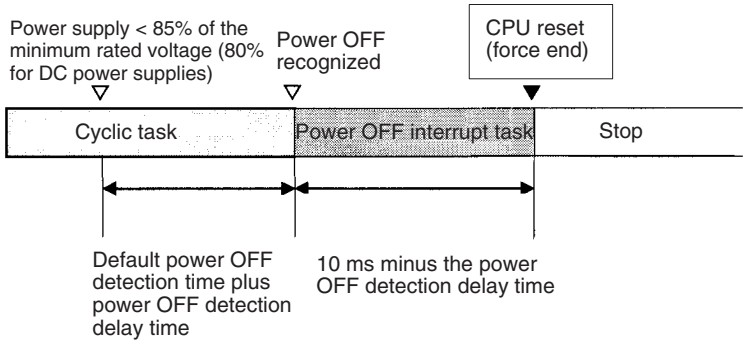
In the default PLC Setup, the power OFF interrupt task will be stopped after 10 ms. The power OFF interrupt task must be executed in less than 10 ms.

If a power OFF detection delay time is set in the PLC Setup, the power OFF interrupt task will be stopped after 10 ms minus the power OFF detection delay time setting in the PLC Setup. In this case, the power OFF interrupt task must execute in less than 10 ms minus the power OFF detection delay time set in the PLC Setup.

Example: If the power OFF detection delay time is set to 4 ms in PLC Setup, then execution time must be less than 10 minus 4 ms, or 6 ms.

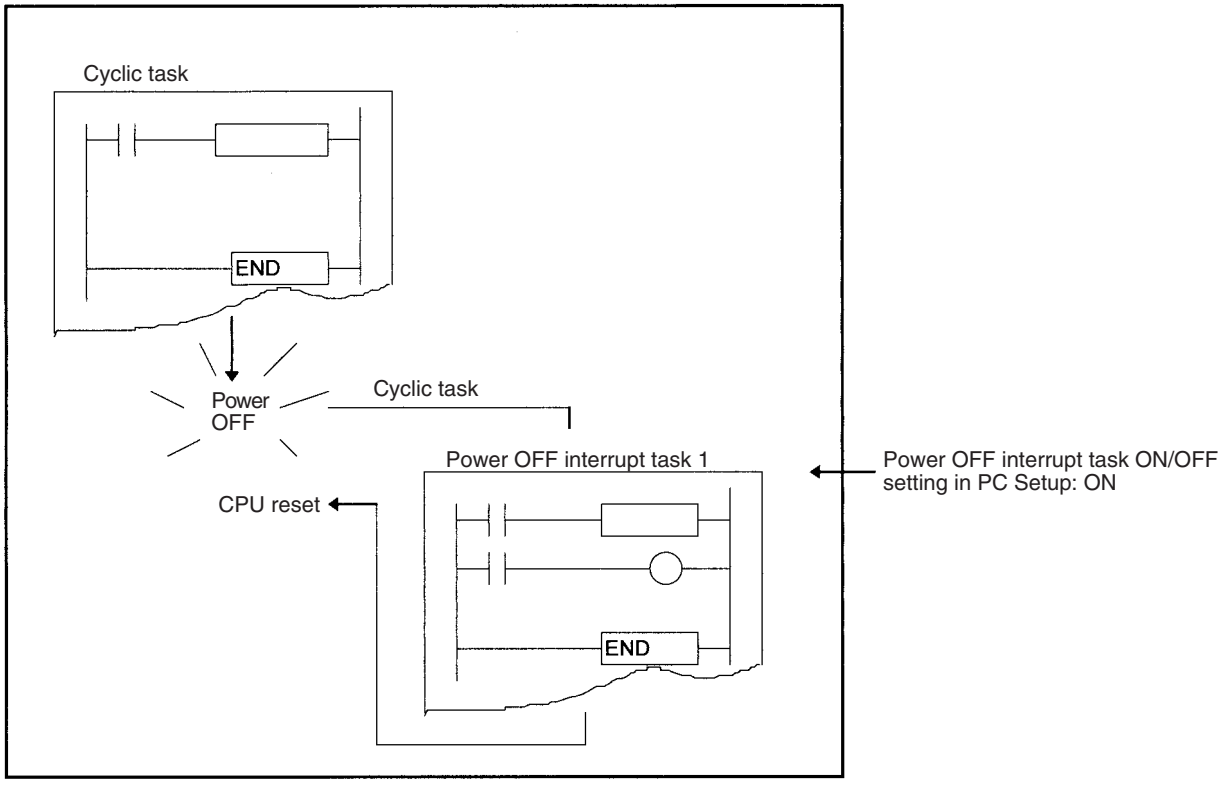


Note A power OFF condition is recognized when the power supply falls below 85% of the minimum rated voltage (80% for DC power supplies), and the time it takes before the power OFF interrupt task actually executes is the default power OFF detection time (10 to 25 ms for AC power supplies and 2 to 5 ms for DC power supplies) plus the power OFF detection delay time in the PLC Setup (0 to 10 ms). Cyclic tasks will be executed for this amount of time.



Note Be sure that the power OFF interrupt task can be executed in less than 10 ms minus the power OFF detection delay time set in the PLC Setup. Any remaining instructions will not be executed after this time has elapsed. The power OFF interrupt task will not be executed if power is interrupted during online editing. In addition to the instructions that cannot be used in any interrupt task (refer to the *Instructions Reference Manual* for details), the following instructions cannot be used in the power OFF interrupt task: READ DATA FILE: FREAD(700), WRITE DATA FILE: FWRIT(701), WRITE TEXT FILE(TWRIT(704)), NETWORK SEND: SEND(090), NETWORK RECEIVE: RECV(098), DELIVER COMMAND: CMND(490), TRANSMIT: TXD(236), RECEIVE: RXD(235), and PROTOCOL MACRO: PMCR(260), EXPLICIT MESSAGE SEND: EXPLT(270), EXPLICIT GET ATTRIBUTE: EGATR(271), EXPLICIT SET ATTRIBUTE: ESATR(272), EXPLICIT WORD READ: ECHRD(273), EXPLICIT WORD WRITE: ECHWR(274), TRANSMIT VIA SERIAL COMMUNICATIONS UNIT: TXDU(256), and RECEIVE VIA SERIAL COMMUNICATIONS UNIT: RXDU(255).

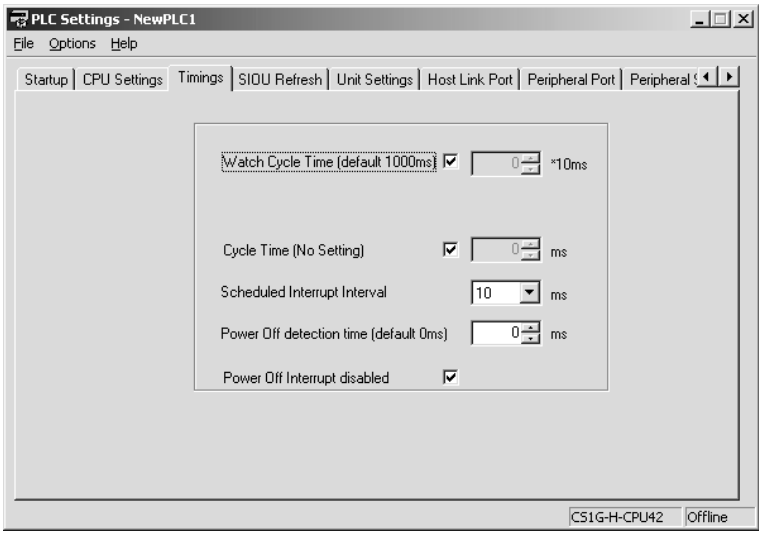
Power OFF Interrupt Task Execution



PLC Setup Settings for Power OFF Interrupt Task (Task Number: 1)

PLC Setup

When using the CX-Programmer, make the settings on the Timings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address	Name	Description	Settings	Default setting
Word +225, bit 15	Power OFF INTERRUPT TASK	If bit 15 of +225 is ON, then a power OFF interrupt task will start if power turns OFF.	0: OFF, 1: ON	0
Word +225, bits 0 to 7	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 5 ms for DC power supplies) expires.	00 to 0A Hex: 0 to 10 ms (1-ms units)	00 Hex

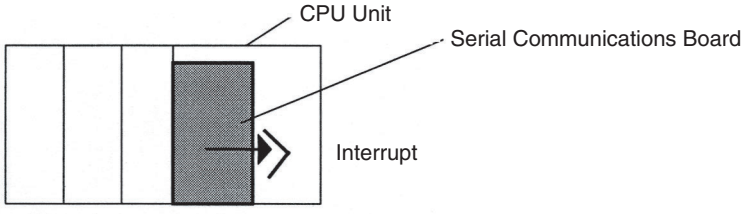
External Interrupt Tasks: Tasks 0 to 255

External interrupt tasks can be received at any time.

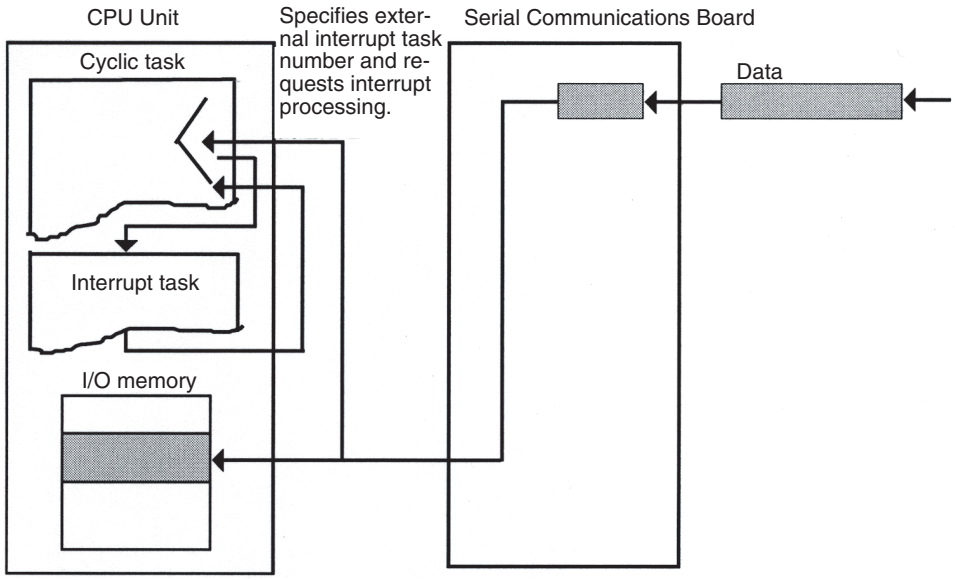
Interrupt processing is performed at the CPU Unit in PLCs containing an Inner Board (CS Series only), Special I/O Units, or CPU Bus Units. Settings don't have to be made at the CPU Unit unless the program contains an external interrupt task for a particular task number.

External interrupts are not supported by CJ1 CPU Units.

Example: The following example shows an external interrupt generated from a CS1W-SCB□1 Serial Communications Board.



When the Serial Communications Board's response notification method is set for interrupt notification (fixed number) or interrupt notification (reception case number), the Board will request execution of an external interrupt task in the CPU Unit after it receives data from its serial port and writes that data into the CPU Unit's I/O memory.



Note

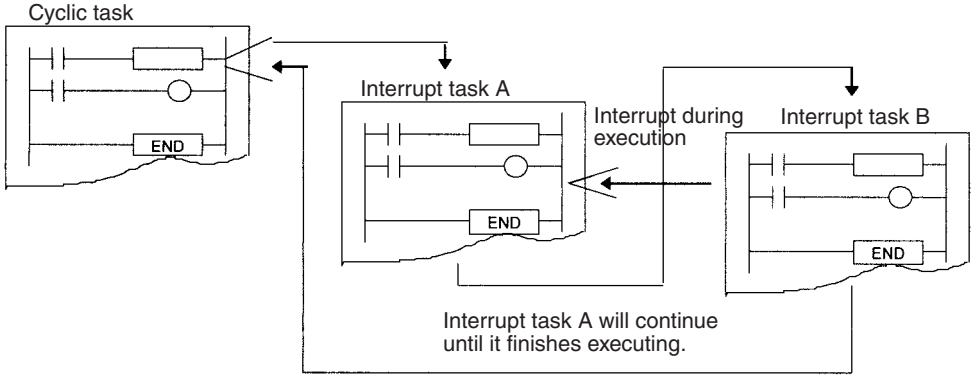
1. When the response notification method is set for interrupt notification (fixed number), the Board requests execution of the interrupt task with the preset task number.
2. When the response notification method is set for interrupt notification (reception case number), the external interrupt task number is calculated with the specified formula and the Board requests execution of the interrupt task with that task number.
3. If an external interrupt task (0 to 255) has the same number as a power OFF task (task 1), scheduled interrupt task (task 2 or 3), or I/O interrupt task (100 to 131), the interrupt task will be executed for either interrupt condition (external interrupt or the other interrupt condition). As a rule, task numbers should not be duplicated.

4-3-2 Interrupt Task Priority

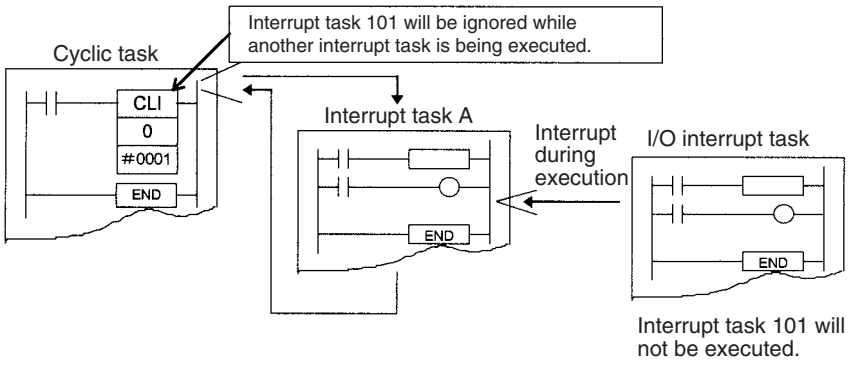
Execution of another interrupt task will be ended to allow the power OFF interrupt task to execute. The CPU will reset but the terminated interrupt task will not be executed following execution of the power OFF interrupt task.

Interrupt during Interrupt Task Execution

If an interrupt occurs while another interrupt task is being executed, the task for the interrupt will not be executed until the original interrupt finishes executing.



Note If you do not want a specific I/O interrupt task number to be saved and executed for a CS-series CPU Unit when it occurs while another interrupt task is being executed, execute the CLI (CLEAR INTERRUPT) instruction from the other interrupt task to CLEAR the interrupt number saved internally. Scheduled interrupts and external interrupts cannot be cancelled.



Multiple Interrupts Occurring Simultaneously

Interrupt tasks other than power OFF interrupt tasks will be executed in the following order of priority whenever multiple interrupts occur simultaneously.

I/O interrupt tasks (CS Series only) > external interrupt tasks (CS Series only) > scheduled interrupt tasks

Each of the various types of interrupt task will be executed in order starting from the lowest number if more than one occurs.

Note Only one interrupt will be recorded in memory for each interrupt task and an interrupt will not be recorded for an interrupt that is already being executed. Because of the low order of priority of scheduled interrupts and because that only one interrupt is recorded at a time, it is possible for a scheduled interrupt to be skipped.

4-3-3 Interrupt Task Flags and Words

Maximum Interrupt Task Processing Time (A440)

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

Interrupt Task with Maximum Processing Time (A441)

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

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.

Interrupt Task Error Flag (Nonfatal Error) (A40213)

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

Interrupt Task Error Flag (A42615)/Task Number Generating the Interrupt Task Error (A42600 to 42611)

If A40213 turns ON, then the following data will be stored in A42615 and A42600 to A42611.

A40213	Interrupt Task Error Description	A42615	A42600 to 42611
Interrupt Task Error (when Interrupt Task Error Detection is enabled in the PLC Setup)	If an interrupt task executes for more than 10 ms during C200H Special I/O Unit or SYSMAC BUS Remote I/O refresh (CS Series only).	OFF	The interrupt task number will be stored in 12 bits of binary data (interrupt task 0 to 255: 000 to OFF Hex).
	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 a Special I/O Unit's 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.	ON	The unit number of the Special I/O Unit being refreshed will be stored in 12 bits of binary data (unit No. 0 to 95: 000 to 05F Hex).

Task Number when Program Stopped (A294)

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

Type	A294
Interrupt task	8000 to 80FF Hex (corresponds to interrupt task No. 0 to 255)
Cyclic task	0000 to 001F Hex (corresponds to task No. 0 to 31)

4-3-4 Application Precautions

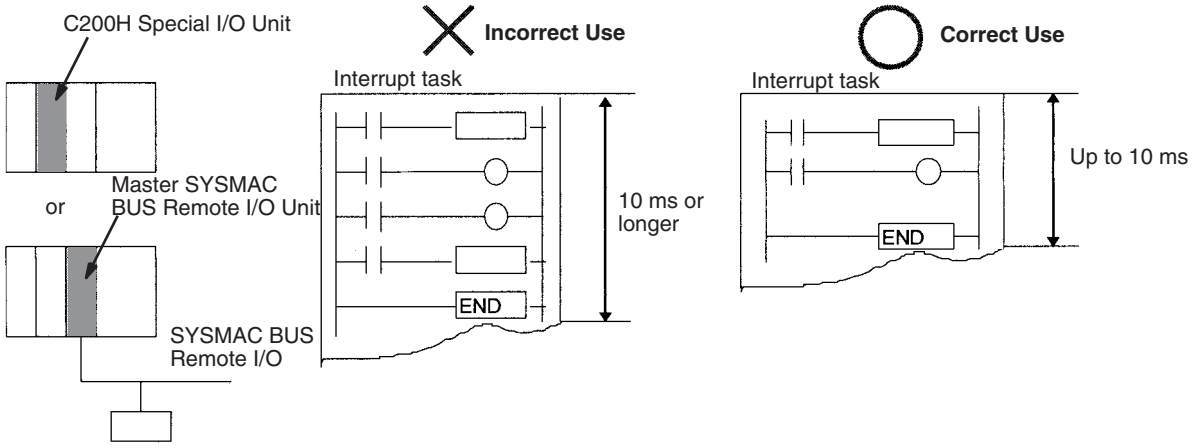
Long Execution Times with C200H Special I/O Units or SYSMAC BUS (CS Series Only)

Be sure all interrupt tasks (I/O, scheduled, power OFF, and external interrupt tasks) execute within 10 ms when using C200H Special I/O Units or SYSMAC BUS Remote I/O.

If an interrupt task executes for more than 10 ms during C200H Special I/O Unit or SYSMAC BUS remote I/O refreshing, an interrupt task error will occur, A40206 (Special I/O Unit Error Flag) will turn ON, and I/O refreshing will be stopped for Special I/O Units. The CPU Unit, however, will continue to operate.

If Interrupt Task Error Detection is turned ON in the PLC Setup, A40213 (Interrupt Task Error Flag) will turn ON when an interrupt task error occurs, and the

offending interrupt task number will be stored in A426 (Interrupt Task Error, Task Number). The CPU Unit however will continue to operate.



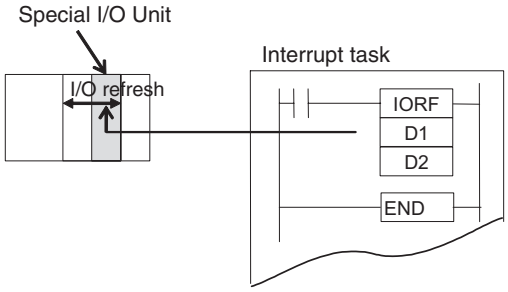
Executing IORF(097), FIORF(225) (CJ1-H-R Only), IORD(222), or IOWR(223) for a Special I/O Unit

If a Special I/O Unit is being used and IORF(097), FIORF(225) (CJ1-H-R only), IORD(222), or IOWR(223) will be executed from an interrupt task, always disable cyclic refreshing for that Special I/O Unit in the PLC Setup.

If a Special I/O Unit is being refreshed by cyclic refreshing or an I/O refreshing instruction, an interrupt task error will occur if you try to refresh the same Special I/O Unit with an IORF(097), FIORF(225) (CJ1-H-R only) in an interrupt task or if an attempt is made to read/write data for the same Special I/O Unit with an IORD(222) or IOWR(223) instruction. In this case, the IORF(097), FIORF(225) (CJ1-H-R only), IORD(222), or IOWR(223) instruction will not be executed, but the Error Flag will not be turned ON. Cyclic refreshing will be performed normally.

If Interrupt Task Error Detection is enabled in the PLC Setup when an interrupt task error occurs, A40213 (Interrupt Task Error Flag) will turn ON and the unit number of the affected Special I/O Unit will be stored in A426 (Interrupt Task Error, Task Number).

IORF(097)



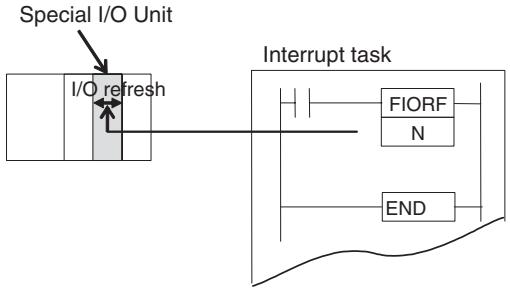
Incorrect Use

Cyclic refreshing is enabled for a Special I/O Unit in the PLC Setup (the default setting), and IORF(097) is executed in an interrupt task to refresh that Special I/O Unit's I/O words.

Correct Use

Cyclic refreshing is disabled for a Special I/O Unit in the PLC Setup, and IORF(097) is executed in an interrupt task to refresh that Special I/O Unit's I/O words.

FIORF(225) (CJ1-H-R CPU Units Only)



Incorrect Use

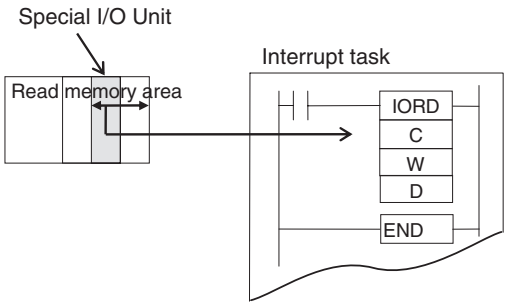
Cyclic refreshing is enabled for a Special I/O Unit in the PLC Setup (the default setting), and FIORF(225) (CJ1H-H-R only) is executed in an interrupt task to refresh that Special I/O Unit's allocated words.

Correct Use

Cyclic refreshing is disabled for a Special I/O Unit in the PLC Setup, and FIORF(225) (CJ1H-H-R only) is executed in an interrupt task to refresh that Special I/O Unit's allocated words.

Note The leftmost bits of A426 (Interrupt Task Error, Task Number) can be used to determine which of the above interrupt task errors occurred. (Bit 15: 10 ms or higher execution error if 0, multiple refresh error if 1)

IORD(222)



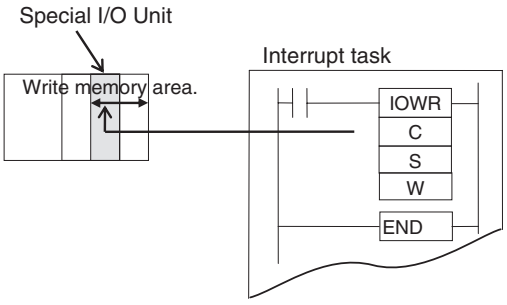
Incorrect Use ✕

Cyclic refreshing is enabled for a Special I/O Unit in the PLC Setup (the default setting), and IORD(222) is executed in an interrupt task to read that Special I/O Unit's allocated words.

Correct Use ○

Cyclic refreshing is disabled for a Special I/O Unit in the PLC Setup, and IORD(222) is executed in an interrupt task to read that Special I/O Unit's allocated words.

IOWR(223)



Incorrect Use ✕

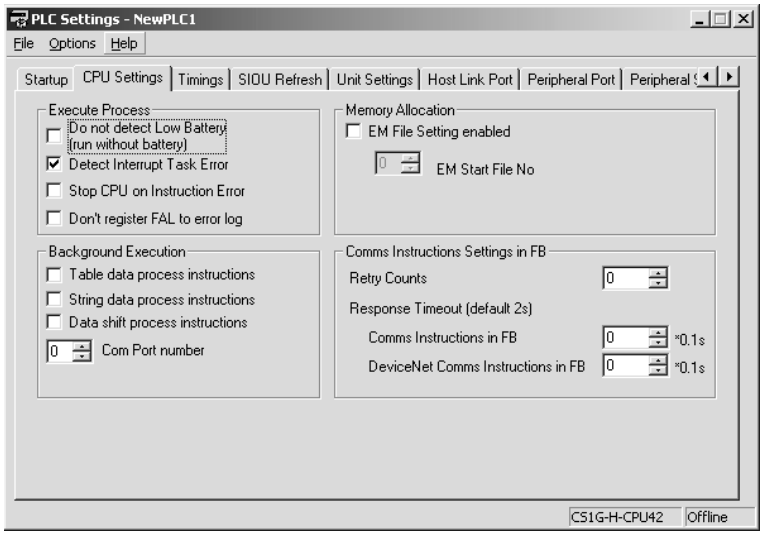
Cyclic refreshing is enabled for a Special I/O Unit in the PLC Setup (the default setting), and IOWR(223) is executed in an interrupt task to write that Special I/O Unit's allocated words.

Correct Use ○

Cyclic refreshing is disabled for a Special I/O Unit in the PLC Setup, and IOWR(223) is executed in an interrupt task to write that Special I/O Unit's allocated words.

PLC Setup

When using the CX-Programmer, make the settings on the CPU Settings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address	Name	Description	Settings	Default setting
Word +128, bit 14	Interrupt Task Error Detection	Specifies whether or not to detect interrupt task errors. The Interrupt Task Error Flag (A40213) will be function when detection is enabled.	0: Detection enabled, 1: Detection disabled	0

Related Auxiliary Area Flags/Words

Name	Address	Description
Interrupt Task Error Flag	A40213	Turns ON if an interrupt task executes for more than 10 ms during C200H Special I/O Unit or SYSMAC BUS Remote I/O refresh, but the CPU Unit will continue running. The ERR/ALM LED will light on the front panel (CS Series only). Also turns ON if Interrupt Task Error Detection is enabled for a Special I/O Unit in the PLC Setup, and one of the following conditions occurs for that 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 a Special I/O Unit's 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	A426	Contains the interrupt task number or the number of the Special I/O Unit being refreshed. (Bit 15 will be OFF when execution of an interrupt task requires 10 ms or longer and ON when duplicated Special I/O Unit refreshing has occurred.)

Disabling Interrupts

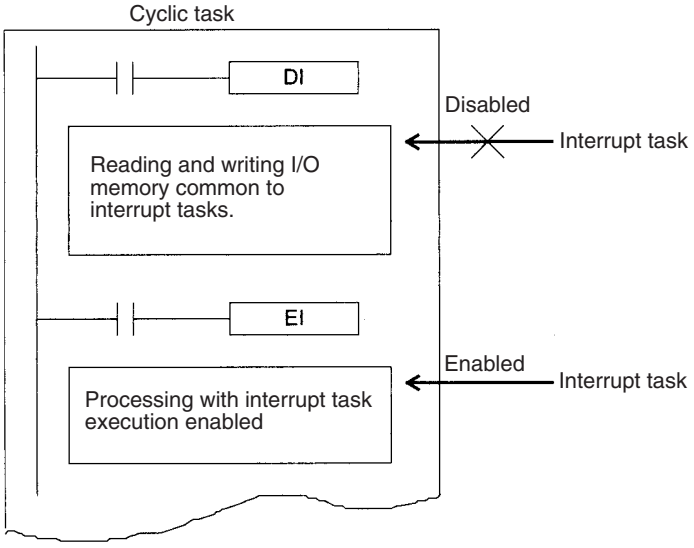
Processing will be interrupted and the interrupt task will be executed in the following instances.

- While an instruction is being executed
- During Basic I/O Unit, CPU Bus Unit, Inner Board (CS Series only), or SYSMAC BUS remote I/O (CS Series only) refreshing
- During HOST LINK servicing

Data Concurrency between Cyclic and Interrupt Tasks

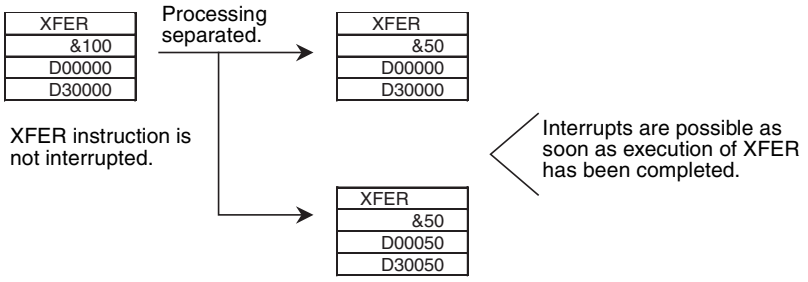
Data may not be concurrent if a cyclic (including extra cyclic tasks) and an interrupt task are reading and writing the same I/O memory addresses. Use the following procedure to disable interrupts during memory access by cyclic task instructions.

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



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

Note Execution of the BIT COUNTER (BCNT), BLOCK SET (BSET), and BLOCK TRANSFER (XFER) instructions will not be interrupted for execution of interrupt task, i.e., execution of the instruction will be completed before the interrupt task is executed, delaying the response of the interrupt. To prevent this, separate data processing for these instructions into more than one instructions, as shown below for XFER.



4-4 Programming Device Operations for Tasks

4-4-1 Using Multiple Cyclic Tasks

Use the CX-Programmer to create more than one cyclic task (including extra cyclic tasks). A Programming Console cannot be used to create new cyclic tasks. Be sure to use a CX-Programmer to allocate the task type and task number for programs that are created.

- Multiple cyclic tasks created and transferred to a CPU Unit from the CX-Programmer can be monitored or edited from a Programming Console.
- The Programming Console can be used to create one cyclic task and one or more specific interrupt tasks simply by using the Programming Console's All Clear function and specifying Interrupt Tasks. Only interrupt tasks 1 (power OFF interrupt), 2 and 3 (scheduled interrupts), and 100 through 131 (I/O interrupts) can be created with a Programming Console. With a CJ1M CPU Unit, however, interrupt tasks 140 through 143 (for built-in inputs) can also be created. Cyclic task 0 will start when PLC operation is started.

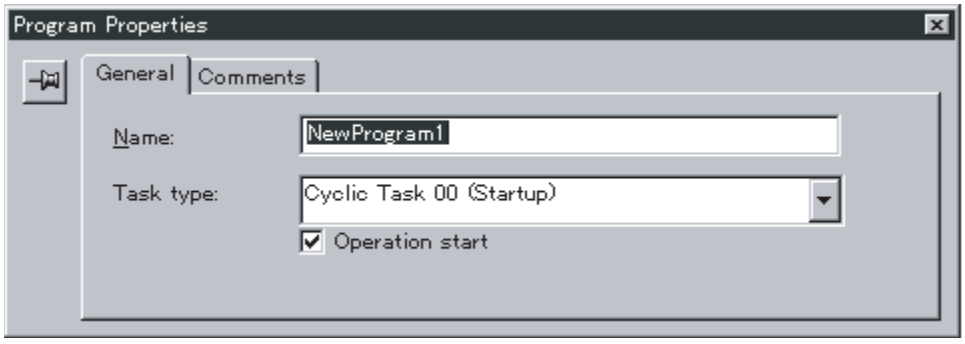
4-4-2 Programming Device Operations

CX-Programmer

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

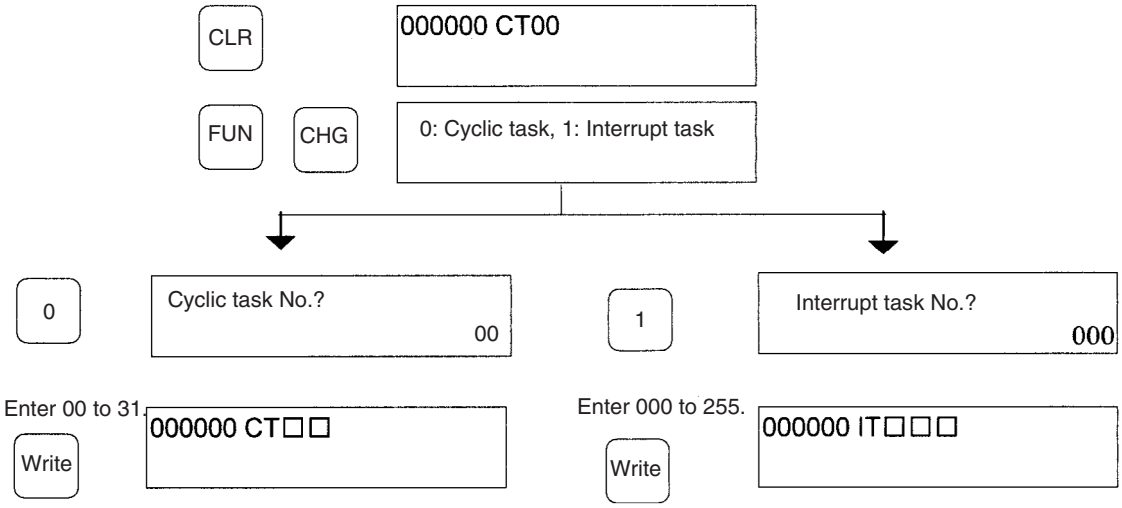
1,2,3...

1. Select ***View/Properties,*** or click the right button and select **Properties** on the popup menu, to display the program that will be allocated a task.
2. Select the **General** tab, and select the **Task Type** and **Task No.** For the cyclic task, click the check box for **Operation start** to turn it ON.



Programming Console

A task is handled as the entire program on the Programming Console. Access and edit a program with a Programming Console by specifying CT00 to CT31 for a cyclic task or IT001 to IT255 for an interrupt task.



Note

1. A Programming Console cannot create new cyclic tasks.
2. The CJ-series CPU Units do not currently support I/O or external interrupt tasks. Only IT001 to IT003 can be specified.

SECTION 5
File Memory Functions

This section describes the functions used to manipulate file memory.

5-1 File Memory

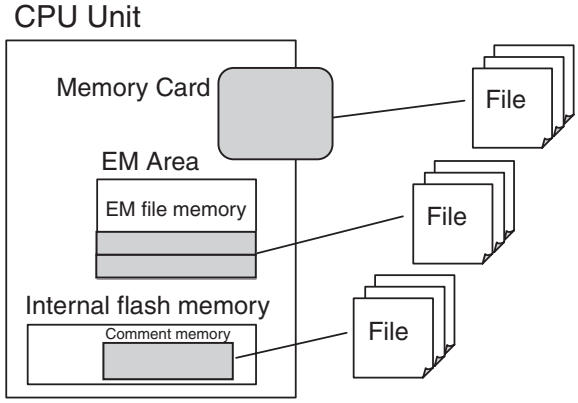
The CS/CJ Series support file memory.

1,2,3... 1. Memory Cards
2. A specified range in the EM Area called EM file memory

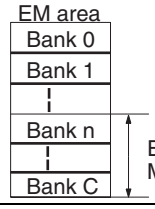
   Note CJ1M CPU Units do not have an EM Area, so EM file memory cannot be used.
3. Comment memory (in CPU Unit's internal flash memory)

   Note File memory can be used in CPU Units with unit version 3.0 and later.

For details on selecting file memory, refer to *5-1-4 Applications* on page 215.



5-1-1 Types of File Memory

Category	Memory media	Storable file types
Memory Card 	Flash memory	Program files Parameter files Data files Symbol table files Comment files Program index files Unit and Board backup files
EM File Memory 	EM area (RAM)	Program files Parameter files Data files Symbol table files Comment files Program index files User's files
Comment memory	CPU Unit's internal flash memory	Symbol table files Comment files Program index files

Note
1. Initialize the Memory Card or EM File Memory before using it for the first time. Refer to *5-3 Using File Memory* for details on initialization.
2. Refer to *5-2 Manipulating Files* for details on installing and removing Memory Cards.
3. The HMC-AP001 Memory Card Adapter can be used to mount a Memory Card in the PLC card slot of a personal computer to use the Memory Card as a storage device.

Memory Card Precautions

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)* in the *CS/CJ Series Instructions Reference* 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.

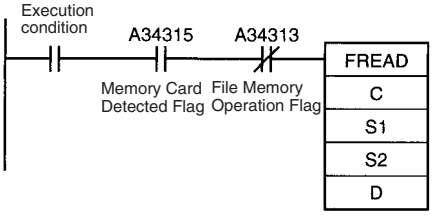
Memory Card Access Precautions

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.



5-1-2 Files Stored in File Memory

The following types of files can be stored in file memory.

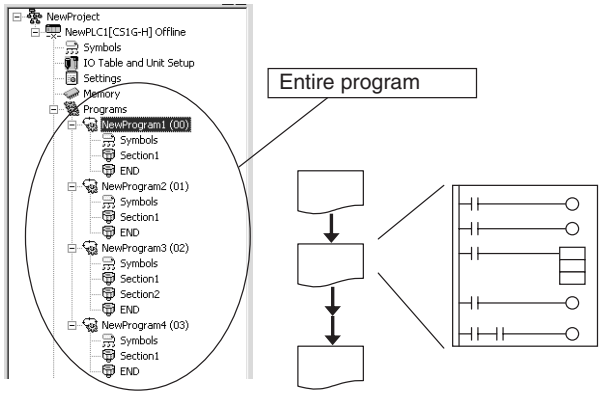
1,2,3...

1. Program files
2. Parameter files
3. Data files
4. Symbol table files
5. Comment files
6. Program index files
7. Unit and Board backup files

File Types

1. Program File

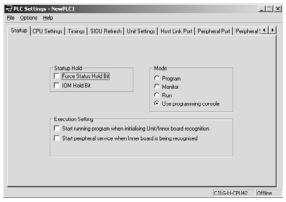
The program file contains the CPU Unit's user program (the programs in the cyclic tasks and interrupt tasks). This file also contains each program's properties



2. Parameter File

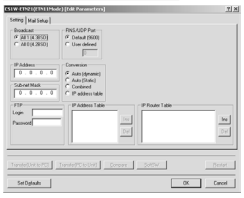
The parameter file contains the CPU Unit's internal Parameter Area data. The Parameter Area data includes the PLC Setup, registered I/O tables, CPU Bus Unit settings (including the data link parameters), and routing tables.

PLC Setup

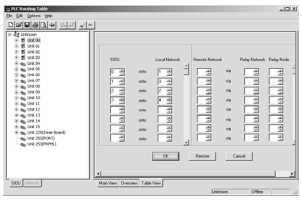


Registered I/O tables

CPU Bus Unit Settings

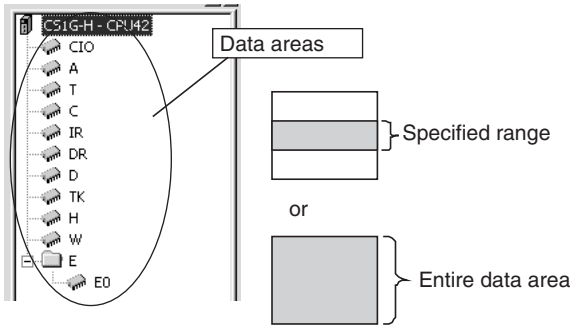


Routing tables



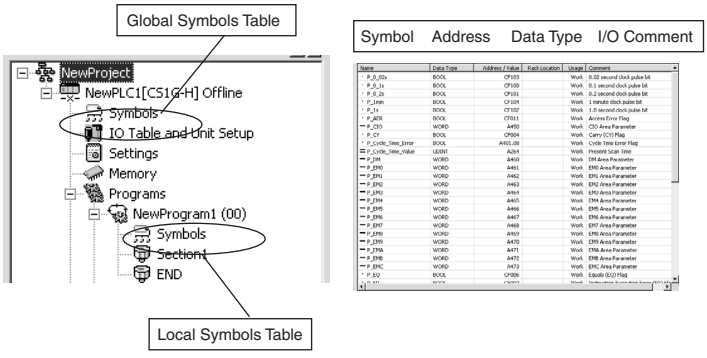
3. Data File

The data file contains the data of one I/O memory data area, in word (16-bit) units. It is possible to store all of the data in the data area or just a specified range of addresses. Any one of the following 6 data areas can be stored: the CIO, Holding, Work, Auxiliary, DM, or EM Area.



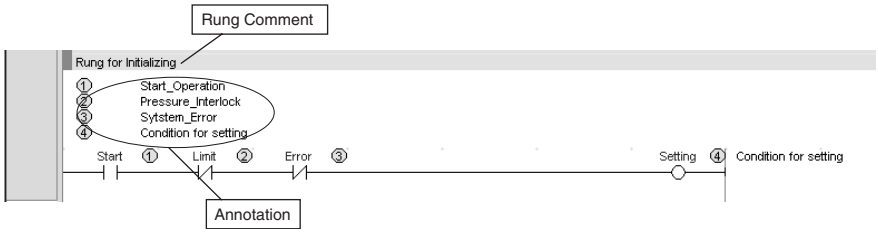
4. Symbols Table File

The symbol table file contains the symbol table information used in the CX-Programmer as well as the automatically allocated PLC addresses.



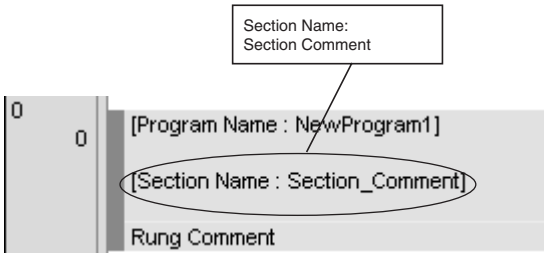
5. Comment File

The comment file contains the comment information used in the CX-Programmer.



6. Program Index File

The program index file contains the section information used in the CX-Programmer.



7. Unit/Board Backup File

The Unit/Board backup file contains the internal data of a PLC Unit or Board, which is used by the simple backup function. These files are created when the simple backup operation is executed. Internal data is stored for each Unit.

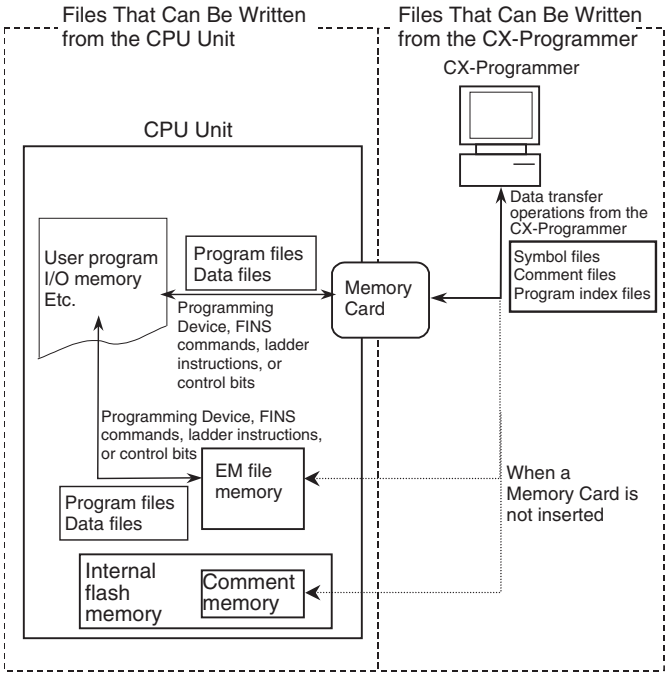
Example:

DeviceNet Units: Device parameters

Serial Communications Units: Protocol macro data

File Types and File Operations

File type	Related file memory operations
• Program files • Data files • Parameter files	• Programming Device (CX-Programmer or Programming Console) • FINS commands • File memory instructions • Simple backup operation • Auxiliary Area Flags
• Symbol table files • Comment files • Program index files	• CX-Programmer • Simple backup operation
• Unit/Board backup files	• Simple backup operation



Restrictions on File Use

Files are formatted in DOS, and therefore can be used as regular files on a Windows computer.

Files

Files are identified by file names and extensions, as shown in the following table. A file name is written using the following characters: Letters A to Z, numbers 0 to 9, !, &, $, #, `, {, }, –, ^, (,), and _

The following characters cannot be used in file names: ,, ., /, ¥, ?, *, “, :, :, <, >, =, +, space, and 2-byte characters.

The filename extensions depend upon the type of file being stored.

Directories

The CS/CJ-series CPU Units can access files located in subdirectories. Specify the directory location in file memory where the file is stored. Directories can be specified up to 5 subdirectories deep (counting the root directory), unless a Programming Console is being used. Only the root directory can be accessed from a Programming Console. The maximum length of a directory path is 65 characters. When creating a Memory Card subdirectory with an operating system such as Windows, do not exceed the maximum subdirectory depth (5 subdirectories).

File Categories

The following 4 categories of files are managed (read and written) by the CPU Unit with file memory operations.

File type	Corresponding file type						
	Program file	Data file	Parameter file	Symbol table file	Comment file	Program index file	Unit/ Board backup file
General files	Yes	Yes	Yes	No	No	No	No
Automatic Transfer at Startup Files	Yes	Yes	Yes	No	No	No	No

File type	Corresponding file type						
	Program file	Data file	Parameter file	Symbol table file	Comment file	Program index file	Unit/ Board backup file
Simple Backup Files (See note.)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CX-Programmer Files	No	No	No	Yes	Yes	Yes	No

Note Not supported by CS-series CS1 CPU Units that are pre-EV1.

General-purpose Files

These files can be manipulated (read, written, etc.) using Programming Devices (CX-Programmer or Programming Console), FINS commands, instructions, or Auxiliary Area control bit operations. The file names can be defined by the user.

Type	Name[1]	Extension	Description	Explanation	
Program File	********	.OBJ	User program	• Contains the programs in the CPU Unit's cyclic and interrupt tasks.	
Parameter Area File	********	.STD	PLC Setup, registered I/O table, routing tables, CPU Bus Unit settings, etc.	• Contains the CPU Unit's internal Parameter Area data. • The user does not have to distinguish parameter data in the file by type.	
Data File	********	.IOM	Specified range in I/O memory	• Data from start to end word in word units (16 bits) located in one area. • The area can be the CIO, HR, WR, AR, DM, or EM Area.	Binary format
		.TXT			TXT format (non-delimited or tab-delimited) (See note 1.)
		.CSV			CSV format (comma-delimited) (See note 1.)

Note
1. Not supported by CS-series CS1 CPU Units that are pre-EV1.
2. File names, represented by "********" above, consist of up to 8 ASCII characters.

Files Automatically Transferred at Startup

This file is automatically transferred from the Memory Card to the CPU Unit when the power is turned ON. There are two ways to transfer files automatically at startup: transferring with a parameter file and transferring without a parameter file.

Type of automatic transfer	File name	Purpose
Transfer with parameter file	AUTOEXEC or ATEXEC□□	Replacement due to a change in the program
Transfer without parameter file (See note.)	REPLACE or REPLC□□	Replacement due to a change in the program and network settings

Note Supported by CS/CJ-series CPU Units with unit version 2.0 or later.

The *File* column indicates the files that must be present in the Memory Card to enable automatic transfer at startup.

There are two ways to transfer files automatically at Startup: Transferring with a parameter area file and transferring without a parameter area file.

Files Used When Transferring with the Parameter File

Type	Name[1]	Extension	Description	Explanation	File
Program File	AUTOEXEC	.OBJ	User program	• Contains the programs in the CPU Unit's cyclic and interrupt tasks. • The file does not have to be on the Memory Card even when automatic transfer at startup is specified. • Transfer will not be possible unless the Memory Card also contains a parameter area file (AUTOEXEC.STD).	Required
Parameter Area File	AUTOEXEC	.STD	PLC Setup, registered I/O table, routing tables, CPU Bus Unit settings, etc.	• Contains the CPU Unit's internal Parameter Area data. • The file must be on the Memory Card when automatic transfer at startup is specified. • The user does not have to distinguish parameter data in the file by type. • Initial setting data will automatically be stored at specific locations in the CPU Unit at startup • The parameter area file will not be transferred if the Memory Card contains a program file called REPLACE.OBJ.	Required
Data File	AUTOEXEC	.IOM	I/O memory data (Contains the specified number of words of data beginning at D20000.)	• Store DM data beginning at D20000 in a file named AUTOEXEC.IOM. • At startup, all of the data in the file will be transferred to the DM Area beginning at D20000. • This file does not have to be on the Memory Card when the automatic transfer at startup function is being used.	---
	ATEXECDM	.IOM	I/O memory data (Contains the specified number of words of data beginning at D00000.)	• Store DM data beginning at D00000 in a file named ATEXECDM.IOM. • At startup, all of the data in the file will be transferred to the DM Area beginning at D00000. • This file does not have to be on the Memory Card when the automatic transfer at startup function is being used. **Note** The data in this file has higher priority if it overlaps the DM data contained in AUTOEXEC.IOM.	---
	ATEXECE□	.IOM	EM Area data (bank □) (Contains the specified number of words of data beginning at E□_00000.)	• Store data for EM bank □ beginning at E□_00000 in a file named ATEXECE□.IOM. The maximum bank number depends upon the model of CPU Unit being used. • At startup, all of the data in the file will be transferred to EM bank □ beginning at E□_00000. • This file does not have to be on the Memory Card when the automatic transfer at startup function is being used.	---

Files Used When Transferring without the Parameter File

Type	Name[1]	Extension	Description	Explanation	File
Program File	REPLACE	.OBJ	User program	• The contents is the same as that of AUTOEXEC.OBJ. • This file will be transferred at startup even if there is not a parameter area file (AUTOEXEC.STD).	Required
Parameter Area File	Not required.	---	---	The parameter area file will not be transferred regardless of the file name.	---
Data File	REPLACE	.IOM	I/O memory data (Contains the specified number of words of data beginning at D20000.)	• The contents is the same as that of AUTOEXEC.IOM. • This file will be transferred at startup if the Memory Card also contains a program file called REPLACE.OBJ.	---
	REPLCDM	.IOM	I/O memory data (Contains the specified number of words of data beginning at D00000.)	• The contents is the same as that of ATEXECDM.IOM. • This file will be transferred at startup if the Memory Card also contains a program file called REPLACE.OBJ.	---
	REPLCE□	.IOM	EM Area data (bank □) (Contains the specified number of words of data beginning at E□_00000.)	• The contents is the same as that of ATEXECE□.IOM. • This file will be transferred at startup if the Memory Card also contains a program file called REPLACE.OBJ.	---

Note Early versions of the CS/CJ-series CPU Units (CPU Units without a unit version number) do not support the files shown in the table above, which are transferred in an automatic startup transfer without the parameter file.

Backup Files

These files are transferred between the Memory Card and CPU Unit during simple backup operations. The file name is fixed as BACKUP□□. Files marked with a circle must exist in the Memory Card when performing a simple backup read (writing from the Memory Card to the CPU Unit).

Type	Name	Description	Explanation
Program file	BACKUP .OBJ	User program	Contains the programs in the CPU Unit's cyclic and interrupt tasks.
Parameter file	BACKUP .STD	PLC Setup, registered I/O table, routing tables, CPU Bus Unit settings. (See note 2.)	• Contains all initial settings for one CPU Unit. • The user does not have to distinguish parameter data in the file by type.

Type	Name	Description	Explanation
Data file	BACKUP .IOM	DM Area words allocated to Special I/O Units, CPU Bus Units, and Inner Boards (CS Series only)	Contains DM data from D20000 to D32767.
	BACKUPIO .IOR	I/O memory data areas	• Contains all of the data in the CIO, WR, HR, and AR data areas as well as timer/counter Completion Flags and PVs. **Note** CIO Area, Work Area, Timer Completion Flags, timer present values, and forced status are cleared when data is read from the Memory Card at startup (default behavior) The PLC Setup, however, can be used to hold the status of the IOM Hold Bit and the Forced Status Hold Bit.
	BACKUPDM .IOM	General-purpose DM Area	Contains DM data from D00000 to D19999.
	BACKUPE□ .IOM	General-purpose EM Area	• Contains all of the EM data for EM bank □ with addresses ranging from E□_00000 to E□_32767. (The maximum bank number depends upon the model of CPU Unit being used.) • When data is backed up to the Memory Card, all of the data in each EM bank is automatically written to a separate file.
Unit/Board backup files	BACKUP□□ .PRM (where □□ is the unit address of the Unit/Board being backed up)	Data for Unit or Board	• Control backup data from one Unit or Board. Refer to *5-2-6 Simple Backup Function* for details.
Symbol table files (See note 1.)	BKUPSYM .SYM	Global symbol tables, local symbol tables, and automatically allocated area setup data	Includes the following data within the CX-Programmer's global/local symbol table: Variables, addresses, data types, I/O comments Includes the data set in the CX-Programmer's automatic PLC address allocation settings.
Comment files (See note 1.)	BKUPCMT .CMT	Rung comments and comments	CX-Programmer rung comments and comments.
Program index files (See note 1.)	BKUPPRG .IDX	CX-Programmer section names and section comments.	CX-Programmer section delimiter data (The delimiter location, however, depends on the section delimiter within the program.)

Note

1. The following backup files can be created only when using CS/CJ-series CPU Units with unit version 3.0 or later.
   Symbol table files, comment files, and program index files
   These files are automatically created from the files in either the Memory Card, EM file memory, or comment memory.
2. One example of the CPU Bus Unit settings would be the Data Link Tables. Refer to the operation manuals for specific Units for other setup data.

CX-Programmer Files

- These files are created by the CX-Programmer. The file names created in file memory are fixed. When a project is transferred, any one of the following memory options can be selected as the transfer destination for these CX-Programmer files.
- Memory Card
- EM file memory

- Comment memory (in the CPU Unit's flash memory)

Note
1. Data can be transferred to comment memory only when the CX-Programmer version is version 5.0 or higher and the CPU Unit has a unit version or 3.0 or later.
2. With CX-Programmer version 4.0 or lower, these files cannot be stored in comment memory, even if a CPU Unit with unit version 3.0 or later is being used.

Type	Name	Description	Explanation
Symbol table files	SYMBOLS .SYM	Global symbol tables and local symbol tables	• Includes the variables in the CX-Programmer's global/local symbol tables, addresses, data types, and I/O comment information. • Also includes the data set in the CX-Programmer's automatic PLC address allocation.
Comment files	COMMENTS .CMT	Rung comments and comments (annotations)	Contains the CX-Programmer's rung comment and comment information.
Program index files	PROGRAM .IDX	Section names, section comments	Contains the CX-Programmer's section delimiter information, although the delimiter locations depend on the section delimiter instructions in the program. **Note** Supported by CX-Programmer version 2.0 or higher only.

Note With CX-Programmer version 1.2 or higher, the symbol table files and comment files in the above table can be transferred online between the CX-Programmer and personal computer RAM and between the personal computer RAM and the memory storage device.

Directories

It is possible to access files in subdirectories with CS/CJ-series PLCs, but Programming Consoles can access files only when they are in the root directory. The maximum length of a directory path is 65 characters. Be sure not to exceed the maximum number of characters when creating subdirectories in the Memory Card with a program such as Windows.

File Sizes

The size of files in bytes can be calculated with the equations in the following table.

File type	File size
Data files (.IOM)	(Number of words × 2) + 48 bytes Example: Entire DM Area (D00000 to D32767) (32,768 words × 2) + 48 = 65,584 bytes
Data files (.TXT or .CSV)	The file size depends upon the number of delimiters and carriage returns being used. The delimiter code is one byte and the carriage return code is two bytes. Example 1: Non-delimited words, no carriage return 123456789ABCDEF012345678 occupies 24 bytes. Example 2: Delimited words, carriage return every 2 fields 1234,5678↵ 9ABC,DEF0↵ 1234,5678↵ occupies 33 bytes. Example 3: Delimited double words, carriage return every 2 fields 56781234,DEF01234↵ 56781234↵ occupies 29 bytes.

File type	File size
Program files (.OBJ)	(Number of steps used × 4) + 48 bytes (See note.)
Parameter files (.STD)	16,048 bytes

Note Calculate the number of steps in the program file by subtracting the available UM steps from the total UM steps. These values are shown in the CX-Programmer's Cross-Reference Report. Refer to the *CX-Programmer Operation Manual* for details.

Data Files

General-purpose Files

1,2,3... 1. There are three kinds of general-purpose data files, with filename extensions IOM, TXT, and CSV. (The TXT and CSV files: Not supported by CS-series CS1 CPU Units that are pre-EV1.)

Filename extension	Contents	Purpose
.IOM	Binary format CS/CJ-series data format	I/O memory backup
.TXT (See note.) .CSV	In these data formats, 1-word or 2-word fields in I/O memory are converted to ASCII data.Records can be delimited with carriage returns.	Exchanging data with spreadsheet software

Note Reading and Writing TXT and CSV Data Files:
TXT and CSV data files can be read and written with FREAD(700) and FWRIT(701) only.

The following six data formats are used in text and CSV files.

Extension	Data format	Contents	
		I/O memory size per field	Delimiter
.TXT (See notes.)	Non-delimited words	1 word	None
	Non-delimited double words	2 words	None
	Tab-delimited words	1 word	Tab code
	Tab-delimited double words	2 words	Tab code
.CSV (See notes.)	Comma-delimited words	1 word	Comma
	Comma-delimited double words	2 words	Comma

Note a) Precautions on Characters:
Data cannot be written to I/O memory properly if the TXT or CSV file contains characters other than hexadecimal characters (0 to 9, A to F, or a to f.)

b) Precautions on Field Size:
When words are being used, data cannot be written to I/O memory properly if the TXT or CSV file contains fields that are not 4-digit hexadecimal. Likewise, when double words are being used, data cannot be written properly if the file contains fields that are not 8-digit hexadecimal.

c) Storage Order:
When words are being used, I/O memory data is converted to ASCII and stored in one-word fields in order from the lowest to the

highest I/O memory address.
When double words are being used, I/O memory data is converted to ASCII and stored in two-word fields in order from the lowest to the highest I/O memory address. (Within the two-word fields, the higher-address word is stored first and the lower-address word is stored second.)

d) Delimiters:
When there are no delimiters, the fields are packed consecutively and then stored. When delimited by commas, commas are inserted between fields before they are stored. When delimited by tabs, tab codes are inserted between fields before they are stored.
When delimiters (commas or tabs) are specified in FREAD(700), the data is read as delimited data with one-word delimiters (commas or tabs).

e) Carriage Returns:
Data is packed consecutively when carriage returns are not used. When carriage returns are used, a carriage return code is inserted after the specified number of fields. An offset from the beginning of the file (starting read word or starting write word) cannot be specified in the FREAD(700)/FWRIT(701) instructions if carriage returns are used in the file.

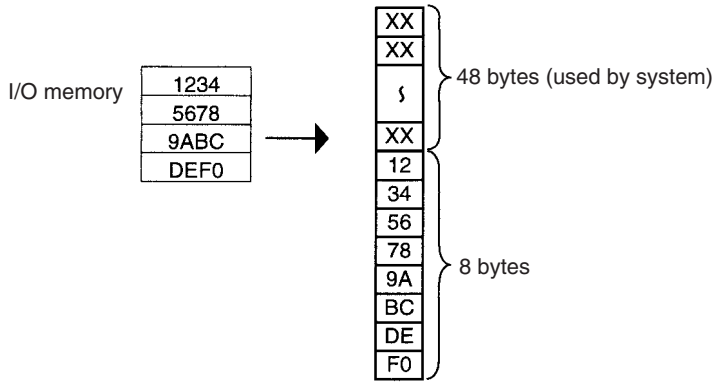
f) Number of Fields:
The overall amount of data in the file depends upon the number of fields (number of write items) specified in the FWRIT(701) instruction and the number of words/field.
The size is specified with the number of fields.
With word data, 1 word in I/O memory = 1 field
With long word data, 2 words in I/O memory = 1 field.

2. Data files do not contain information indicating what data is stored, i.e., what memory area is stored. Be sure to give file names that indicate the contents, as shown in the examples below, to aid in file management.
Examples: D00100.IOM, CIO0020.IOM
Data from the beginning of the file will be written starting at the address specified in I/O memory even if the data originally written to the data file (IOM, TXT, or CSV) is not from the same area. For example, if CIO data in a file is written to the DM Area from a Programming Device, the data will be read to the DM Area of the CPU Unit without any indication that the area is different.

Note Data files with the TXT and CSV format contain hexadecimal (0 to 9, A to F) data that allows the I/O memory numerical data to be exchanged with spreadsheet programs.

IOM Data File Structure

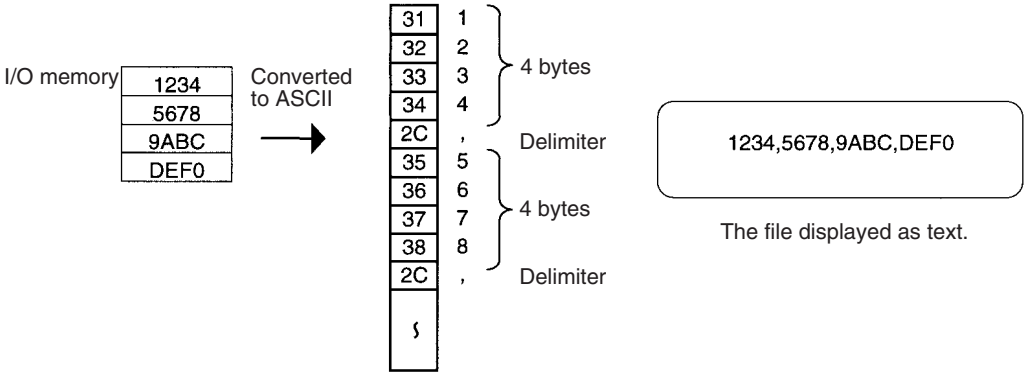
The following illustration shows the binary data structure of a data file (ABC.IOM) containing four words from I/O memory: 1234 Hex, 5678 Hex, 9ABC Hex, and DEF0 Hex. The user, however, does not have to consider the data format in normal operations.



Contents of ABC.IOM

CSV/TXT Data File Structure (Single Word)

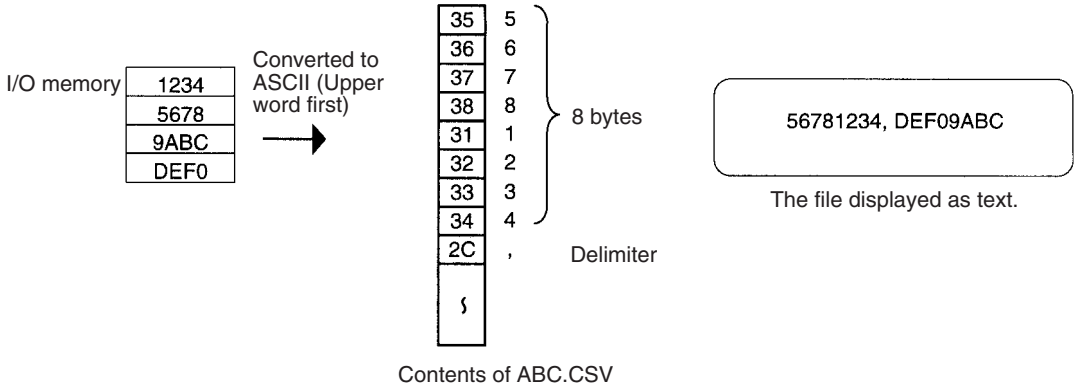
The following illustration shows the data structure of a CSV data file (ABC.CSV) with single-word fields containing four words from I/O memory: 1234 Hex, 5678 Hex, 9ABC Hex, and DEF0 Hex. The structure of the TXT file with single-word fields is the same.



Contents of ABC.CSV

CSV/TXT Data File Structure (Double Word)

The following illustration shows the data structure of a CSV data file (ABC.CSV) with double-word fields containing four words from I/O memory: 1234 Hex, 5678 Hex, 9ABC Hex, and DEF0 Hex. The structure of the TXT file with double-word fields is the same.



Contents of ABC.CSV

Creating Data Files with Spreadsheet Software

Use the following procedure to create TXT and CSV data files with spreadsheet software such as Microsoft Excel.

- Set the cell contents to characters.

- Input 4 characters in each cell if single-word fields are being used or 8 characters if double-word fields are being used. For example, if single-word fields are being used input 000A, not just A.
- Be sure to input only hexadecimal characters (0 to 9, A to F, or a to f) in the cells. Other characters and codes cannot be used.

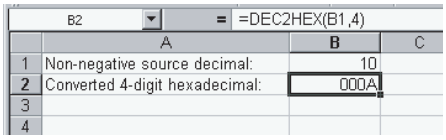
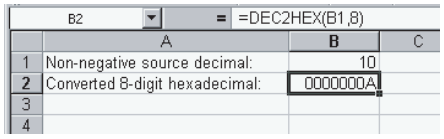
When you want to store hexadecimal digits in I/O memory, it is helpful to convert the spreadsheet's decimal inputs to hexadecimal. Use the following procedure to convert to hexadecimal.

1,2,3...

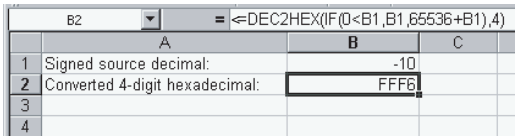
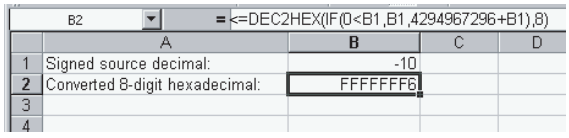
1. Select ***Add-Ins...*** from the Tools Menu.
2. Select **Analysis ToolPak** in the Add-Ins Menu.
3. Select ***Function*** from the Insert Menu at the cell where the function will be used.
4. Select ***DEC2HEX (number, digits)*** from Engineering in the Category Field.
5. When converting to 4-digit hexadecimal, input the following at the number variable: IF(0<=cell location,cell location,65535+cell location)

   When converting to 8-digit hexadecimal, input the following at the number variable: IF(0<=cell location,cell location,4294967296+cell location)

- **Example 1:** Inputting non-negative decimal values.

Item	Converting unsigned decimal to 4-digit hexadecimal	Converting unsigned decimal to 8-digit hexadecimal
Function used	DEC2HEX(*cell_location*,4)	DEC2HEX(*cell_location*,8)
Example	Input 10 in decimal and convert to 000A in 4-digit hexadecimal. 	Input 10 in decimal and convert to 0000000A in 8-digit hexadecimal. 

- **Example 2:** Inputting signed decimal values.

Item	Converting signed decimal to 4-digit hexadecimal	Converting signed decimal to 8-digit hexadecimal
Function used	DEC2HEX(IF(0<=*cell_location*,*cell_location*,65536+*cell_location*),4)	DEC2HEX(IF(0<=*cell_location*,*cell_location*,4294967296+*cell_location*),8)
Example	Input -10 in decimal and convert to FFF6 in 4-digit hexadecimal. 	Input -10 in decimal and convert to FFFFFFF6 in 8-digit hexadecimal. 

Data Files Transferred Automatically at Startup

There are 6 kinds of files that are transferred automatically at startup when the automatic transfer at startup function is being used.

Data file	Transferring with the parameter file	Transferring without the parameter file
Data file starting at D20000 (See note.)	AUTOEXEC.IOM	REPLACE.IOM
Data file starting at D00000	ATEXECDM.IOM	REPLCDM.IOM
Data file starting at E□_20000	ATEXECDM.IOM	REPLCDM.IOM

Note This part of the DM Area is allocated to Special I/O Units, CPU Bus Units, and Inner Boards.

When creating the data files listed above, always specify the first address shown above (D20000, D00000, or E□_00000) and make sure that the size of the file does not exceed the capacity of the specified data area.

All of the data in each file will always be transferred starting at the specified first address (D20000, D00000, or E□_00000).

Note

1. When creating the AUTOEXEC.IOM, ATEXECDM.IOM, and ATEXECE□.IOM files or the REPLACE.IOM, REPLCDM.IOM, or REPLCE□.IOM files from a Programming Device (Programming Console or CX-Programmer), always specify the proper first address (D20000, D00000, or E□_00000) and make sure that the size of the file does not exceed the capacity of the DM Area or specified EM bank. The contents of the file will always be transferred starting at the proper first address (D20000, D00000, or E□_00000) even if another starting word is specified, which could result in the wrong data overwriting the contents of that part of the DM Area or EM bank. Furthermore, if the capacity of the DM Area or EM bank is exceeded (as is possible when making settings from the CX-Programmer), the remaining data will be written to EM bank 0 if the DM Area is exceeded or the following EM bank if an EM bank is exceeded.
2. When using the CX-Programmer, you can specify a data file that will exceed the maximum DM Area address D32767 or maximum EM Area address of E□_32767. If the AUTOEXEC.IOM file exceeds the boundary of the DM area, all remaining data will be written to the EM Area starting at E0_00000 and continuing in order of memory address and banks through the final bank. It is thus possible to automatically transfer data to both the DM and EM Areas at startup. Likewise, if the ATEXECE□.IOM file is larger than an EM bank, the remaining data will be written to subsequent EM banks.
3. The System Setups for Special I/O Units, CPU Bus Units, and the Inner Board (CS Series only) can be changed by using different AUTOEXEC.IOM files containing different settings for the Special I/O Unit Area (D20000 to D29599), CPU Bus Unit Area (D30000 to D31599), and the Inner Board Area (CS Series only, D32000 to D32099). Memory Cards can thus be used to create libraries of System Setup data for Special I/O Units, CPU Bus Units, and Inner Boards (CS Series only) for different systems or devices.

Backup Data Files

The backup function creates 4 kinds of data files as described below.

To backup data, turn pin 7 ON on the CPU Unit's DIP switch, insert the Memory Card, and press and hold the Memory Card Power Supply Switch for three seconds. The four backup files (BACKUP.IOM, BACKUPIO.IOR, BACKUPDM.IOM, and BACKUPE□.IOM) will be created automatically and written to the Memory Card.

The four backup files are used exclusively by the backup function, although three of the files (BACKUP.IOM, BACKUPDM.IOM, and BACKUPE□.IOM) can be created with Programming Device operations. (BACKUPIO.IOR cannot be created with Programming Device operations.)

5-1-3 Description of File Operating Procedures

The following table summarizes the 6 methods that can be used to read and write files.

Read: Transfers files from file memory to the CPU Unit.
Write: Transfers files from the CPU Unit to file memory.

OK: Possible; ---: Not possible

Operating procedure		Medium	Description	Entire program	Data Area data (See note 3.)	Parameter Area data	Symbol tables, comment files, program index files (See note 6.)	Unit/ Board backup files
Programming Device	CX-Programmer or Programming Console	Memory Card EM file memory	Read	**OK**	**OK**	**OK**	**---**	**---**
			Write	**OK**	**OK**	**OK**	**---**	**---**
			Other operations (See note 2.)	**OK**	**OK**	**OK**	**---**	**---**
	CX-Programmer only	Memory Card EM file memory, comment memory	Read	**---**	**---**	**---**	**OK**	**---**
			Write	**---**	**---**	**---**	**OK** (See note 6.)	**---**
			Other operations (See note 2.)	**---**	**---**	**---**	**---**	**---**
FINS command (See note 1.)		Memory Card EM file memory	Read	**OK**	**OK**	**OK**	**---**	**---**
			Write	**OK**	**OK**	**OK**	**---**	**---**
			Other operations (See note 2.)	**OK** (See note 4.)	**OK**	**OK**	**---**	**---**
FREAD(700) and FWRIT(701) Instructions		Memory Card EM file memory	Read data from one file.	**---**	**OK**	**---**	**---**	**---**
			Write data to one file.	**---**	**OK**	**---**	**---**	**---**
Auxiliary Area control bit operation replaces the entire program during operation. (Not supported by CS-series CS1 CPU Units that are pre-EV1)		Memory Card	Read	**OK**	**---**	**---**	**---**	**---**
Automatic Transfer at Startup		Memory Card	Read	**OK**	**OK**	**OK**	**---**	**---**
			Write	**---**	**---**	**---**	**---**	**---**
Backup operation (Not supported by CS-series CS1 CPU Units that are pre-EV1)		Memory Card	Read	**OK**	**OK**	**OK**	**OK**	**OK**
			Write	**OK**	**OK**	**OK**	**OK** (See note 6.)	**OK**

Note

1. FINS commands for file memory operations can be sent from host computers connected via a Host Link, another PLC connected to a network (using CMND(490)), or the local PLC's program (using CMND(490)). (For CS-series CS1 CPU Units that are pre-EV1, file memory operations cannot be executed using CMND(490) in the same CPU Unit for which the file memory operations are being performed.
2. Other Operations: Format file memory, read file data, write file data, change file name, read file memory data, delete file, copy file, create subdirectory, and change file name.

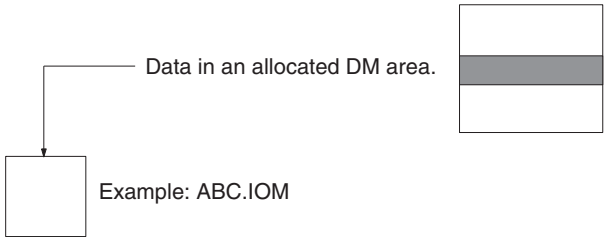
3. Data files with the TXT or CSV formats can be read and written only with the FREAD(700) and FWRIT(701) instructions. They cannot be read and written with a Programming Device.
4. Version V1.2 and higher versions of the CX-Programmer can be used to transfer program files (.OBJ) between the computer's RAM and a storage device.
5. With CS/CJ-series CPU Unit Ver. 2.0, files can be automatically transferred to the CPU Unit at startup without a parameter file stored in the Memory Card. This is achieved by changing the program file name to REPLACE.OBJ. Data files can also be transferred along with REPLACE.OBJ by using the following file names: REPLACE.IOM, REPLCDM.IOM, and REPLCE□.IOM.
6. When transferring projects to a CS/CJ-series CPU Unit with unit version 3.0 or later from a CX-Programmer version 5.0 or higher, the symbol tables, comment files, and program index files can be stored in the comment memory within the CPU Unit's internal flash memory (only if there is no Memory Card or EM file memory, or no available area).
   Backup files of the symbol tables, comment files, and program index files that are stored in either the Memory Card, EM file memory, or comment memory will be automatically created and stored in the Memory Card.

5-1-4 Applications

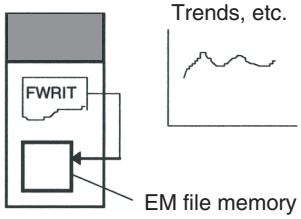
File memory can be used for the following applications.

Data Files

In this application, DM Area data settings (for Special I/O Units, CPU Bus Units, and Inner Boards (CS Series only)) are stored in the Memory Card. If the data file is named AUTOEXEC.IOM, the settings stored in the file will be automatically transferred when power is turned ON.



In this application, operation data (trends, quality control, and other data) generated during program execution is stored in EM file memory using the WRITE DATA FILE instruction (FWRIT(701)).

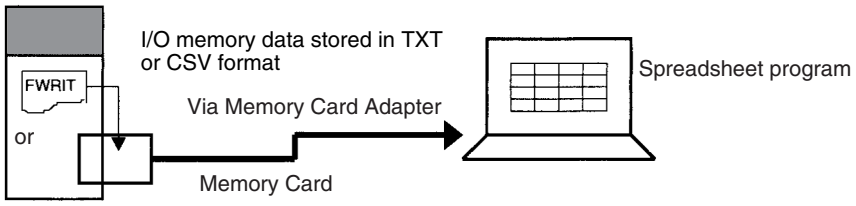


Note Data that is often accessed, such as trend data, is better stored in EM file memory rather than on a Memory Card.

ASCII Data Files (.TXT and .CSV)

Production data that has been saved on the Memory Card in the TXT or CSV format can be transferred to a personal computer via a Memory Card Adapter

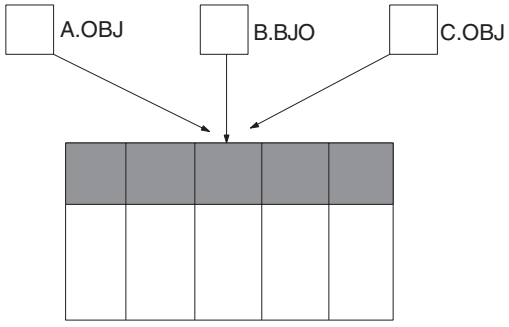
and edited with a spreadsheet program (Not supported by CS-series CS1 CPU Units that are pre-EV1).



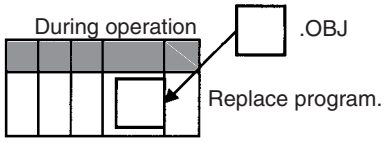
Conversely, data such as Special I/O Unit settings can be created with a spreadsheet program in TXT or CSV format, stored on a Memory Card, and read to the CPU Unit by FREAD(700) (Not supported by CS-series CS1 CPU Units that are pre-EV1).

Program Files(.OBJ)

In this application, programs that control different processes are stored on individual Memory Cards. The entire PLC configuration (program, PLC Setup, etc.) can be changed by inserting a different Memory Card and using the automatic transfer at startup function.

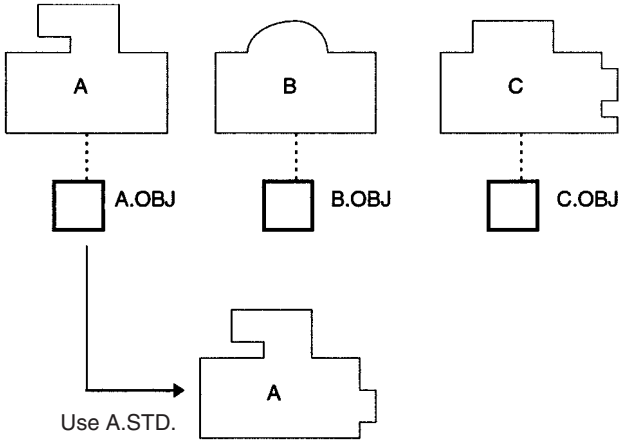


The entire program can be replaced during operation from the program itself (without a Programming Device) using an Auxiliary Area control bit (Not supported by CS-series CS1 CPU Units that are pre-EV1).



Parameter Area Files (.STD)

In this application, the PLC Setup, routing tables, I/O table, and other data for particular devices or machines are stored in Memory Cards. The data can be transferred to another device or machine just by switching the Memory Card.



Backup Files

The backup function can be used to store all of the CPU Unit's data (the entire I/O memory, program, and parameter area) on the Memory Card without a Programming Device. If a problem develops with the CPU Unit's data, the backed-up data can be restored immediately. (Not supported by CS-series CS1 CPU Units that are pre-EV1)

Symbol Table Files

The CX-Programmer can be used to save program symbols and I/O comments in symbols table files called SYMBOLS.SYM in Memory Cards or EM file memory.

When CX-Programmer version 5.0 or higher is used with a CS/CJ-series CPU Unit with unit version 3.0 or later, it is also possible to select the comment memory, which is in the CPU Unit's internal flash memory.

Comment Files

The CX-Programmer can be used to save program rung comments in comment files called COMMENTS.CMT in Memory Cards or EM file memory.

When CX-Programmer version 5.0 or higher is used with a CS/CJ-series CPU Unit with unit version 3.0 or later, it is also possible to select the comment memory, which is in the CPU Unit's internal flash memory.

5-2 Manipulating Files

The following procedures are used to read, write and otherwise work with files using the following methods.

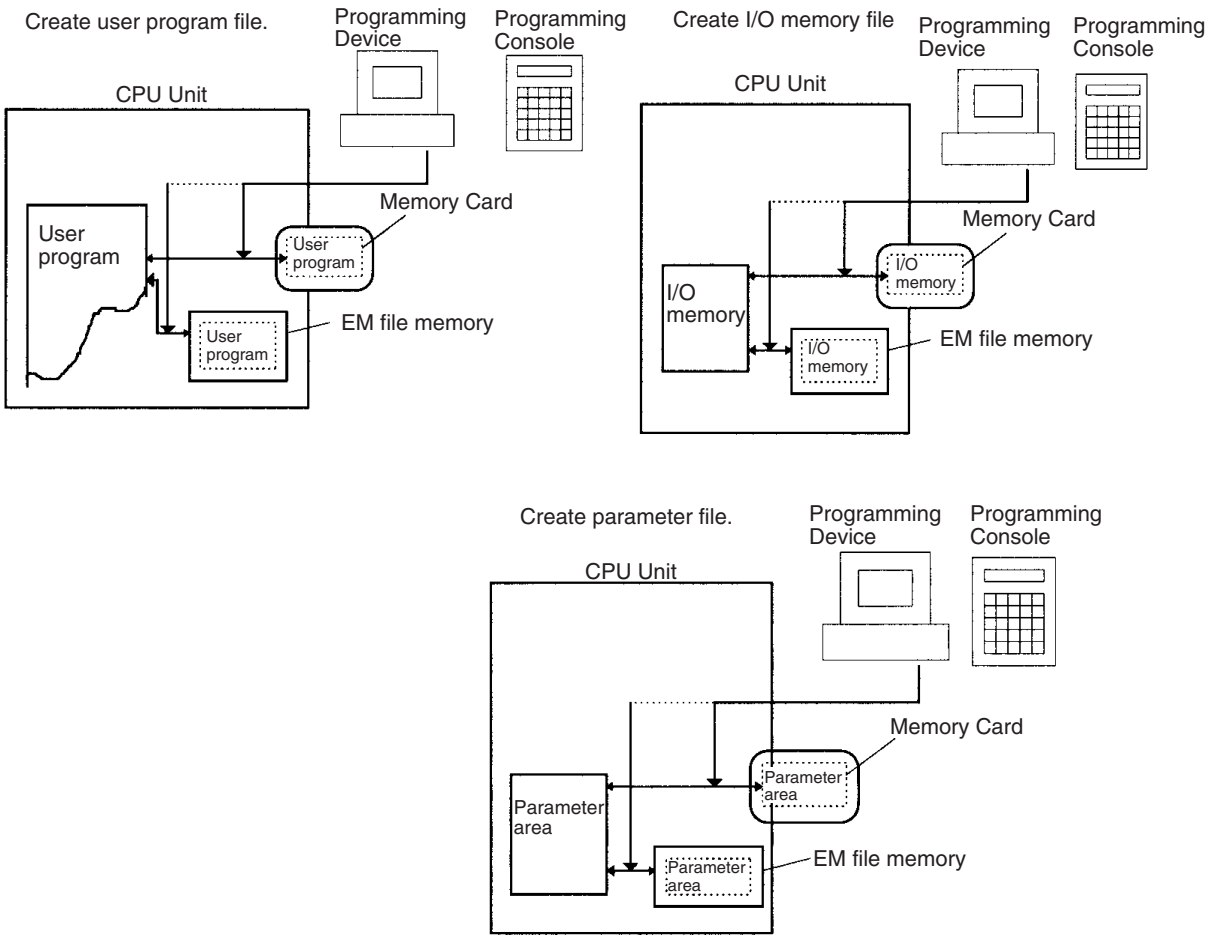
- Programming Devices
- FINS commands
- FREAD(700), FWRIT(701), and CMND(490) instructions in the user program (CMND(490): Not supported by CS-series CS1 CPU Units that are pre-EV1.)
- Replacement of the entire program using Auxillary Area control bits (Not supported by CS-series CS1 CPU Units that are pre-EV1)
- Automatic transfer at startup
- Backup function (Not supported by CS-series CS1 CPU Units that are pre-EV1)

5-2-1 Programming Devices (Including Programming Consoles)

The following operations are available through Programming Devices.

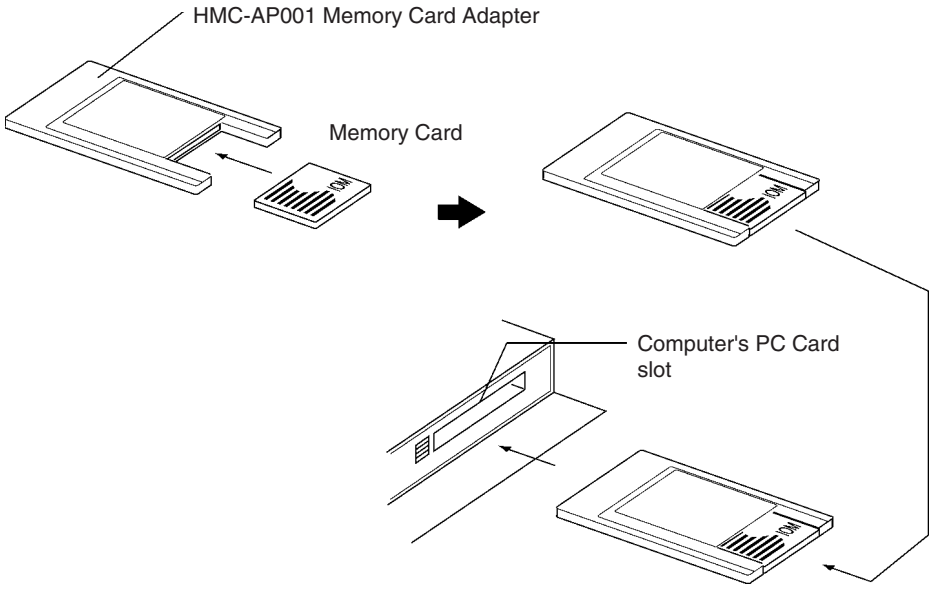
Operation		CX-Programmer	Programming Console
Reading files (transfer from file memory to CPU Unit)		**OK**	**OK**
Writing files (transfer from CPU Unit to file memory)		**OK** (See note.)	**OK** (See note.)
Comparing files (compare files in the CPU Unit and file memory)		Not possible	**OK**
Formatting file memory	Memory Cards	**OK**	**OK**
	EM files	**OK**	**OK**
Changing file names		**OK**	Not possible
Reading file memory data		**OK**	Not possible
Deleting files		**OK**	**OK**
Coping files		**OK**	Not possible
Deleting/Creating subdirectories		**OK**	Not possible

Note With CS/CJ-series CPU Unit Ver. 2.0 or later, password read protection can be used to prohibit writing a program file to file memory (i.e., a Memory Card or EM file memory).



Note With CS/CJ-series CPU Unit Ver. 2.0 or later and CX-Programmer Ver. 4.0 or higher, creating a backup program file (.OBJ) can be prohibited as an option when registering a password for the entire user program or for specific tasks. For details, refer to *Program Write Protection* under *1-4-2 Improved Read Protection Using Passwords* in the *CS Series PLC Operation Manual* or the *CJ Series PLC Operation Manual*.

A Memory Card can be installed in a computer's PLC Card slot with the HMC-AP001 Memory Card Adapter (sold separately). Installing a Memory Card in the computer allows the files in the card to be read and written by other programs, such as Windows Explorer.



CX-Programmer

Use the following procedure for file memory operations.

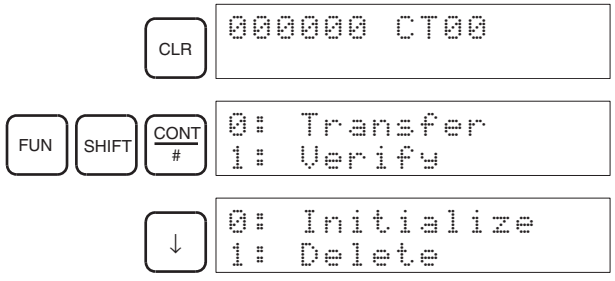
1,2,3...

1. Double-click the Memory Card icon in the Project Window with the CPU Unit online. The Memory Card Window will be displayed.
2. To transfer from the CPU Unit to file memory, select the program area, I/O memory area, or parameter area in the project work space, select ***Transfer*** from the File Memory, and then select transfer to the Memory Card or to EM file memory.

or To transfer from file memory to the CPU Unit, select file in file memory and then drag it to the program area, I/O memory area, or parameter area in the project work space and drop it.

Note Use project transfer operations to create and read symbol table files (SYMBOLS.SYM) and comment files (COMMENTS.CMT) on the CX-Programmer.

Programming Console



The following operations can be performed.

Item 1	Item 2	Item 3	Item 4	Item 5
0: Send	0: PLC to Memory Card	Select OBJ, CIO, HR, WR, AR, DM, EM, or STD.	Set transfer start and end addresses.	Media type, file name
	1: Memory Card to PLC	Select OBJ, CIO, HR, WR, AR, DM, EM, or STD.	Set transfer start and end addresses.	Media type, file name
1: Verify		Select OBJ, CIO, HR, WR, AR, DM, EM, or STD.	Set comparison start and end addresses.	Media type, file name

Item 1	Item 2	Item 3	Item 4	Item 5
2: Initialize		Enter 9713 (Memory Card) or 8426 (EM file memory).	---	---
3: Delete		Select OBJ, CIO, HR, WR, AR, DM, EM, or STD.	Media type, file name	---

Note The file types are shown in the following table.

Symbol	File type	
OBJ	Program file (.OBJ)	
CIO	Data file (.IOM)	CIO Area
HR		HR Area
WR		WR Area
AR		Auxiliary Area
DM		DM Area
EM0_		EM Area
STD	Parameter file (.STD)	

Precautions when Comparing Data after Transferring Parameter Files

Verification errors may occur at the Programming Console when comparing parameter data between files before transfer and the data after transfer if the parameter files (.STD) created in one CJ-series CPU Unit are saved to the Memory Card in another CJ-series CPU Unit with a different unit version. The occurrence of errors for different unit version combinations is shown in the following table.

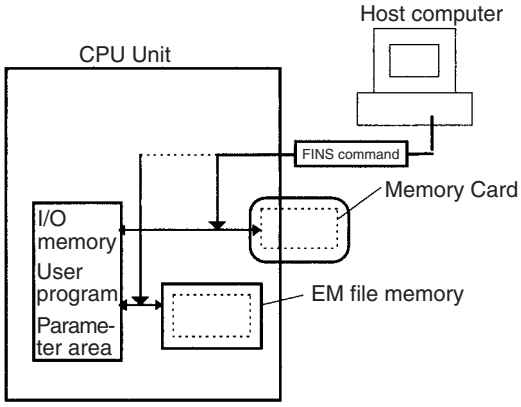
Source CPU Unit		Original I/O allocation status	Destination CPU Unit		
			Unit version of CPU Unit to which parameter files will be transferred		
			Pre-Ver. 2.0	Unit Ver. 2.0	Unit Ver. 3.0 or later
Unit version of the CPU Unit in which the parameter file were created	**Pre-Ver. 2.0**	Automatic allocation	Verification possible	Verification possible	Verification possible
		User-specified			Verification possible
	Unit Ver. 2.0	Automatic allocation			Verification possible
		User-specified			Verification possible
	Unit Ver. 3.0 or later	Automatic allocation		Verification error	Verification possible
		User-specified		Verification possible	

5-2-2 FINS Commands

The CPU Unit can perform the following file memory operations when it receives the proper FINS command. These are similar to the Programming Device functions.

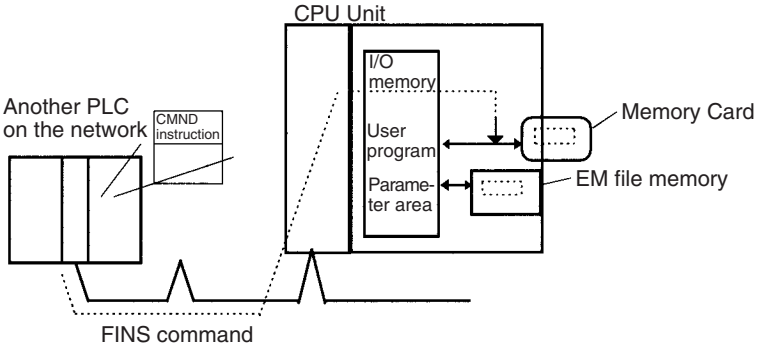
FINS Commands via Host Link

A computer connected via a Host Link System can send a FINS command with a Host Link header and terminator.

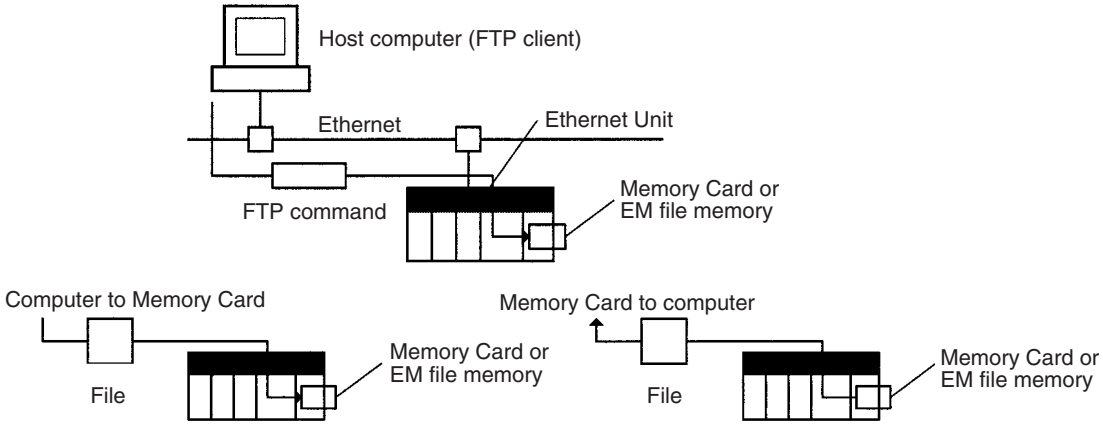


FINS Command from Another Network PLC

Another PLC on a network can send FINS command using CMND(490).



Note A computer on an Ethernet Network can read and write file memory (Memory Cards or EM file memory) on a CPU Unit through an Ethernet Unit. Data in files can be exchanged if the host computer functions as an FTP client and the CS/CJ-series PLC functions as an FTP server.



The following FINS commands can be used to perform a variety of functions, including reading and writing files.

Command	Name	Description
2201 Hex	FILE NAME READ	Reads file memory data.
2202 Hex	SINGLE FILE READ	Reads a specified length of file data from a specified position within a single file.
2203 Hex	SINGLE FILE WRITE	Writes a specified length of file data from a specified position within a single file.
2204 Hex	FILE MEMORY FORMAT	Formats (initializes) the file memory.
2205 Hex	FILE DELETE	Deletes specified files stored in the file memory.
2207 Hex	FILE COPY	Copies files from one file memory to another file memory.
2208 Hex	FILE NAME CHANGE	Changes a file name.
220A Hex	MEMORY AREA FILE TRANSFER	Transfers or compares data between the I/O memory area and the file memory.
220B Hex	PARAMETER AREA FILE TRANSFER	Transfers or compares data between the parameter area and the file memory.
220C Hex	PROGRAM AREA FILE TRANSFER	Transfers or compares data between the UM (User Memory) area and the file memory.
2215 Hex	CREATE/DELETE SUBDIRECTORY	Creates and deletes subdirectories.

Note The time from the CPU Unit's internal clock is used to date files created in file memory with the 220A, 220B, 220C, and 2203 commands.

5-2-3 User Instructions for File Memory Operations

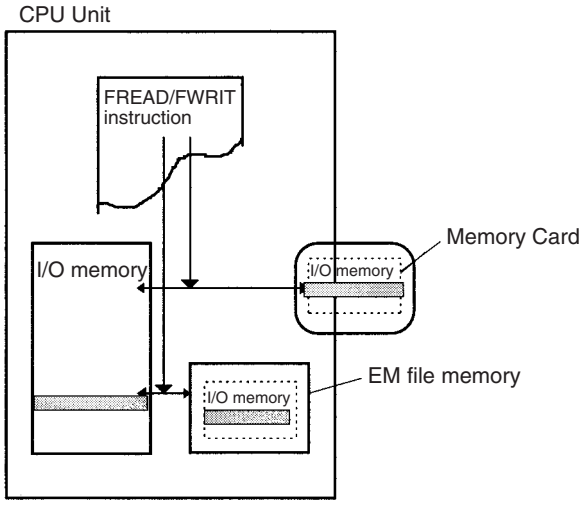
The FWRIT(701) (WRITE DATA FILE)/TWRIT(704) (WRITE TEXT FILE) instruction can be used to create a data file containing the specified I/O memory data in a Memory Card or EM file memory. It can also add to or overwrite from any point in existing files.

The FREAD(700) (READ DATA FILE) instruction will read I/O memory data from a specified location from a data file in a Memory Card or EM file memory and write it to the specified portion of I/O memory. It can read from any point in the specified file.

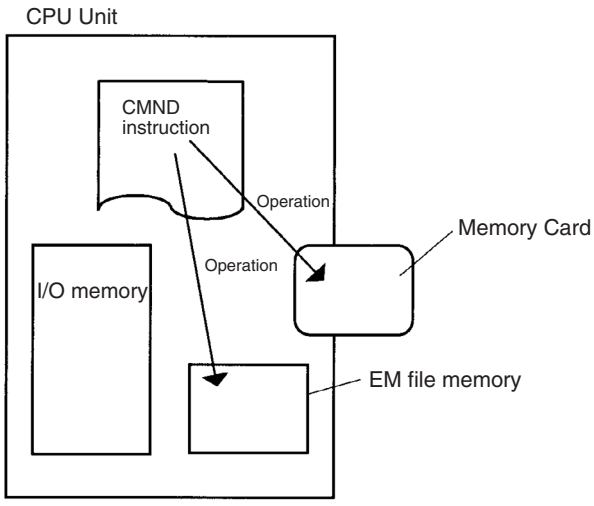
Note These instructions do not transfer the specified file, but rather the specified amount of data beginning at the specified start position in the file.

The CMND(490) (DELIVER COMMAND) instruction can be executed to issue a FINS command to the CPU Unit itself to perform file operations. File operations such as file formatting, deletion, copying, and renaming can be performed on files in the Memory Card or EM file memory (Not supported by CS-series CS1 CPU Units that are pre-EV1).

FREAD(700)/FWRIT(701): Transfers between I/O memory and file memory



CMND(490): File memory operations (Not possible for CS-series CPU Units that are pre-EV1)



FREAD(700)/FWRIT(701) Instructions

FREAD(700) and FWRIT(701) transfer data between I/O memory and file memory. All CJ CPU Units can transfer binary data (.IOM files) and the V1 CPU Units can also transfer ASCII files (.TXT and .CSV files).

Name	Mnemonic	Description
READ DATA FILE	FREAD(700)	Reads specified data file data or data elements to specified I/O memory.
WRITE DATA FILE	FWRIT(701)	Uses specified I/O memory area data to create a specified data file.
WRITE TEXT FILE (See note.)	TWRIT(704)	Creates a text file from ASCII data in the specified I/O memory data area location.

Note This instruction can be used in CPU Units with unit version 4.0 or later.

Transferring ASCII Files (Not supported by CS-series CS1 CPU Units that are pre-EV1)

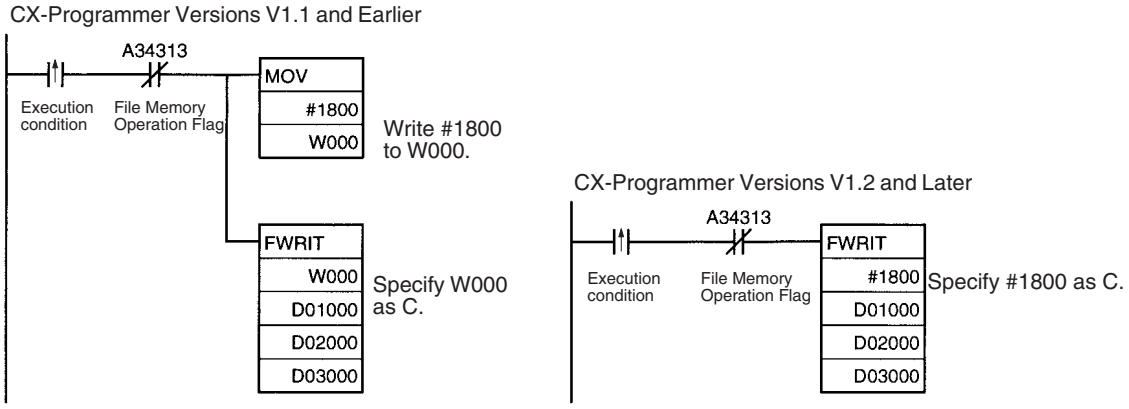
ASCII files can be transferred as well as binary files, so the third and fourth digits of the instruction's control word operand (C) indicate the type of data file being transferred and the number of fields between carriage returns.

Bits in C	Settings	Programming Device limitations
12 to 15	Data type 0: Binary (.IOM) 1: Non-delimited words (.TXT) 2: Non-delimited double-words (.TXT) 3: Comma-delimited words (.CSV) 4: Comma-delimited double-words (.CSV) 5: Tab-delimited words (.TXT) 6: Tab-delimited double-words (.TXT)	If CX-Programmer V1.1 or an lower version is being used, only 0 Hex (.IOM files) can be specified directly. If CX-Programmer V1.2 or a higher version (or a Programming Console) is being used, the control word bits can be set to between 0 and 6 Hex.
08 to 11	Carriage returns 0: No returns 8: Return every 10 fields 9: Return every 1 field A: Return every 2 fields B: Return every 4 fields C: Return every 5 fields D: Return every 16 fields	If CX-Programmer V1.1 or an lower version (or a Programming Console) is being used, only 0 Hex (no returns) can be specified directly. If CX-Programmer V1.2 or a higher version is being used, the control word bits can be set to 0 Hex or to between 8 and D Hex.

CX-Programmer V1.1 or Lower Version: Indirectly Setting the Control Word

When V1.1 or an lower version of CX-Programmer is being used, ASCII files cannot be transferred with FREAD(700) and FWRIT(701) if a constant is input for the control word to specify the data type and carriage return treatment. Only binary data with no carriage returns can be transferred if a constant is used.

ASCII files can be transferred with FREAD(700) and FWRIT(701), however, by indirectly setting the control word. Write the desired control word setting to a word and specify that word as the control word in FREAD(700) or FWRIT(701), as shown on the left in the following diagram.



Note The time from the CPU Unit's internal clock is used to date files created in file memory with FWRIT(701).

Only one file memory operation may be executed at a time, so FREAD(700) and FWRIT(701) must not be executed when any of the following file memory operations are being performed:

1,2,3...

1. Execution of A File Memory Instruction
2. Execution of CMND(490) to send a FINS command to the CPU Unit itself
3. Replacement of the entire program by Auxiliary Area control bit operations
4. Execution of a simple backup operation

Use the File Memory Operation Flag (A34313) for exclusive control of file memory instructions to prevent them from being executed while another file memory operation is in progress.

When FREAD(700) is being executed, the File Read Error Flag (A34310) will turn ON and the instruction won't be executed if the specified file contains data that is not hexadecimal data or if delimiters are not positioned 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.

Related Auxiliary Bits/Words

Name	Address	Operation
Memory Card Type	A34300 to A34302	Indicates the type of Memory Card, if any, that is installed.
EM File Memory Format Error Flag	A34306	ON when a format error occurs in the first EM bank allocated for file memory. OFF when formatting is completed normally.
Memory Card Format Error Flag	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 couldn't be written because the file was write-protected or 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.
No File Flag	A34311	ON when data could not be read because the specified file doesn't exist.
File Memory Operation Flag	A34313	ON for any of the following: The CPU Unit is processing a FINS command sent to itself using CMND(490). FREAD(700) or FWRIT(701) is being executed. The program is being overwritten using an Auxiliary Area control bit. A simple backup operation is being performed.
Accessing File Flag	A34314	ON when file data is actually being accessed.
Memory Card Detected Flag	A34315	ON when a Memory Card has been detected. (Not supported by CS-series CS1 CPU Units that are pre-EV1)
Number of Items to Transfer	A346 to A347	These words indicate the number of words or fields remaining to be transferred (32 bits). When a binary (.IOM) file is being transferred, this number is decremented each time a word is read. When a text or CSV file is being transferred, this number is decremented each time a field is transferred.

CMND(490): DELIVER COMMAND

CMND(490) can be used to issue a FINS command to the local CPU Unit itself to perform file memory operations such as formatting or deleting files. Make the following settings in CMND(490)'s control words when issuing a file-memory FINS command to the local PLC:

1,2,3...

1. Set the destination network address to 00 (local network) in C+2.
2. Set the destination unit address to 00 (PLC's CPU Unit) and the destination node to 00 (within local node) in C+3.
3. Set the number of retries to 0 in C+4. (The number of retries setting is invalid, so set it to 0.)

FINS Commands Related to File Memory

Refer to *5-2-2 FINS Commands* for information on FINS commands.

Note There are other FINS commands related to file memory that are not shown in the following table that can be executed. Refer to the *Communications Command Reference Manual* (W342) for details on FINS commands.

CMND(490) cannot be executed to the local CPU Unit if another CMND(490) or file memory instruction is being executed to another CPU Unit, A File Mem-

ory Instruction is being executed, the program is being replaced by an Auxiliary Area control bit operation, or a simple backup operation is being executed. Be sure to include the File Memory Operation Flag (A34313) as a normally closed condition to prevent CMND(490) from being executed while another file memory operation is in progress.

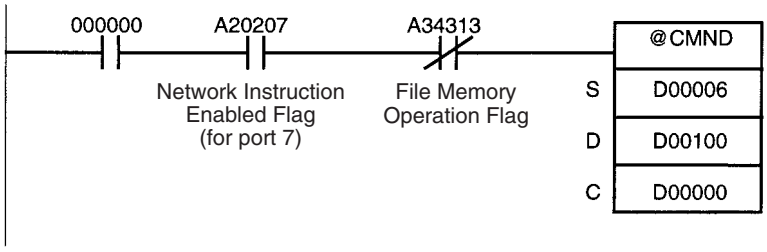
If CMND(490) cannot be executed for the local CPU Unit, the Error Flag will be turned ON.

Related Auxiliary Bits/Words

Name	Address	Operation
File Memory Operation Flag	A34313	ON for any of the following: • The CPU Unit is processing a FINS command sent to itself using CMND(490). • FREAD(700) or FWRIT(701) is being executed. • The program is being overwritten using an Auxiliary Area control bit. • A simple backup operation is being performed.
Memory Card Detected Flag	A34315	ON when a Memory Card has been detected. (Not supported by CS-series CS1 CPU Units that are pre-EV1)

Example Program

The following example shows how to use CMND(490) to create a subdirectory in the Memory Card.



When 000000 and A20207 are ON and A34313 is OFF, CMND(490) issues FINS command 2215 (CREATE/DELETE SUBDIRECTORY) is sent to the local CPU Unit and the response is stored in D00100 and D00101.

In this case, the FINS command creates a subdirectory named "CS1" within the OMRON directory in the CPU Unit's Memory Card. The response is composed of the 2-byte command code (2215) and the 2-byte response code.

		15–8	7–0	
S:	D00006	22	15	Command code: 2215 Hex (CREATE/DELETE SUBDIRECTORY)
S+1:	D00007	80	00	Disk number: 8000 Hex (Memory Card)
S+2:	D00008	00	00	Parameter: 0000 Hex (Create subdirectory.)
S+3:	D00009	43	53	Subdirectory name: CS1□□□□□.□□□ (□: a space)
S+4:	D00010	31	20	
S+5:	D00011	20	20	
S+6:	D00012	20	20	
S+7:	D00013	2E	20	
S+8:	D00014	20	20	
S+9:	D00015	00	06	Directory length: 0006 Hex (6 characters)
S+10:	D00016	5C	4F	Directory path: \OMRON
S+11:	D00017	4D	52	
S+12:	D00018	4F	4E	

		15–8	7–0	
C:	D00000	00	1A	Number of bytes of command data: 001A Hex (26 bytes)
C+1:	D00001	00	04	Number of bytes of response data: 0004 Hex (4 bytes)
C+2:	D00002	00	00	Destination address: 0000 Hex (local network)
C+3:	D00003	00	00	00 Hex (local node) and 00 Hex (CPU Unit)
C+4:	D00004	07	00	Response requested, communications port 7, 0 retries
C+5:	D00005	00	00	Response monitor time: FFFF Hex (6,553.5 s)

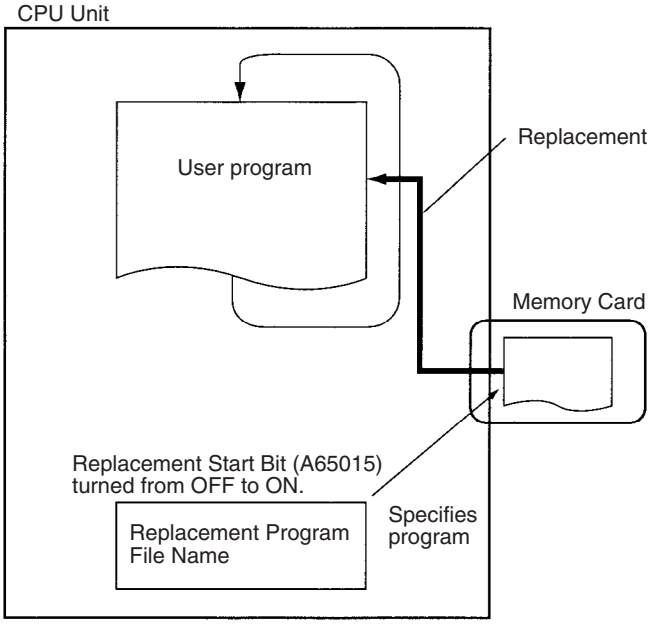
Note There are other FINS commands that can be sent to the local PLC in addition to the ones related to file memory operations that are listed in the table above.

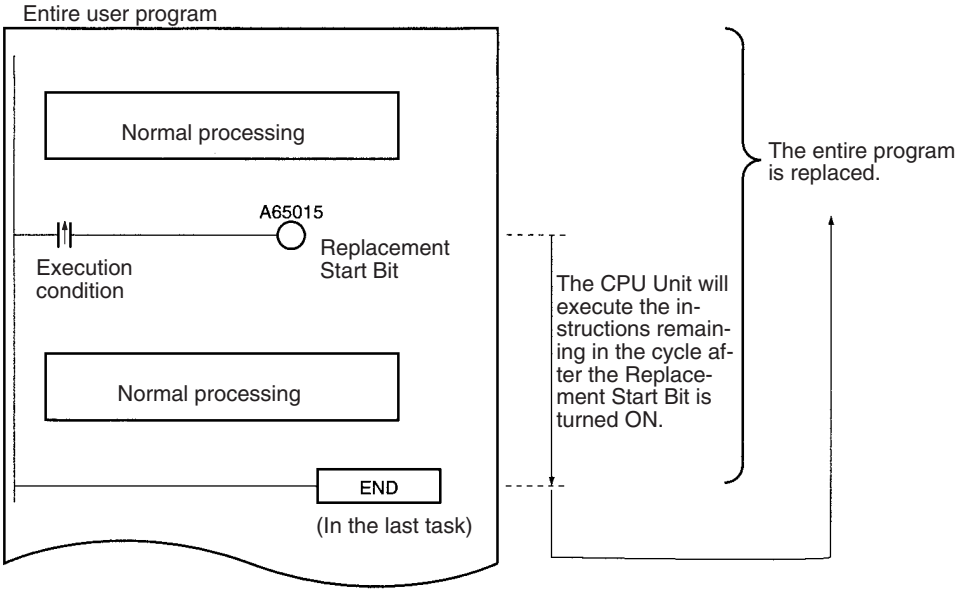
The File Memory Operation Flag must be used to prevent simultaneous execution of these other FINS commands, too.

5-2-4 Replacement of the Entire Program During Operation

The entire program can be replaced during operation (RUN or MONITOR mode) by turning ON the Replacement Start Bit (A65015). The specified file will be read from the Memory Card and it will replace that program will replace the executable program at the end of the current cycle. The replacement Program Password (A651) and Program File Name (A654 to A657) must be recorded in advance and the specified program file must exist on the Memory Card in order to replace the program during operation.

Note Early versions of the CS1 CPU Units (model numbers without a -V□ suffix) do not support this function.





The program can also be replaced when program execution is stopped (PROGRAM mode) by turning ON the Replacement Start Bit from a Programming Device.

Note The replacement program file cannot be read from EM file memory.

The Replacement Start Bit (A65015) can be turned ON at any location (program address) in the program. The CPU Unit will execute the instructions remaining in the cycle after the Replacement Start Bit goes from OFF to ON.

The program will not be executed while the program is being replaced. After the program has been replaced, operation will be started again just as if the CPU Unit were switched from PROGRAM mode to RUN or MONITOR mode.

The program will be replaced at the end of the cycle in which the Replacement Start Bit was turned from OFF to ON, i.e., after END(001) is executed in the last task in the program.

Note

1. Turn ON the IOM Hold Bit (A50012) if you want to maintain the status of I/O memory data through the program replacement.

   Turn ON the Forced Status Hold Bit (A50013) if you want to maintain the status of force-set and force-reset bits through the program replacement.

2. If the IOM Hold Bit (A50012) is ON before the program is replaced, the status of bits in I/O memory will be maintained after program replacement. Be sure that external loads will operate properly with the same I/O memory data.

   Likewise, if the Forced Status Hold Bit (A50013) is ON before the program is replaced, the status of force-set and force-reset bits will be maintained after program replacement. Be sure that external loads will operate properly with the same force-set and force-reset bits.

Replacement File

The program file specified in the Program File Name (A654 to A657) will be read from the Memory Card and will replace the existing program at the end of the cycle in which the Replacement Start Bit (A65015) is turned from OFF to ON.

File	File name and extension	Specifying the replacement file name (********)
Program file	********.OBJ	Write the replacement program file name to A654 through A657 before program replacement.

Conditions Required for Program Replacement

The following conditions are required in order to replace the program during operation.

- The program password (A5A5) has been written to A651.
- The program file specified in the Program File Name words (A654 to A657) exists in the Memory Card's root directory.
- The Memory Card has been detected by the CPU Unit. (A34315 ON)
- No fatal errors have occurred.
- No file memory operations are being executed. (A34313 OFF)
- Data is not being written to the Program Area.
- The access right is available. (For example, data is not being transferred from the CX-Programmer to the PLC.)

Note The program may be transferred in any operating mode.

CPU Operation during Program Replacement

The CPU Unit's operation will be as follows during program replacement:

- Program execution: Stopped
- Cycle time monitoring: No monitoring

Operations Continuing during and after Program Replacement

When the IOM Hold Bit (A50012) is ON, the data in the following memory areas will be maintained: the CIO Area, Work Area (W), Timer Completion Flags (T), Index Registers (IR), Data Registers (DR), and the current EM bank number.

Note Timer PVs will be cleared during program replacement.

If the IOM Hold Bit is ON when the program is transferred, loads that were being output before program replacement will continue to be output after replacement. Be sure that external loads will operate properly after program replacement.

The status of force-set and force-reset bits will be maintained through the program replacement if the Forced Status Hold Bit (A50013) is ON.

Interrupts will be masked.

If data tracing is being performed, it will be stopped.

Instruction conditions (interlocks, breaks, and block program execution) will be initialized.

Differentiation Flags will be initialized whether the IOM Hold Bit is ON or OFF.

Operations after Program Replacement

The status of the cyclic tasks depends upon their operation-start properties. (Their status is the same as it would be if the PLC were switched from PROGRAM to RUN/MONITOR mode.)

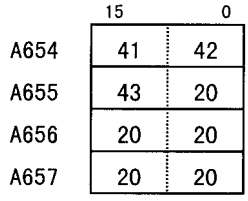
The First Cycle Flag (A20011) will be ON for one cycle after program execution resumes. (The status is the same as it would be if the PLC were switched from PROGRAM to RUN/MONITOR mode.)

Time Required for Program Replacement

Size of entire program	Peripheral servicing time set in PLC Setup	Approx. time required for program replacement
60 Ksteps	Default (4% of cycle time)	6 s
250 Ksteps		25 s

Related Auxiliary Bits/Words

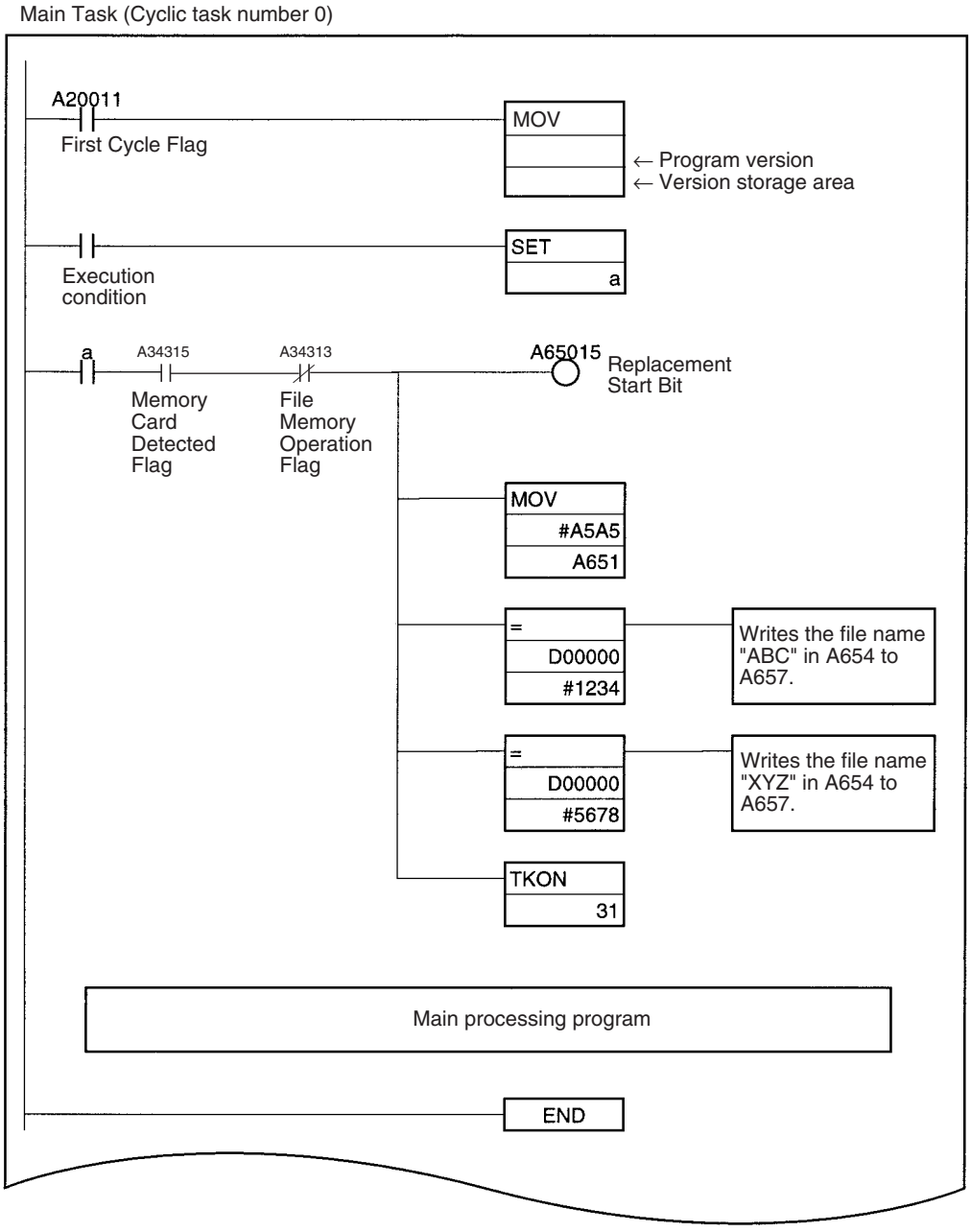
Name	Address	Operation
File Memory Operation Flag	A34313	ON for any of the following: The CPU Unit has sent a FINS command to itself using CMND(490). FREAD(700) or FWRIT(701) are being executed. The program is being overwritten using an Auxiliary Area control bit (A65015). A simple backup operation is being performed.
Memory Card Detected Flag (Not supported by CS-series pre-EV1 CS1 CPU Units)	A34315	ON when a Memory Card has been detected.
IOM Hold Bit	A50012	When this bit is ON, the contents of I/O memory are retained through program replacement.
Forced Status Hold Bit	A50013	When this bit is ON, the status of force-set and force-reset bits is maintained through program replacement.

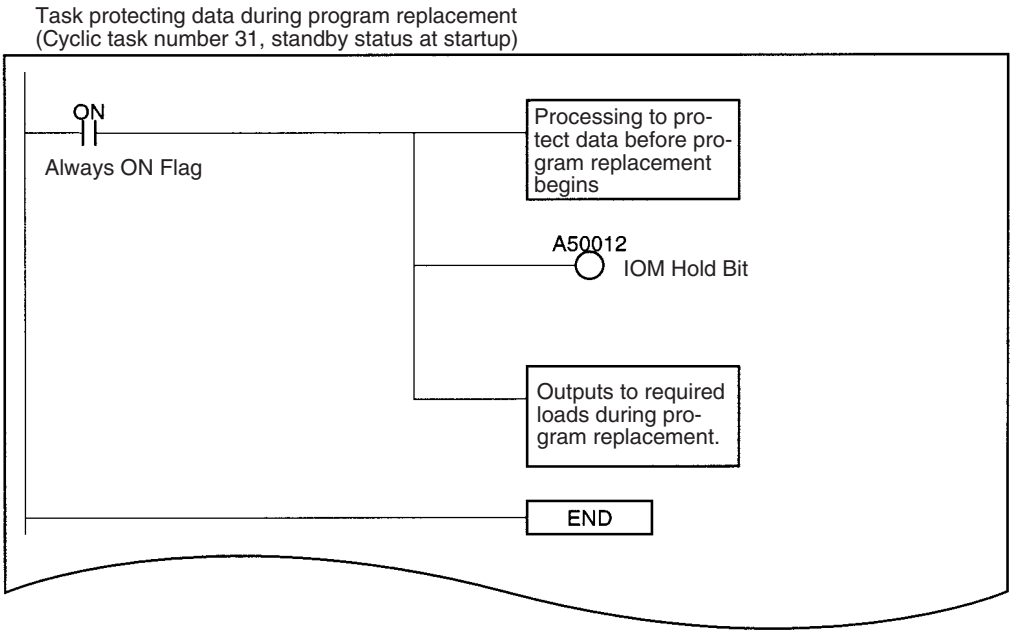
Name	Address	Operation
Replacement Completion Code (Not supported by CS-series pre-EV1 CS1 CPU Units)	A65000 to A65007	Codes for normal program replacement (A65014 OFF): 01 Hex: The program file (.OBJ) replaced the program. Codes for incomplete program replacement (A65014 ON): 00 Hex: A fatal error occurred. 01 Hex: A memory error occurred. 11 Hex: The program is write-protected. 12 Hex: The program password in A651 is incorrect. 21 Hex: A Memory Card is not installed. 22 Hex: The specified file does not exist. 23 Hex: The specified file is too large (memory error). 31 Hex: One of the following operations was being performed: • A file memory operation was being performed. • The program was being written. • The operating mode was being changed.
Replacement Error Flag (Not supported by CS-series pre-EV1 CS1 CPU Units)	A65014	Turned ON when an error occurred while trying to replace the program after A65015 was turned from OFF to ON. Turned OFF the next time that A65015 is turned from OFF to ON again.
Replacement Start Bit (Not supported by CS-series pre-EV1 CS1 CPU Units)	A65015	If this bit has been enabled by the setting the Program Password (A651) to A5A5 Hex, program replacement will start when this bit is turned from OFF to ON. Do not turn this bit from OFF to ON again during program replacement. This bit is automatically turned OFF when program replacement is completed (normally or with an error) or the power is turned ON. The status of this bit can be read from a Programming Device, PT, or host computer to determine whether program replacement has been completed or not.
Program Password (Not supported by CS-series pre-EV1 CS1 CPU Units)	A651	Write the password to this word to enable program replacement. A5A5 Hex: Enables the Replacement Start Bit (A65015). Other value: Disables the Replacement Start Bit (A65015). This bit is automatically turned OFF when program replacement is completed (normally or with an error) or the power is turned ON.
Program File Name (Not supported by CS-series pre-EV1 CS1 CPU Units)	A654 to A657	Before starting program replacement, write the file name of the replacement program file in these words in ASCII. Just write the 8-character filename; the .OBJ extension is added automatically. Write the characters in order from A654 (most significant byte first). If the file name has fewer than 8 characters, pad the remaining bytes with space codes (20 Hex). Do not include any NULL characters or spaces within the file name itself. The following example shows the data for the program file ABC.OBJ: 

Example Program 1

Store program files ABC.OBJ and XYZ.OBJ in the Memory Card and select one program or the other depending upon the value of D00000. Set D00000 to #1234 when selecting ABC.OBJ or set it to #5678 when selecting XYZ.OBJ.

Start and execute another task to perform any processing required before program replacement or IOM Hold Bit processing.

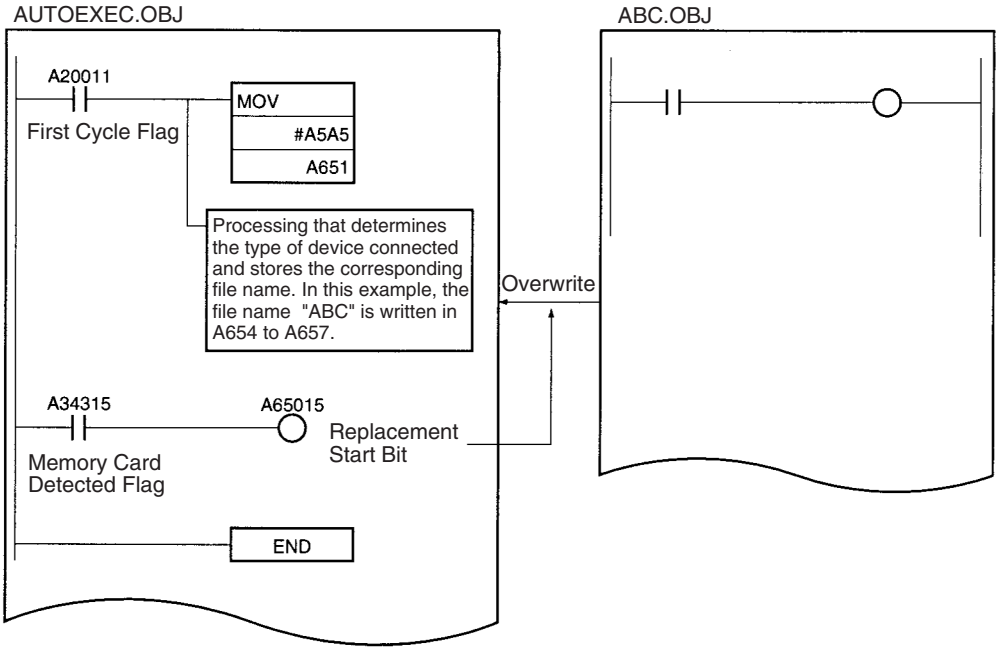




Example Program 2

Store program files for several devices and the program file for automatic transfer at startup (AUTOEXEC.OBJ or REPLACE.OBJ (see note)) in a Memory Card. When the PLC is turned ON, the automatic transfer at startup file is read and that program is replaced later with a program file for a different device.

Note REPLACE.OBJ is supported only by CS/CJ-series CPU Unit Ver. 2.0 or later.



5-2-5 Automatic Transfer at Startup

Automatic transfer at startup is used to read the user program, parameters, and I/O memory data from a Memory Card to the CPU Unit when the power is turned ON.

The following files can be read automatically to CPU Unit memory.

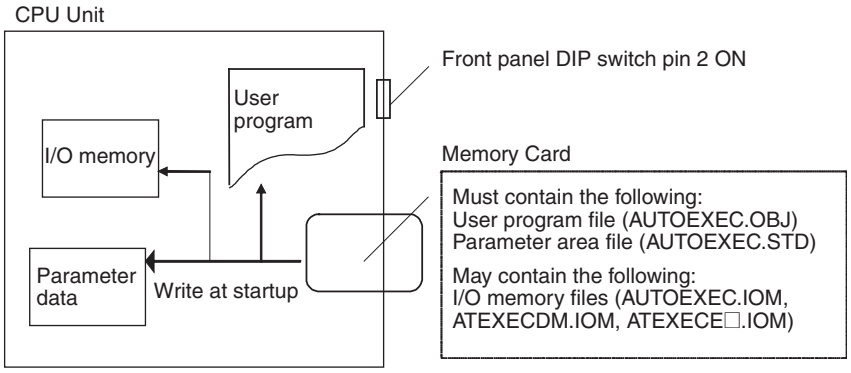
Note This function cannot be used to read EM file memory.

The program file name depends on whether a parameter area file is also going to be transferred.

Transferring a Parameter Area File

Use the following file names.

Program file: AUTOEXEC.OBJ
Parameter area file: AUTOEXEC.STD
Data files: AUTOEXEC.IOM, ATEXECDM.IOM, ATEXECE□.IOM



File	File name	At startup	Required for automatic transfer
Program File	AUTOEXEC.OBJ	The contents of this file are automatically transferred and overwrite the entire user program including CPU Unit task attributes.	Required on Memory Card.
Parameter Area File	AUTOEXEC.STD	The contents of this file are automatically transferred and overwrite all initial settings data in the CPU Unit.	Required on Memory Card.
Data File	AUTOEXEC.IOM	DM words allocated to Special I/O Units, CPU Bus Units, and Inner Boards (CS Series only). The contents of this file are automatically transferred to the DM Area beginning at D20000 when power is turned ON. (See note 1.)	Not required on Memory Card.
	ATEXECDM.IOM	General-purpose DM words The contents of this file are automatically transferred to the DM Area beginning at D00000 when power is turned ON. (Not supported by CS-series CS1 CPU Units that are pre-EV1) (See note 1.)	
	ATEXECE□.IOM	General-purpose DM words The contents of this file are automatically transferred to the EM Area beginning at E□_00000 when power is turned ON. (Not supported by CS-series CS1 CPU Units that are pre-EV1)	

Note

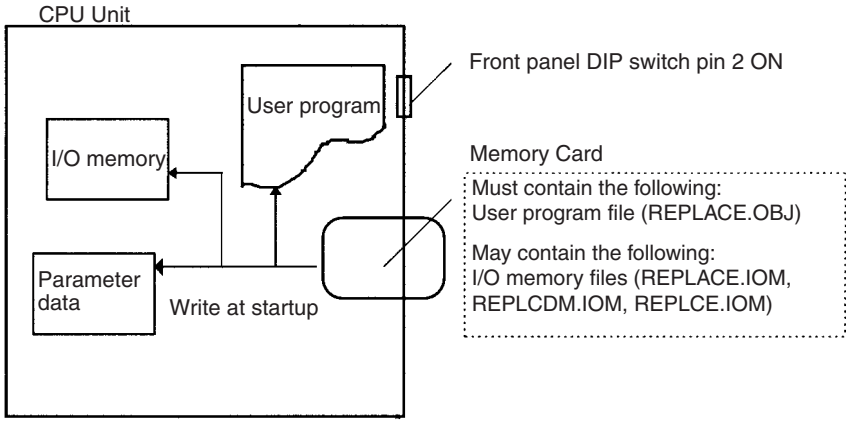
1. If the data contained in AUTOEXEC.IOM and ATEXECDM.IOM overlap, the data in ATEXECDM.IOM will overwrite any overlapping data transferred from AUTOEXEC.IOM since ATEXECDM.IOM is written later.
2. The program file (AUTOEXEC.OBJ) and parameter file (AUTOEXEC.STD) must be on the Memory Card. Without these files, automatic transfer will fail, a memory error will occur, and A40115 (Memory Error Flag: fatal error) will turn ON. (It is not necessary for the I/O memory file (AUTOEXEC.IOM) to be present.)

3. It is possible to create the AUTOEXEC.IOM, ATEXECDM.IOM, and ATEXECE□.IOM files from a Programming Device (Programming Console or CX-Programmer), with starting addresses other than D20000, D00000, and E□_00000 respectively. The data will be written beginning with the correct starting address anyway, but do not specify other starting addresses.
4. If DIP switch pin 7 is turned ON and pin 8 is turned OFF to use the simple backup function, the simple backup function will take precedence even if pin 2 is also ON. In this case, the BACKUP□□ files will be transferred to the CPU Unit but the automatic transfer at startup files will not be transferred. (Not supported by CS-series CS1 CPU Units that are pre-EV1.)
5. The automatic transfer at startup function can be used together with the program replacement function. The Replacement Start Bit (A65015) can be turned ON from program that is automatically transferred at startup to replace it with another program.
6. The automatic transfer function at startup and total program replacement using Auxiliary Area bits can be used together, i.e., the program automatically transferred to the CPU Unit at startup can contain programming to manipulate the Auxiliary Area bits to replace the program with another one.

Transferring without a Parameter Area File (CS/CJ-series CPU Unit Ver. 2.0 or Later Only)

Use the following file names.

Program file: REPLACE.OBJ
Parameter area file: Not required and not transferred regardless of the file name.
Data files: REPLACE.IOM, REPLCDM.IOM, REPLCE□.IOM



File	File name	At startup	Required for automatic transfer
Program File	REPLACE.OBJ Note: CS/CJ-series CPU Unit Ver. 2.0 or later only	The contents of this file are automatically transferred and overwrite the entire user program including CPU Unit task attributes.	Required on Memory Card.
Parameter Area File		Not transferred regardless of the file name.	Not required.

File	File name	At startup	Required for automatic transfer
Data File	REPLACE.IOM Note: CS/CJ-series CPU Unit Ver. 2.0 or later only	DM words allocated to Special I/O Units, CPU Bus Units, and Inner Boards (CS Series only). The contents of this file are automatically transferred to the DM Area beginning at D20000 when power is turned ON.	Not required on Memory Card.
	REPLCDM.IOM Note: CS/CJ-series CPU Unit Ver. 2.0 or later only	General-purpose DM words The contents of this file are automatically transferred to the DM Area beginning at D00000 when power is turned ON.	
	REPLCE□.IOM Note: CS/CJ-series CPU Unit Ver. 2.0 or later only	General-purpose DM words The contents of this file are automatically transferred to the EM Area beginning at E□_00000 when power is turned ON. □ indicates the bank number.	

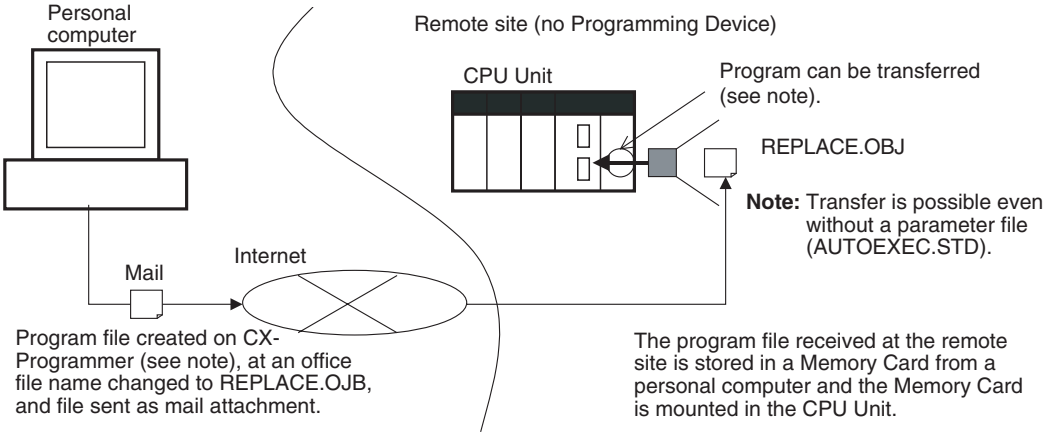
Note

1. If the program file name is REPLACE.OBJ (CS/CJ-series CPU Unit Ver. 2.0 or later only), the parameter area file will not be transferred even if it is on the Memory Card and regardless of the name of the parameter area file.
2. When creating the REPLACE.IOM, REPLCDM.IOM, or REPLCE□.IOM file from a Programming Device (Programming Console or CX-Programmer), always specify the proper first address (D20000, D00000, or E□_00000). The contents of the file will always be transferred starting at the proper first address (D20000, D00000, or E□_00000) even if another starting word is specified, which could result in the wrong data overwriting the contents of that part of the DM Area or EM bank.
3. If DIP switch pin 7 is turned ON and pin 8 is turned OFF to use the simple backup function, the simple backup function will take precedence even if pin 2 is also ON. In this case, the BACKUP□□ files will be transferred to the CPU Unit but the automatic transfer at startup files will not be transferred.

Application Example

Automatically Transferring Files without a Parameter Area File

A program file (.OBJ) can be created offline in a office and transferred to a remote location without a parameter area file (.STD). The program file can be stored in a Memory Card at the remote site without using a Programming Device and the Memory Card can be used to automatically transfer the program to the CPU Unit at startup.



Supported File Transfer Combinations

The following tables list whether files are automatically transferred to the CPU Unit at startup depending on which files are present on the Memory Card.

■ Program File: AUTOEXEC.OBJ

Program file	Parameter area file	Data files	Transferred/ Not transferred
AUTOEXEC.OBJ	AUTOEXEC.STD	One or more of the following: AUTOEXEC.IOM, ATEXECDM.IOM, ATEXECE□.IOM	Transferred.
		None	
	None	One or more of the following: AUTOEXEC.IOM, ATEXECDM.IOM, ATEXECE□.IOM	Not transferred.
		None	

■ Program File: REPLACE.OBJ

Program file	Parameter area file	Data files	Transferred/ Not transferred
REPLACE.OBJ	Present	One or more of the following: REPLACE.IOM, REPLCDM.IOM, REPLCE□.IOM	Transferred, but parameter area file is not transferred.
		None	
	None	One or more of the following: REPLACE.IOM, REPLCDM.IOM, REPLCE□.IOM	Transferred.
		None	

■ No Program File

Program file	Parameter area file	Data files	Transferred/ Not transferred
None	AUTOEXEC.STD	One or more of the following: AUTOEXEC.IOM, ATEXECDM.IOM, ATEXECE□.IOM	Not transferred.
		One or more of the following: REPLACE.IOM, REPLCDM.IOM, REPLCE□.IOM	
		None	
	None	One or more of the following: AUTOEXEC.IOM, ATEXECDM.IOM, ATEXECE□.IOM	
		One or more of the following: REPLACE.IOM, REPLCDM.IOM, REPLCE□.IOM	

■ Both AUTOEXEC and REPLACE Files

More Than One Program File

Program files		Parameter area file	Data files	Transferred/ Not transferred
AUTOEXEC.OBJ	REPLACE.OBJ	AUTOEXEC.STD	One or more of the following: AUTOEXEC.IOM, ATEXECDM.IOM, ATEXECE□.IOM	Not transferred.
			One or more of the following: REPLACE.IOM, REPLCDM.IOM, REPLCE□.IOM	
			None	
		None	One or more of the following: AUTOEXEC.IOM, ATEXECDM.IOM, ATEXECE□.IOM	
			One or more of the following: REPLACE.IOM, REPLCDM.IOM, REPLCE□.IOM	

More Than One Type of Data File

Program file	Parameter area file	Data files		Transferred/ Not transferred
AUTOEXEC.OBJ	AUTOEXEC.STD	One or more of the following: AUTOEXEC.IOM, ATEXECDM.IOM, ATEXECE□.IOM	One or more of the following: REPLACE.IOM, REPLCDM.IOM, REPLCE□.IOM	The following data files are transferred: AUTOEXEC.IOM, ATEXECDM.IOM, ATEXECE□.IOM
	None			Not transferred.
REPLACE.OBJ	Ignored			The following data files are transferred: REPLACE.IOM, REPLCDM.IOM, REPLCE□.IOM

Procedure

1,2,3...

1. Turn OFF the PLC power supply.
2. Turn ON DIP switch pin 2 on the front panel of the CPU Unit. Be sure that pins 7 is OFF.

   Note The simple backup function will take precedence over the automatic transfer at startup function, so be sure that pins 7 is OFF.
3. Prepare a Memory Card as follows:
   a) Transferring with a Parameter Area File
   Insert a Memory Card containing the user program file (AUTOEXEC.OBJ), parameter area file (AUTOEXEC.STD), and/or the I/O memory files (AUTOEXEC.IOM, ATEXECDM.IOM, and ATEXECE□.IOM) created with a CX-Programmer. (The program file and parameter area file must be on the Memory Card. The I/O memory files are optional.)
   b) Transferring without a Parameter Area File
   Insert a Memory Card containing the user program file (REPLACE.OBJ) and/or the I/O memory files (REPLACE.IOM, REPLCDM.IOM, and REPLCE□.IOM) created with a CX-Programmer. (The program file must be on the Memory Card. The I/O memory files are optional.)
4. Turn ON the PLC power supply.

Note Automatic Transfer Failure at Startup
If automatic transfer fails at startup, a memory error will occur, A40115 will turn ON, and the CPU Unit will stop. If an error occurs, turn OFF the power to clear the error. (The error cannot be cleared without turning OFF the power.)

DIP Switch on the Front Panel of the CPU Unit

Pin(s)	Name	Setting
2	Automatic transfer at startup pin	ON: Execute automatic transfer at startup. OFF: Do not execute automatic transfer at startup.
7	Used for simple backup operations	Turn OFF.
8	---	Always OFF.

Related Auxiliary Bits/Words

Name	Address	Setting
Memory Error Flag (Fatal error)	A40115	ON when an error occurred in memory or there was an error in automatic transfer from the Memory Card when the power was turned on (automatic transfer at start-up). The CPU Unit will stop and the ERR/ALM indicator on the front of the CPU Unit will light. **Note:** A40309 will be turned ON if the error occurred during automatic transfer at startup. (The error cannot be cleared in this case.)
Memory Card Start-up Transfer Error Flag	A40309	ON when automatic transfer at start-up has been selected and an error occurs during automatic transfer (DIP switch pin 2 ON). An error will occur if there is a transfer error, the specified file does not exist, or the Memory Card is not installed. **Note:** The error can be cleared by turning the power off. (The error cannot be cleared while the power is on.)

Precautions when Changing I/O Allocation Status during Automatic Transfer at Startup

The I/O allocation status depends on the unit versions of the source and destination CPU Units when using a single CJ-series CPU Unit to create parameter files for automatic transfer at startup, save them in the Memory Card, and then automatically transfer them to another CJ-series CPU Unit at startup. The changes to I/O allocation status for different unit version combinations is shown in the following table.

Source CPU Unit		Original I/O allocation status	Destination CPU Unit		
			Unit version of CPU Unit to which files for automatic transfer at startup will be sent		
			Pre-Ver. 2.0	Unit Ver. 2.0	Unit Ver. 3.0 or later
CPU Unit's unit version used to create files for automatic transfer at startup	Pre-Ver. 2.0	Automatic allocation	Switches to user-specified	(Same) automatic allocation	Switches to user-specified
		User-specified	(Same) User-specified	Switches to automatic allocation (See note 1.)	(Same) User-specified
	Unit Ver. 2.0	Automatic allocation	Switches to user-specified	(Same) automatic allocation	Switches to user operation (See note 2.)
		User-specified	(Same) User-specified	(Same) User-specified	(Same) User-specified
	Unit Ver. 3.0 or later	Automatic allocation	Switches to user-specified	(Same) Automatic allocation	(Same) Automatic allocation
		User-specified	(Same) User-specified	(Same) User-specified	(Same) User-specified

Note

1. When files for automatic transfer at startup (AUTOEXEC.STD) are created and saved in a Memory Card using user-specified I/O allocations with a pre-Ver. 2.0 CJ-series CPU Unit, the system will automatically switch to automatic I/O allocation at startup if the data is automatically transferred from the Memory Card.
2. When files for automatic transfer at startup (AUTOEXEC.STD) are created and saved in the Memory Card using a CJ-series CPU Unit with unit version 2.0, the I/O allocation status will switch automatically to user-specified I/O allocations if the data is automatically transferred from the Memory Card to a CJ-series CPU Unit with unit version 3.0 or later.

Precautions when Comparing Automatic Transfer at Startup Parameter Files

Verification errors may occur at the Programming Console when comparing parameter data between files before transfer and the data after transfer when creating automatic transfer at startup parameter files (AUTOEXEC.STD) and executing automatic transfer at startup for combinations of pre-Ver. 2.0, unit version 2.0, and unit version 3.0 CJ-series CPU Units.

Source CPU Unit		Original i/O allocation status	Destination CPU Unit		
			Unit version of CPU Unit at backup/restore destination		
			Pre-Ver. 2.0	Unit Ver. 2.0 or later	Unit Ver. 3.0 or later
Unit version of CPU Unit at backup source	Pre-Ver. 2.0	Automatic allocation	Verification possible	Verification possible	Verification possible
		User-specified			Verification possible
	Unit Ver. 2.0 or later	Automatic allocation			Verification possible
		User-specified			Verification possible
	Unit Ver. 3.0 or later	Automatic allocation		Verification error	Verification possible
		User-specified		Verification possible	

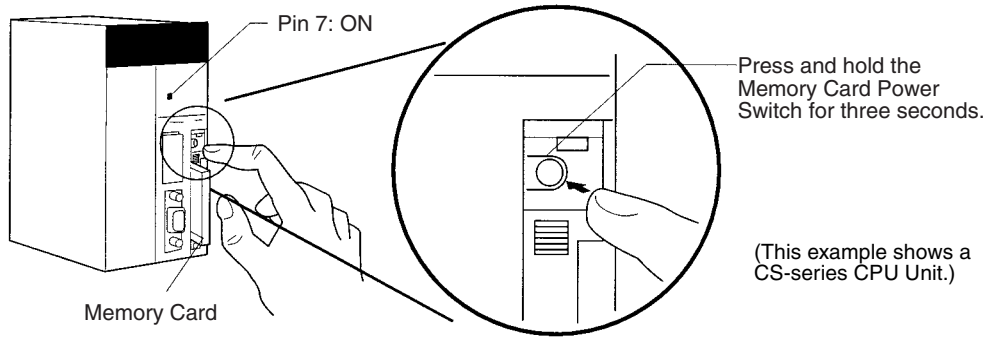
5-2-6 Simple Backup Function

This function is not supported by CS-series CS1 CPU Units that are pre-EV1.

Backing Up Data from the CPU Unit to the Memory Card

To backup data, turn ON pin 7 on the CPU Unit's DIP switch, press and hold the Memory Card Power Supply Switch for three seconds. The backup function will automatically create backup files with fixed file names and extensions, and write them to the Memory Card. The backup files contain the program, parameter area data, I/O memory data, symbol tables (see note), comment files (see note), and program index files (see note). This function can be executed in any operating mode.

Note This data is supported by CS/CJ-series CPU Units with unit version 3.0 or later only. The backup files are created automatically from files in either the Memory Card, EM file memory, or comment memory.



Restoring Data from the Memory Card to the CPU Unit

To restore the backup files to the CPU Unit, check that pin 7 is ON and turn the PLC's power OFF and then ON again. The backup files containing the program, parameter area data, I/O memory data, symbol tables (see note), comment files (see note), and program index files (see note) will be read from the Memory Card to the CPU Unit all at the same time.

*1 This data is supported by CS/CJ-series CPU Units with unit version 3.0 or later only. The backup files are read to either the Memory Card, EM file memory, or comment memory.

Note

1. The backup function will override the automatic transfer at startup function, so the backup files will be read to the CPU Unit when the PLC is turned ON even if pin 2 of the DIP switch is ON.
2. Data will not be read from the Memory Card to the CPU Unit if pin 1 of the DIP switch is ON (write-protecting program memory).
3. When the backup files are read from the Memory Card by the backup function, the status of I/O memory and force-set/force-reset bits will be cleared unless the necessary settings are made in the Auxiliary Area and PLC Setup.

   If the IOM Hold Bit (A50012) is ON and the PLC Setup is set to maintain the IOM Hold Bit Status at Startup when the backup files are written, the status of I/O memory data will be maintained when data is read from the Memory Card.

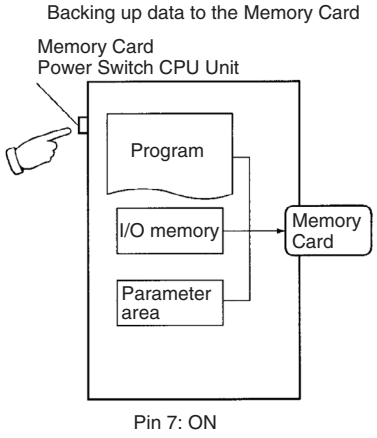
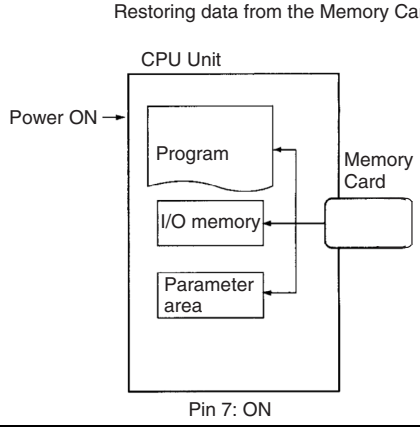
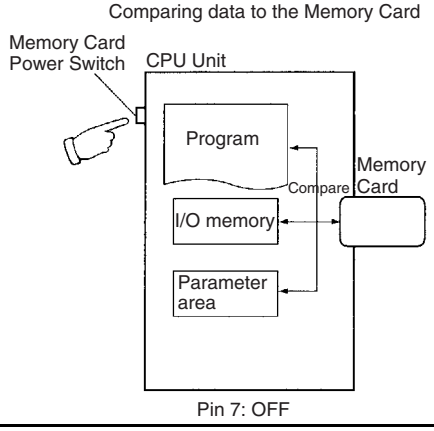
   If the Forced Status Hold Bit (A50013) is ON and the PLC Setup is set to maintain the Forced Status Hold Bit Status at Startup when the backup files are written, the status of force-set and force-reset bits will be maintained when data is read from the Memory Card.
4. A CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit will remain in PROGRAM mode after the simple backup operation has been performed and cannot be changed to MONITOR or RUN mode until the power supply has been cycled. After completing the backup operation, turn OFF the power supply to the CPU Unit, changes the settings of pin 7, and then turn the power supply back ON.
5. File backups may take from several seconds to several minutes. Refer to page 256 for information on execution times.

Comparing Data in the Memory Card and CPU Unit

To compare the backup files in the Memory Card with the data in the CPU Unit, turn OFF pin 7 on the CPU Unit's DIP switch, and press and hold the Memory Card Power Supply Switch for three seconds. The backup function will compare the program, parameter area data, I/O memory data, symbol tables (see note), comment files (see note), and program index files (see note) in the Memory Card with the corresponding data in the CPU Unit. This function can be executed in any operating mode.

Note This data is supported by CS/CJ-series CPU Units with unit version 3.0 or later only.

The following table provides a summary of the simple backup operations.

Backup operation	Pin status	Procedure
	Pin 7	
Backing up data from the CPU Unit to the Memory Card 	ON	Press and hold the Memory Card Power Switch for three seconds.
Restoring data from the Memory Card to the CPU Unit 	ON	Turn the PLC OFF and ON again. (See note 1.)
Comparing data between the CPU Unit and the Memory Card 	OFF	Press and hold the Memory Card Power Switch for three seconds.

Note

1. Refer to *Verifying Backup Operations with Indicators* on page 244 for details on the results of read, write, and compare operations.
2. Refer to *5-3-2 Operating Procedures* for guidelines on the time required for Memory Card backup operations.

Backup Files

Data Files

File name and extension	Data area and range of addresses stored		Backup from CPU Unit to Memory Card (creating files)	Restore from Memory Card to CPU Unit	Comparing Memory Card to CPU Unit		Files required when restoring data
CPU Unit	**CS/CJ**				**CS1/ CJ1**	**CS1-H/ CJ1-H**	
BACKUP.IOM	DM	D20000 to D32767	Yes	Yes	Yes	---	Required in Memory Card
BACKUPIO.IOR	CIO	0000 to 6143 (Including forced bit status.)	Yes	---[4]	Yes	---	Required in Memory Card
	WR	W000 to W511 (Including forced bit status.)	Yes	---[4]	Yes	---	
	HR	H000 to H511	Yes	Yes	Yes	---	
	AR	A000 to A447	Yes	---	---	---	
		A448 to A959	Yes	Yes	Yes	---	
	Timer[1]	T0000 to T4095	Yes	Yes[4]	Yes	---	
	Counter[1]	C0000 to C4095	Yes	Yes	Yes	---	
BACKUPDM.IOM	DM	D00000 to D19999	Yes	Yes	Yes	---	Required in Memory Card
BACKUPE□.IOM[2,3]	EM	E□_00000 to E□_32767	Yes	Yes	Yes	---	Required in Memory Card (must match CPU Unit)

Note

1. The Completion Flags and PVs are backed up.
2. The □ represents the bank number and the number of banks depends upon the CPU Unit being used.

   When the BACKUPE□.IOM files in the Memory Card are restored to the CPU Unit, the files are read in order beginning with bank 0 and ending with the maximum bank number in the CPU Unit. Excess BACKUPE□.IOM files will not be read if the number of banks backed up exceeds the number of banks in the CPU Unit. Conversely, any remaining EM banks in the CPU Unit will be left unchanged if the number of banks backed up is less than the number of banks in the CPU Unit.

   If a BACKUPE□.IOM file is missing (for example: 0, 1, 2, 4, 5, 6), only the consecutive files will be read. In this case, data would be read to banks 0, 1, and 2 only.
3. The EM Area data will be backed up as binary data. EM banks that have been converted to file memory will be backed up along with EM banks that have not.

   EM file memory can be restored to another CPU Unit's EM Area only if the BACKUPE□.IOM files are consecutive and the number of backed-up EM banks matches the number of banks in the CPU Unit. If the BACKUPE□.IOM files are not consecutive or the number of EM banks does not match the number of banks in the CPU Unit, the EM file memory will revert to its unformatted condition and the files in file memory will be invalid. (The regular EM Area banks will be read normally.)
4. Normally, the contents of the CIO Area, WR Area, Timer Completion Flags, Timer PVs, and the status of force-set/force-reset bits will be cleared when the PLC is turned ON and BACKUPIO.IOR is read from the Memory Card.

If the IOM Hold Bit (A50012) is ON and the PLC Setup is set to maintain the IOM Hold Bit Status at Startup when the backup files are written, the status of I/O memory data will be maintained when data is read from the Memory Card.

If the Forced Status Hold Bit (A50013) is ON and the PLC Setup is set to maintain the Forced Status Hold Bit Status at Startup when the backup files are written, the status of force-set and force-reset bits will be maintained when data is read from the Memory Card.

Program Files

File name and extension	Contents	Backup from CPU Unit to Memory Card (creating files)	Restore from Memory Card to CPU Unit	Comparing Memory Card to CPU Unit	Files required when restoring data
CPU Unit	**CS/CJ**				
BACKUP.OBJ	Entire user program	Yes	Yes	Yes	Required in Memory Card

Parameter Files

File name and extension	Contents	Backup from CPU Unit to Memory Card (creating files)	Restore from Memory Card to CPU Unit	Comparing Memory Card to CPU Unit	Files required when restoring data
CPU Unit	**CS/CJ**				
BACKUP.STD	PLC Setup Registered I/O tables Routing tables CPU Bus Unit setup Etc.	Yes	Yes	Yes	Required in Memory Card

Unit/Board Backup Files

File name and extension	Contents	Backup from CPU Unit to Memory Card (creating files)	Restore from Memory Card to CPU Unit	Comparing Memory Card to CPU Unit	Files required when restoring data
CPU Unit	**CS1-H, CS1D, CJ1-H, CJ1M, or CS1D CPU Unit only**				
BACKUP□□.PRM (where □□ is the unit address of the Unit/Board being backed up)	Backup data from the Unit or Board with the specified unit address (Specific contents depends on the Unit or Board.)	Yes	Yes	Yes	Required in Memory Card (See note 2.)

Note 1. Unit addresses are as follows:
CPU Bus Units: Unit number + 10 Hex
Special I/O Units: Unit number + 20 Hex
Inner Board: E1 Hex

2. An error will not occur in the CPU Unit even if this file is missing when data is transferred from the Memory Card to I/O memory, but an error will occur in the Unit or Board if the data is not restored. Refer to the operation manual for the specific Unit or Board for details on Unit or Board errors.

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support the Unit/Board backup function.

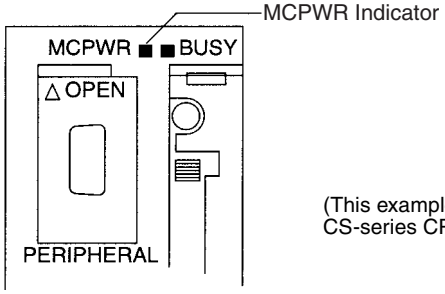
Symbol Tables, Comment Files, Program Index Files (CS1-H/CJ1-H, CJ1M, CS1D CPU Units with Unit Version 3.0 or Later Only)

File name and extension	Contents	Backup from CPU Unit to Memory Card (creating files)	Restore from Memory Card to CPU Unit	Comparing Memory Card to CPU Unit	Files required when restoring data
CPU Unit	**CS1-H, CJ1-H, CJ1M, or CS1D CPU Unit with unit version 3.0 only**				
BKUPSYM.SYM	Symbol table files	Yes (writes if CPU Unit comment memory contains files)	Yes (reads to CPU Unit comment memory if Memory Card contains files)	Yes (compares with files contained in CPU Unit's comment memory)	These files are not absolutely required in the Memory Card.
BKUPCMT.CMT	Comment files				
BKUPPRG.IDX	Program index files				

Note The backup, restore, and comparison operations in the above table are performed only for files contained in comment memory.

Verifying Backup Operations with Indicators

The status of the Memory Card Power (MCPWR) indicator shows whether a simple backup operation has been completed normally or not.



(This example shows a CS-series CPU Unit.)

Backup operation	Normal completion (See note 1.)	Error occurred	
	MCPWR status	**MCPWR status**	**Error**
Backing up data from the CPU Unit to the Memory Card	Lit → Remains lit while the Memory Card Power Switch is pressed. → Flashes once. → Lit while writing. → OFF after data is written.	Lit → Remains lit while the Memory Card Power Switch is pressed. → Remains flashing.	No files will be created with the following errors: Insufficient Memory Card capacity (See note 2.) Memory error in CPU Unit I/O bus error (when writing data to a Unit or Board, CS1-H, CS1D, or CJ1-H CPU Units only)
Restoring data from the Memory Card to the CPU Unit	Lit when power is turned ON. → Flashes once. → Lit while reading. → OFF after data is read.	Lit when power is turned ON. → Flashes five times. → Goes OFF.	Data won't be read with the following errors: Program in Memory Card exceeds CPU Unit capacity Required backup files do not exist in Memory Card. Program can't be written because it is write-protected (Pin 1 of the DIP switch is ON.)
		Lit when power is turned ON. → Flashes once. → Lit while reading. → Flashes three times. → OFF after data is read.	Caution: Data will be read with the following error. EM files and CPU Unit EM banks do not match (non-consecutive bank numbers or max. bank number mismatch).

Backup operation	Normal completion (See note 1.)	Error occurred	
	MCPWR status	MCPWR status	Error
Comparing data between the CPU Unit and the Memory Card	Lit → Remains lit while the Memory Card Power Switch is pressed. → Flashes once. → Lit while comparing. → OFF after data is compared.	Lit → Remains lit while the Memory Card Power Switch is pressed. → Remains flashing.	The following comparison errors can occur (See note 3.): Memory Card and CPU Unit data do not match. Required backup files do not exist in Memory Card. EM files and CPU Unit EM banks do not match (non-consecutive bank numbers or max. bank number mismatch). Memory error in CPU Unit I/O bus error (when comparing data to a Unit or Board, CS1-H, CS1D, or CJ1-H CPU Units only)
Common to all three backup operations.	---	Reading: Flashes five times. → Goes OFF. Writing or comparing: Remains flashing.	Memory Card access error (format error or read/write error)

Note

1. When the backup operation is completed normally, power to the Memory Card will go OFF when the MCPWR indicator goes OFF. If the Memory Card will be used again, press the Memory Card Power Switch to supply power and execute the desired operation.
2. When data is written for a simple backup operation, errors for insufficient Memory Card capacity can be checked in A397 (Simple Backup Write Capacity). If A397 contains any value except 0000 Hex after the write operation has been executed, the value will indicate the capacity that is required in the Memory Card in Kbytes.
3. The backup files for Units and Boards are also compared.

   Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support the Unit/Board backup function.

Related Auxiliary Bits/Words

Name	Address	Description
File Memory Operation Flag	A34313	ON when any of the following are being performed. OFF when execution has been completed. • Memory Card detection • CMND instruction executed for local CPU Unit • FREAD/FWRIT instructions • Program replacement via special control bits • Simple backup operation Wiring data to or verifying the contents of the Memory Card is not possible while this flag is ON.
EM File Memory Starting Bank	A344	When the CPU Unit starts reading from the Memory Card, it references this value. If the maximum EM bank number of the BACKUPE□.IOM files (maximum consecutive bank number counting from 0) matches the maximum bank number of the CPU Unit, the EM area will be formatted based on the value in this word. If the maximum EM bank numbers do not match, the EM Area will revert to its unformatted condition.
Network Communications Instruction Enabled Flags (See note.)	A20200 to A20207	• Turns OFF when writing or comparing Memory Card data begins. • Turn ON when writing or comparing Memory Card data has been completed. Unit and Board data cannot be written or compared if all of the Network Communications Instruction Enabled Flags are OFF when Memory Card write or compare operations are started and an error will occur if this is attempted.
Network Communications Completion Code (See note.)	A203 to A210	Provide the results of communications with the Unit or Board when Memory Card write or compare operations are performed.
Network Communications Error Flags (See note.)	A21900 to A21907	• Turns ON is an error occurs in communications with the Unit or Board when Memory Card write or compare operations are performed. • Remains OFF (or turns OFF) is no error occurs in communications with the Unit or Board when Memory Card write or compare operations are performed.
Simple Backup Write Capacity (See note.)	A397	Provides the data capacity in Kbytes that would be required on the Memory Card when writing fails for a simple backup operation, indicating that a write error occurred because of insufficient capacity. 0001 to FFFF Hex: Write error (Indicates required Memory Card capacity between 1 and 65,535 Kbytes.) (Cleared to 0000 Hex when successful write is performed.) 0000 Hex: Write completed normally.

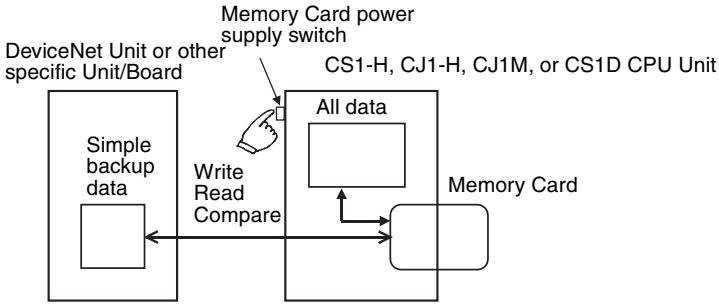
Note When a simple backup operation is performed, an available communications port will be automatically found and used. These network communications flags and words are thus related. The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support the Unit/Board backup function, so they do not use these Auxiliary Area words and flags.

Backing Up Board and Unit Data to Memory Cards

The following data in the CPU Unit is backed up by the simple backup operation: the user program, parameter area, and all of the data areas in I/O memory. The simple backup operation also backs up data from certain Special I/O Units, CPU Bus Units, and Inner Boards mounted to the PLC.

The data in each Unit or Board is backed up separately for each Unit or Board.

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units do not support the Unit/Board backup function.



Application

This function can be used to back up data for the entire PLC, including the CPU Unit, DeviceNet Units, Serial Communications Units/Boards, etc. It can also be used for Unit replacement.

Unit/Board Backup Files

The data from each Unit and Board is stored in the Memory Card using the following file names: BACKUP□□.PRM. Here, "□□" is the unit address of the Unit or Board in hexadecimal.

Note Unit addresses are as follows:
CPU Bus Units: Unit number + 10 hex
Special I/O Units: Unit number + 20 hex
Inner Board: E1 Hex

These files are also used when reading from the Memory Card or comparing Memory Card data.

Applicable Units and Boards

For Unit and Board data to be backed up, the Unit/Board must also support the backup function. Refer to the operation manual for the Unit/Board for details on support.

Unit/Board	Model numbers	Data backed up for simple backup when used with CS1-H/CJ1-H CPU Unit	Data capacity used in Memory Card for simple backup
Serial Communications Units	CS1W-SCU21-V1 CS1W-SCU31-V1 CJ1W-SCU21 CJ1W-SCU41 CJ1W-SCU21-V1 CJ1W-SCU31-V1 CJ1W-SCU41-V1	Protocol macro data (Including both standard system protocols and user-defined protocols from the flash memory in the Unit or Board)	129 Kbytes
Serial Communications Boards	CS1W-SCB21-V1 CS1W-SCB41-V1		129 Kbytes
FL-net Units	CS1W-FLN22 CJ1W-FLN22	Data link tables	1.3 Kbytes
DeviceNet Units	CJ1W-DRM21-V1 CJ1W-DRM21	Device parameters (all data in EEPROM in the Unit) (Although this is the same data as is backed up from the Memory Card backup function supported by the Unit or the DeviceNet Configuration (Ver. 2.0), there is no file compatibility.)	7 Kbytes

Unit/Board	Model numbers	Data backed up for simple backup when used with CS1-H/CJ1-H CPU Unit	Data capacity used in Memory Card for simple backup
CompoNet Units	CS1W-CRM21 CJ1W-CRM21	• Registration tables • Software settings • Network parameters • Slave parameters	3.5 Kbytes
Customizable Counter Units	CS1W-HIO01-V1 CS1W-HCP22-V1 CS1W-HCA22-V1 CS1W-HCA12-V1	• User program • General-purpose read-only DM • Unit function setting area • Information on expansion instructions • Ladder library	64 Kbytes
Motion Control Units	CS1W-MCH71 CJ1W-MCH71	• Positioning data • System parameters • G language programs	8,192 Kbytes
	CS1W-MC221-V1 CS1W-MC421-V1		142 Kbytes
Position Control Units	CJ1W-NCF71	• Common parameters • Individual axis parameters	64 Kbytes
Position Control Units	CS1W-NC113/133/213/233/413/433 Ver. 2.0 or later CJ1W-NC113/133/213/233/413/433 Ver. 2.0 or later	• Axis parameters • Sequence data • Speed data • Acceleration/deceleration time data • Dual timer data • Zone data	7 Kbytes
Loop Control Board	CS1W-LCB01 CS1W-LCB05	• Function block data • Block connection data	1,563 Kbytes

Note Data from the Units and Boards listed above will be automatically backed up for the simple backup operation. There is no setting available to include or exclude them. If a Programming Console is used, however, operations are supported individually for the user program area, parameter area, and I/O memory areas. Refer to the *Programming Console Operation Manual* (W314) for details.

Procedure

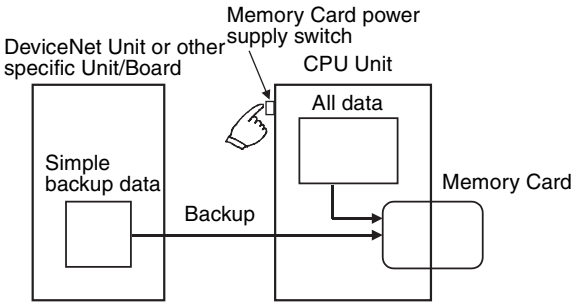
The procedure for the simple backup operation is the same regardless of whether or not data is being backed up from specific Units and Boards (including writing, reading, and comparing).

■ Backing Up Data

1,2,3... 1. Turn ON pin 7 on the CPU Unit's DIP switch.

2. Press and hold the Memory Card Power Supply Switch for three seconds.

The backup data for the Units and Boards will be created in a file and stored in the Memory Card with the other backup data.



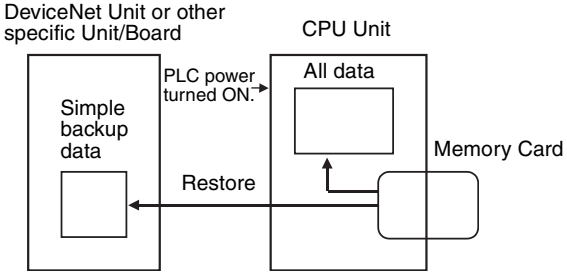
When the Power Supply Switch is pressed, the MCPWR Indicator will flash once, light during the write operation, and then go OFF if the write is completed normally.

■ Restoring Data

1,2,3...

1. Turn ON pin 7 on the CPU Unit's DIP switch.
2. Turn ON the PLC. The backup files will be restored to the Units and Boards.

   The backup data for the Units and Boards will be restored from the Memory Card to the Units and Boards.



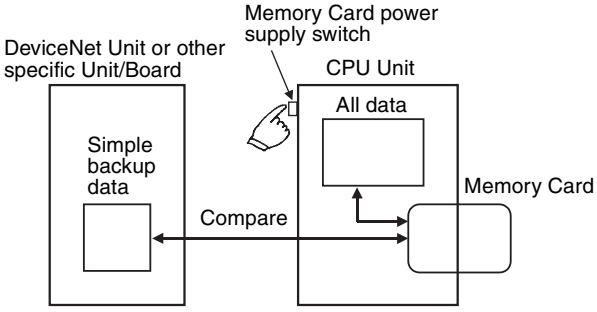
When the power supply is turned ON, the MCPWR Indicator will flash once, light during the read operation, and then go OFF if the read is completed normally.

■ Comparing Data

1,2,3...

1. Turn OFF pin 7 on the CPU Unit's DIP switch.
2. Press and hold the Memory Card Power Supply Switch for three seconds.

   The backup data on the Memory Card will be compared to the data in the Units and Boards.



When the Power Supply Switch is pressed, the MCPWR Indicator will flash once, light during the compare operation, and then go OFF if the compare is completed normally and the data is the same.

Note When the CS1W-SCU21-V1 is being used, the time required for a simple backup operation will be longer than when the CS1W-SCU21-V1 is not being used by the times given in the following tables.

Additional Time when the CPU Bus Unit Settings File (BACKUP□□.PRM) on the Memory Card Is 60 Kbytes

Operating mode	Additional time when writing to a Memory Card	Additional time when verifying a Memory Card	Additional time when reading from a Memory Card
PROGRAM	Approx. 25 s	Approx. 10 s	Approx. 4 s
RUN	Approx. 1 min 30 s	Approx. 30 s	Approx. 4 s

Additional Time when the CPU Bus Unit Settings File (BACKUP□□.PRM) on the Memory Card Is 128 Kbytes

Operating mode	Additional time when writing to a Memory Card	Additional time when verifying a Memory Card	Additional time when reading from a Memory Card
PROGRAM	Approx. 40 s	Approx. 14 s	Approx. 8 s
RUN	Approx. 2 min 30 s	Approx. 1 min	Approx. 8 s

Note

1. Confirm that the Units and Boards that are being backed up are running properly before attempting the above operations. The write, read, and compare operations will not be performed unless the Units and Boards are running properly.
2. Before performing a simple backup operation for specific Units/Boards, make sure either that the CPU Unit is in PROGRAM mode or that performing the simple backup operation will not adversely affect instructions that use communications port numbers. When data is backed up from specific Units/Boards, a communications port will be searched for beginning from port 0 and the first available port will be used. If the port number is the same as one used by a network communications instruction, the network communications instruction will not be executed until the simple backup operation has been completed.

Precautions When Changing the I/O Allocation Status During Backup/Restore Operations

The I/O allocation status depends on the unit versions of the source and destination CPU Units when using a single CJ-series CPU Unit to create backup parameter files (BKUP.STD), save them in the Memory Card, and then back up or restore them to another CJ-series CPU Unit. The changes to I/O allocation status for different unit version combinations are shown in the following table.

Source CPU Unit		Original I/O allocation status	Destination CPU Unit		
			Unit version of CPU Unit at the backup/restore destination		
			Pre-Ver. 2.0	Unit Ver. 2.0	Unit Ver. 3.0 or later
Unit version of CPU Unit at the backup source	Pre-Ver. 2.0	Automatic allocation	(Same) automatic allocation		
		User-specified	(Same) User-specified	(Same) automatic allocation	(Same) User-specified
	Unit Ver. 2.0	Automatic allocation	(Same) automatic allocation		
		User-specified	(Same) User-specified		
	Unit Ver. 3.0 or later	Automatic allocation	(Same) automatic allocation		
		User-specified	(Same) User-specified		

Precautions When Mismatch Occurs During Backup Comparison

When using a CJ-series CPU Unit with unit version 2.0 or earlier with a CJ-series CPU Unit with unit version 3.0, verification errors may occur when comparing parameter data with the backup data restored from the simple backup file that was created.

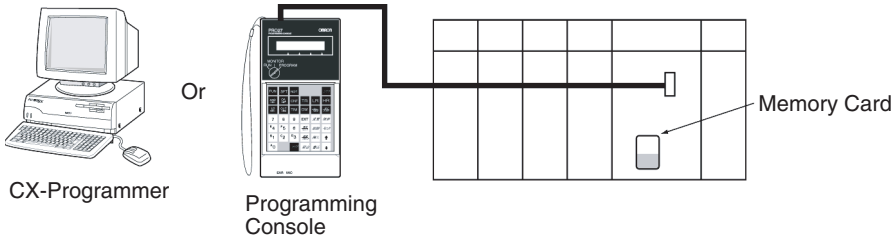
Source CPU Unit		Previous I/O allocation status	Destination CPU Unit		
			Unit version of CPU Unit at the backup/restore destination		
			Pre-Ver. 2.0	Unit Ver. 2.0	Unit Ver. 3.0 or later
Unit version of CPU Unit at the backup source	Pre-Ver. 2.0	Automatic allocation	Verification possible	Verification possible	Verification possible
		User-specified			
	Unit Ver. 2.0	Automatic allocation	Verification possible		Verification possible
		User-specified			
	Unit Ver. 3.0 or later	Automatic allocation	Verification error	Verification error	Verification possible
		User-specified	Verification possible	Verification possible	

5-3 Using File Memory

5-3-1 Initializing Media

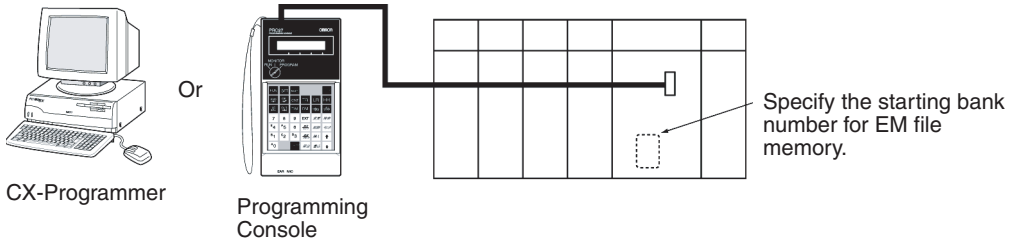
Memory Cards

1,2,3... 1. Use a Programming Device, such as a Programming Console, to initialize Memory Cards.

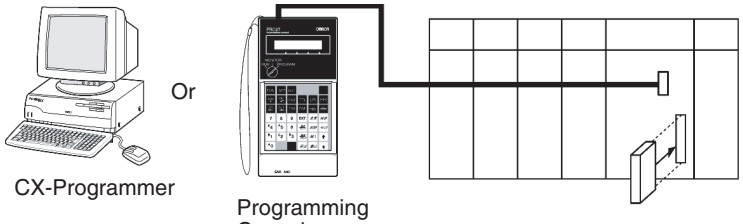


EM File Memory

1,2,3... 1. Use a Programming Device like a Programming Console and set EM file memory settings in the PLC Setup to enable EM file memory, and then set the specified bank number for EM file memory to 0 to C Hex.



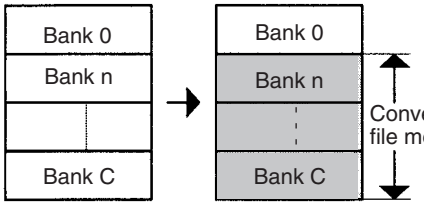
2. Initialize EM file memory.



Initializing Individual EM File Memory

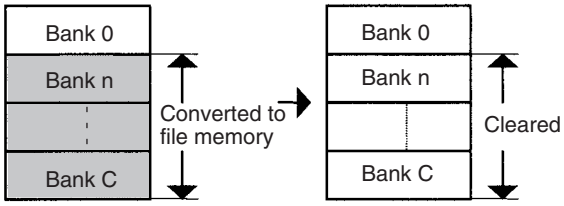
A specified EM bank can be converted from ordinary EM to file memory.

Note The maximum bank number for CJ-series CPU Units is 6.



1. Set n in PLC Setup.
2. Use a Programming Device or FINS command to format starting at n.
3. "n" is stored in A344.

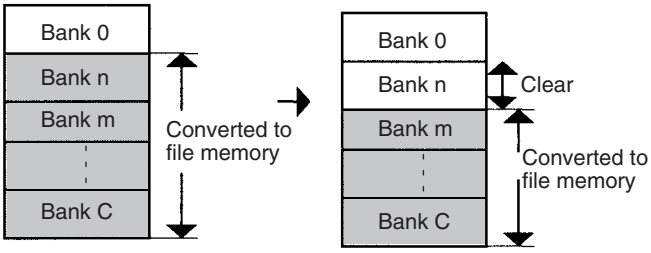
EM used for file memory can be restored to ordinary EM status.



1. Set file memory OFF in PLC Setup.
2. If a Programming Device or FINS command is used for formatting, memory starting at n will be cleared to 0000 Hex.
3. FFFF Hex will be stored in A344 to indicate that there is no EM file memory.

Note: Any file data present will be deleted at this time.

The start bank number for file memory can be changed.



1. Change n to m in PLC Setup.
2. Use a Programming Device or FINS command to convert banks starting at m to file memory.

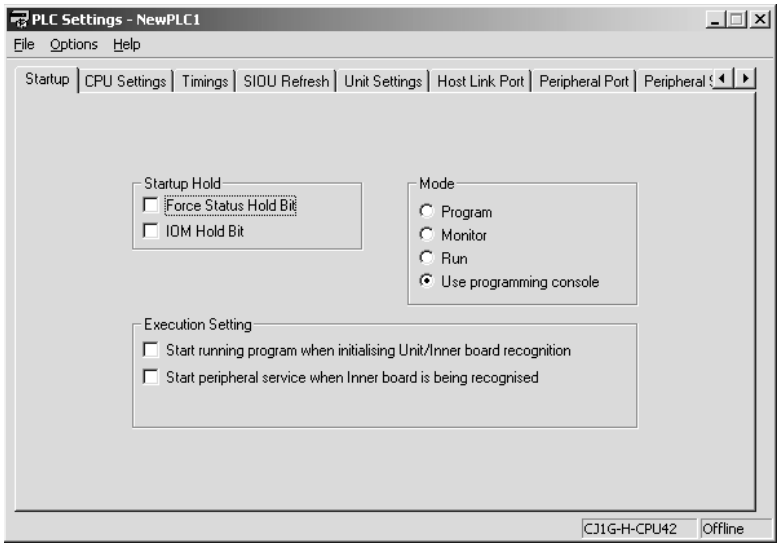
Note: Banks n to m-1 will be cleared to 0000 Hex.

3. m will be stored in A344.

Note: Any file data present will be deleted at this time.

PLC Setup

When using the CX-Programmer, make the settings on the CPU Settings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address	Name	Description	Initial setting
+136	EM File Memory Starting Bank	0000 Hex: None 0080 Hex: Starting at bank No. 0 008C hex: Bank No. C The EM area starting from the specified bank number will be converted to file memory.	0000 Hex

Related Special Auxiliary Relay

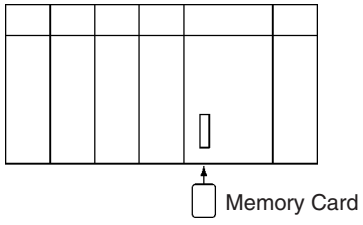
Name	Address	Description
EM File Memory Starting Bank	A344	The bank number that actually starts the EM file memory area at that time will be stored. The EM file from the starting bank number to the last bank will be converted to file memory. FFFF Hex will indicate that there is no EM file memory.

5-3-2 Operating Procedures

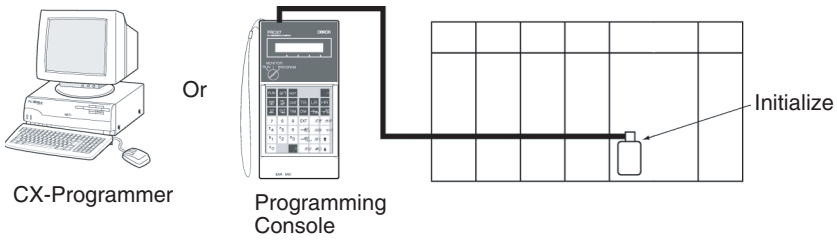
Memory Cards

Using a Programming Device

1,2,3... 1. Insert a Memory Card into the CPU Unit.



2. Initialize the Memory Card with a Programming Device.

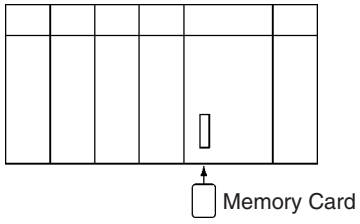


3. Use a Programming Device to name the CPU Unit data (user program, I/O memory, parameter area), and then save the data to Memory Card. (Use a Programming Device to read the Memory Card file to the CPU Unit.)

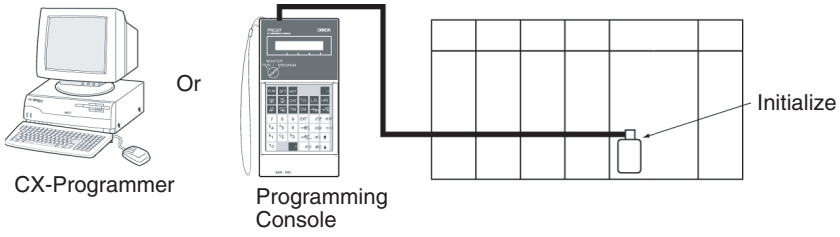
Automatically Transferring Files at Startup

Procedure when Transferring a Parameter Area File

1,2,3... 1. Insert an initialized Memory Card into the CPU Unit.

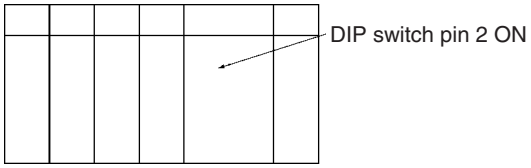


2. Use a Programming Device to write the automatic transfer at startup files to the Memory Card. These files include the program file (AUTOEXEC.OBJ), parameter area file (AUTOEXEC.STD), and I/O memory files (AUTOEXEC.IOM, ATEXECDM.IOM and/or ATEXECE□.IOM).



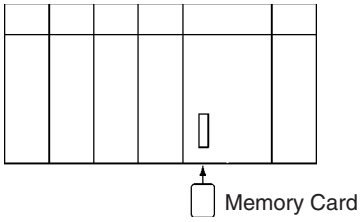
Note A user program and parameter area file must be on the Memory Card.

3. Turn OFF the PLC power supply.
4. Turn ON DIP switch pin 2 (automatic transfer at startup).



Note If pin 7 is ON, the backup function will be enabled and will override the automatic transfer at startup function. Turn OFF pins 7 and 8 for automatic transfer at startup.

5. Insert the Memory Card into the CPU Unit.



6. Turn ON the PLC power supply to read the file.

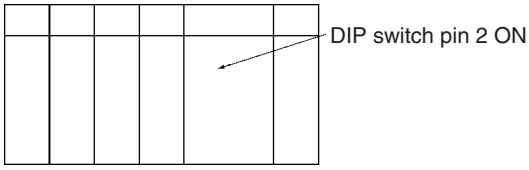
Procedure When Not Transferring a Parameter Area File

1,2,3...

1. Insert an initialized Memory Card into the CPU Unit.
2. Use a Programming Device to write the automatic transfer at startup files to the Memory Card. These files include the program file (REPLACE.OBJ) and I/O memory files (REPLACE.IOM, REPLCDM.IOM, and/or REPLCDE□.IOM).

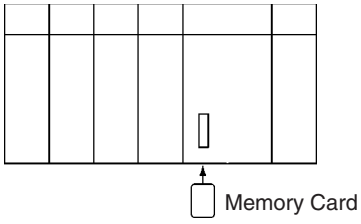
Note A parameter area will not be transferred even if one is on the Memory Card.

3. Turn OFF the PLC power supply.
4. Turn ON DIP switch pin 2 (automatic transfer at startup).



Note If pin 7 is ON, the backup function will be enabled and will override the automatic transfer at startup function. Turn OFF pins 7 and 8 for automatic transfer at startup.

5. Insert the Memory Card into the CPU Unit.



6. Turn ON the PLC power supply to read the file.

Using FREAD(700)/FWRIT(701)/CMND(490)

1,2,3...

1. Insert a Memory Card into the CPU Unit. (Already initialized.)
2. Use FWRIT(701) to name the file in the specified area of I/O memory and then save the file to Memory Card.

Note A Memory Card containing TXT or CSV data files can be installed into a personal computer's PLC card slot with an HMC-AP001 Memory Card Adapter and the data files can be read into a spreadsheet program using standard Windows functions (Not supported by CS-series CS1 CPU Units that are pre-EV1).

3. Use FREAD(700) to read the file from the Memory Card to I/O memory in the CPU Unit.

Memory Card file operations can be executed by issuing FINS commands to the local CPU Unit with CMND(490). (Not supported by CS-series CS1 CPU Units that are pre-EV1)

Replacing the Program during Operation

1,2,3...

1. Insert a Memory Card into the CPU Unit. (Already initialized.)
2. Write the Program Password (A5A5 Hex) in A651 and the Program File Name in A654 to A657.
3. Turn the Replacement Start Bit (A65015) from OFF to ON.

Simple Backup Function

There are 3 backup operations: backing up data to the Memory Card, restoring data from the Memory Card, and comparing data with the Memory Card.

Backing Up Data from the CPU Unit to the Memory Card

1,2,3...

1. Insert a Memory Card into the CPU Unit. (Already initialized.)
2. Turn ON pin 7 on the CPU Unit's DIP switch.
3. Press and hold the Memory Card Power Supply Switch for three seconds.
4. Verify that the MCPWR Indicator flashes once and then goes OFF. (Other changes indicate that an error occurred while backing up the data.)

Restoring Data from the Memory Card to the CPU Unit

1,2,3...

1. Insert the Memory Card containing the backup files into the CPU Unit.
2. Turn ON pin 7 on the CPU Unit's DIP switch.
3. The backup files will be restored when the PLC is turned ON.
4. Verify that the MCPWR Indicator flashes once and then goes OFF. (Other changes indicate that an error occurred while restoring the data.)

Comparing Data in the Memory Card and CPU Unit

1,2,3...

1. Insert the Memory Card containing the backup files into the CPU Unit.
2. Turn OFF pin 7 on the CPU Unit's DIP switch.
3. Press and hold the Memory Card Power Supply Switch for three seconds.
4. The data matches if the MCPWR Indicator flashes once and then goes OFF.

Note The following table lists the benchmark execution times for the simple backup operation. These times were measured under the following conditions, with different PLC Unit configurations.
CPU Unit: Unit version 4.0
User program: 20 Ksteps
Cycle time in RUN mode: 10 ms

Configuration: CPU Unit only

Operating mode	Writing to Memory Card	Comparing with Memory Card	Reading from Memory Card
PROGRAM mode	Approx. 20 s	Approx. 15 s	Approx. 10 s
RUN mode	Approx. 1 min	Approx. 30 s	Approx. 10 s

Configuration: CPU Unit + 1 Serial Communications Unit (See note 1.)

Operating mode	Writing to Memory Card	Comparing with Memory Card	Reading from Memory Card
PROGRAM mode	Approx. 50 s	Approx. 25 s	Approx. 20 s
RUN mode	Approx. 1 min 40 s	Approx. 1 min 30 s	Approx. 20 s

Configuration: CPU Unit + 1 Motion Control Unit (See note 2.)

Operating mode	Writing to Memory Card	Comparing with Memory Card	Reading from Memory Card
PROGRAM mode	Approx. 8 min	Approx. 10 min	Approx. 5 min
RUN mode	Approx. 30 min	Approx. 40 min	Approx. 5 min

Note

1. Tested with the Serial Communications Unit's internal parameters at maximum capacity.
2. Tested with a program of the following size in the Motion Control Unit
   - 8 tasks and 20 axes
   - Main program: 2,790 blocks
   - Sub programs: 990 blocks
   - Cam tables: 800 points

Creating Variable Table and Comment Files

Use the following CX-Programmer procedure to create variable table files or comment files on Memory Cards or in EM file memory.

1,2,3...

1. Insert a formatted Memory Card into the CPU Unit or format EM file memory.
2. Place the CX-Programmer online.
3. Select ***Transfer*** and then ***To PLC*** or ***From PLC*** from the PLC Menu.
4. Select either **Symbols** or **Comments** as the data to transfer.

Note If a Memory Card is installed in the CPU Unit, data can be transferred only with the Memory Card. (It will not be possible with EM file memory.)

EM File Memory

Using a Programming Device

1,2,3...

1. Use PLC Setup to specify the starting EM bank to convert to file memory.
2. Use a Programming Device to initialize EM file memory.
3. Use a Programming Device to name the CPU Unit data (user program, I/O memory, parameter area), and then save the data to EM file memory.
4. Use a Programming Device to read the file in EM file memory to the CPU Unit.

Using FREAD(700)/FWRIT(701)/CMND(490)

1,2,3...

1. Use PLC Setup to specify the starting EM bank to convert to file memory.
2. Use a Programming Device to initialize EM file memory.
3. Use FWRIT(701) to name the file in the specified area of I/O memory and then save the file to EM file memory.
4. Use FREAD(700) to read the file from the EM file memory to I/O memory in the CPU Unit.

EM file memory operations can be executed by issuing FINS commands to the local CPU Unit with CMND(490).

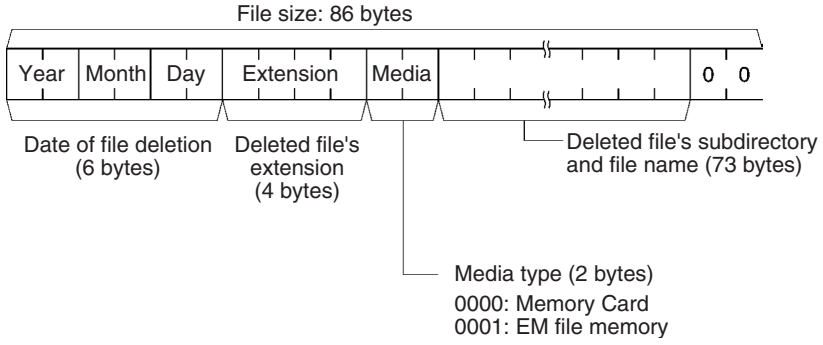
5-3-3 Power Interruptions while Accessing File Memory

If a power interruption occurs while the CPU is accessing file memory (the Memory Card or EM file memory) the contents of the Memory Card may not be accurate. The file being updated may not be overwritten correctly and, in some cases, the Memory Card itself may be damaged.

The affected file will be deleted automatically by the system the next time that power is turned ON. The corresponding File Deletion Notification Flag (A39507 for the Memory Card, A39506 for EM file memory) will be turned ON. The flag will be turned OFF the next time that the power is turned OFF.

When a file is deleted, a deletion log file (DEL_FILE.IOM) will be created in the root directory of the Memory Card or EM file memory. The deletion log file can be read with CX-Programmer or FREAD(700) to check the following information: The date that the file was deleted, the type of file memory (media) that existed, the subdirectory, file name, and extension. When necessary, recreate or recopy the deleted file.

The following diagram shows the structure of the deletion log file.



SECTION 6
Advanced Functions

This section provides details on the following advanced functions: cycle time/high-speed processing functions, index register functions, serial communications functions, startup and maintenance functions, diagnostic and debugging functions, Programming Device functions, and the Basic I/O Unit input response time settings.

6-1 Cycle Time/High-speed Processing

The following functions are described in this section

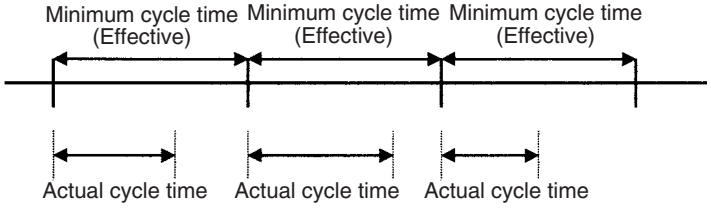
- Minimum cycle time function
- Maximum cycle time function (watch cycle time)
- Cycle time monitoring
- Quick-response inputs
- Interrupt functions
- I/O refresh method
- Data background processing

6-1-1 Minimum Cycle Time

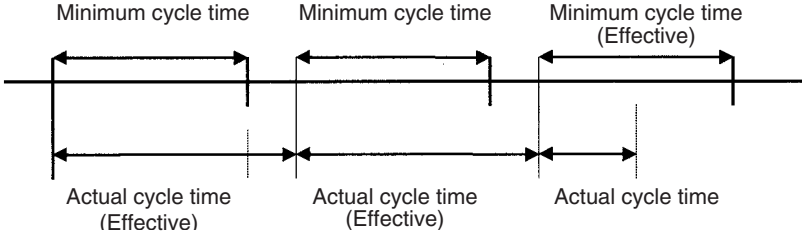
A minimum (or fixed) cycle time can be set in CS/CJ-series PLCs. (See note.) Variations in I/O response times can be eliminated by repeating the program with a fixed cycle time.

Note Indicates the cycle time for program execution when using a parallel processing mode.

The minimum cycle time (1 to 32,000 ms) is specified in the PLC Setup in1-ms units.

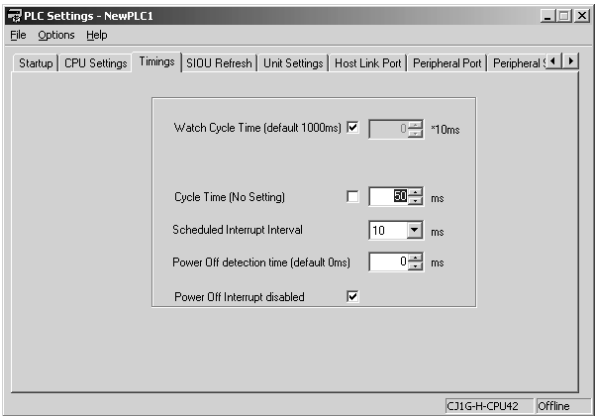


If the actual cycle time is longer than the minimum cycle time, the minimum cycle time function will be ineffective and the cycle time will vary from cycle to cycle.



PLC Setup

When using the CX-Programmer, make the settings on the Timings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address	Name	Setting	Default
+208, bits 0 to 3	Minimum Cycle Time	0001 to 7D00: 1 to 32,000 ms (1-ms units)	0000 (no minimum)

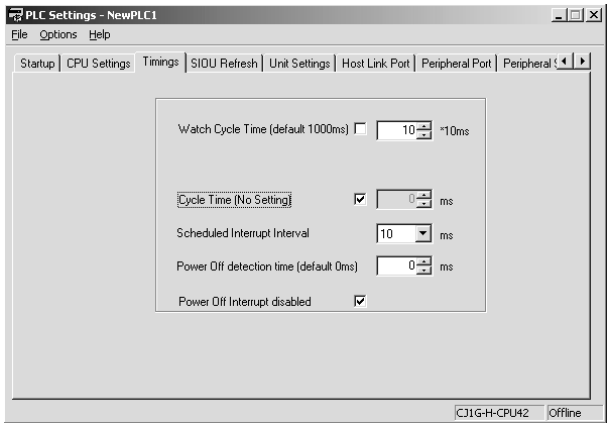
6-1-2 Maximum Cycle Time (Watch Cycle Time)

If the cycle time (see note) exceeds the maximum cycle time setting, the Cycle Time Too Long Flag (A40108) will be turned ON and PLC operation will be stopped.

Note Indicates the cycle time for program execution when using parallel processing mode.

PLC Setup

When using the CX-Programmer, make the settings on the Timings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address	Name	Setting	Default
+209, bit 15	Enable Watch Cycle Time Setting	0: Default (1s) 1: Bits 0 to 14	0001 (1 s)
+209, bits 0 to 14	Watch Cycle Time Setting (Enabled when bit 15 is set to 1.)	001 to FA0: 10 to 40,000 ms (10-ms units)	

Auxiliary Area Flags and Words

Name	Address	Description
Cycle Time Too Long Flag	A40108	A40108 will be turned ON and the CPU Unit will stop operation if the cycle time exceeds the watch cycle time setting. The “cycle time” would be the program execution time when using a Parallel Processing Mode for CS1-H, CJ1-H, or CJ1M CPU Units, or CS1D CPU Units for Single-CPU Systems.

Note If the peripheral servicing cycle exceeds 2.0 s, a peripheral servicing cycle time exceeded error will occur and the CPU Unit will stop operation. If this happens, A40515 (Peripheral Servicing Cycle Time Over Flag) will turn ON.

Note The CS1G/H-CPU□□ (-V1), CJ1□-CPU□□, and CS1D (for Duplex-CPU Systems), CPU Units do not support the parallel processing modes.

6-1-3 Cycle Time Monitoring

The maximum cycle time and present cycle time are stored in the Auxiliary Area every cycle.

Note Indicates the cycle time for program execution when using a parallel processing mode.

Auxiliary Area Flags and Words

Name	Address	Description
Maximum Cycle Time (program execution time for CS1-H, CJ1-H, or CJ1M CPU Units in parallel processing mode)	A262 and A263	Stored every cycle in 32-bit binary in the following range: 0 to 429,496,729.5 ms in 0.1 ms units (0 to FFFF FFFF)
Present Cycle Time (program execution time for CS1-H, CJ1-H, or CJ1M CPU Units in parallel processing mode)	A264 and A265	Stored every cycle in 32-bit binary in the following range: 0 to 429,496,729.5 ms in 0.1 ms units (0 to FFFF FFFF)

A Programming Device (CX-Programmer or Programming Console) can be used to read the average of the cycle times in the last 8 cycles.

Reducing the Cycle Time

The following methods are effective ways to reduce the cycle time in CS/CJ-series PLCs:

1,2,3...

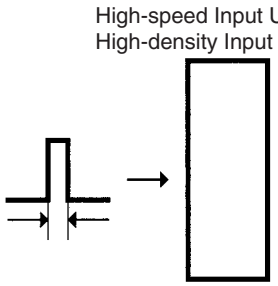
1. Put tasks that aren't being executed in standby.
2. Jump program sections that aren't being executed with JMP(004) and JME(005).

Note Communications response can be increased by using a parallel processing mode. (CS1-H and CJ1-H CPU Units, and CS1D CPU Units for Single-CPU Systems do not support this function.) If a parallel processing mode is used, the current peripheral servicing cycle will be stored in A268 (Peripheral Servicing Cycle Time) each peripheral servicing cycle.

6-1-4 High-speed Inputs

When you want to receive pulses that are shorter than the cycle time, use the CS1W-IDP01 High-speed Input Unit or use the high-speed inputs of the C200H-ID501/ID215 and C200H-MD501/MD115/MD215 High-density I/O Units.

The high-speed inputs can receive pulses with a pulse width (ON time) of 1 ms or 4 ms for the C200H High-density Input Units and 0.1 ms for the CS1W-IDP01 High-speed Input Unit.



CS1W-IDP01: 0.1 ms
CJ1W-IDP01: 0.05 ms
C200H-ID501/ID215/MD501/MD115/MD215: 4 ms

6-1-5 Interrupt Functions

Interrupt tasks can be executed for the following conditions. Refer to *4-3 Interrupt Tasks* for more details.

Note The CS1D CPU Units for Duplex-CPU Systems do not support interrupts. With the CS1D CPU Units, interrupt tasks can be used only as extra cyclic tasks, i.e., no other type of interrupt task can be used.

I/O Interrupts (Interrupt tasks 100 to 131)

An I/O interrupt task is executed when the corresponding input (on the rising edge of the signal or, for CS/CJ-series Interrupt Input Units, on either the rising or falling edge) is received from an Interrupt Input Unit.

Scheduled Interrupts (Interrupt tasks 2 and 3)

A scheduled interrupt task is executed at regular intervals.

Power OFF Interrupt (Interrupt task 1)

This task is executed when the power is interrupted.

External Interrupts (Interrupt tasks 0 to 255)

An external interrupt task is executed when an interrupt is received from a Special I/O Unit, CPU Bus Unit, or Inner Board.

Note The built-in interrupt inputs and high-speed counter inputs on a CJ1M CPU Unit can be used to activate interrupt tasks. Refer to the *CJ Series Built-in I/O Operation Manual* for details.

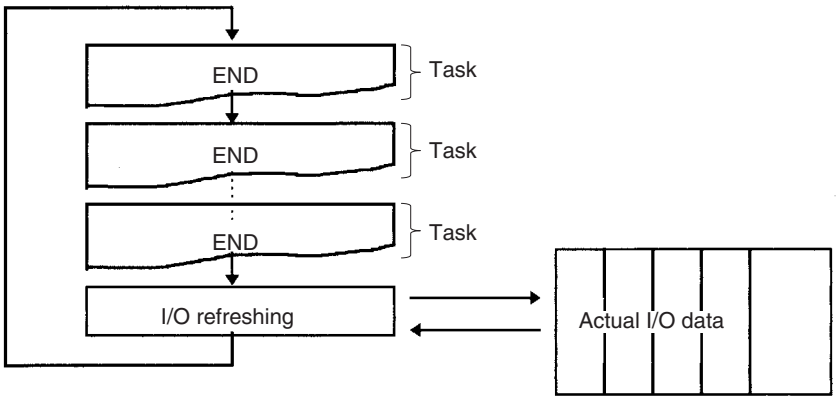
6-1-6 I/O Refreshing

There are two ways to refresh real I/O, cyclic refreshing and immediate refreshing, which have different timing. The following table shows when these methods can be used.

Relevant Unit / I/O refreshing method		Basic I/O Units	Special I/O Units	CPU Bus Units
Cyclic refreshing		Can be used.	Can be used.	Can be used.
Immediate refreshing	**Specified as an instruction variation (!)**	Can be used.	Cannot be used.	Cannot be used.
	IORF(097) instruction	Can be used.	Can be used.	Cannot be used.
	FIORF(225) instruction (CJ1-H-R CPU Units only)	Cannot be used.	Can be used.	Cannot be used.
	DLNK(226) instruction	Cannot be used.	Cannot be used.	Can be used.

1. Cyclic Refreshing

I/O refreshing is performed after all of the instructions in executable tasks have been executed. (The PLC Setup can be set to disable cyclic refreshing of individual Special I/O Units.)

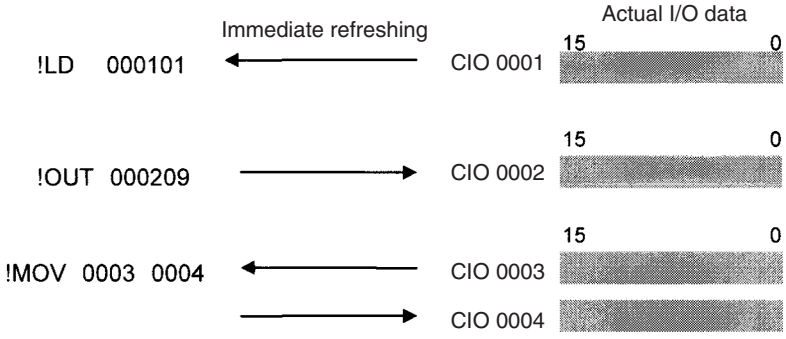


2. Immediate Refreshing

Using the Instruction's Immediate Refresh Variation

When an address in the I/O Area is specified as an operand in the immediate-refreshing variation of an instruction, that operand data will be refreshed when the instruction is executed. Immediate-refreshing instructions can refresh data allocated to Basic I/O Units.

Immediate refreshing is also possible for the built-in I/O on CJ1M CPU Units.



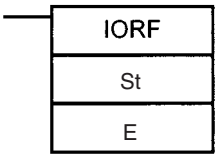
Operand specification / Operand type	Bit address specification	Word address specification
Input or source	The entire word containing the specified bit will be refreshed just before the instruction is executed (input refreshing).	The specified word (16 bits) will be refreshed just before the instruction is executed (input refreshing).
Output or destination	The entire word containing the specified bit will be refreshed just after the instruction is executed (output refreshing).	The specified word (16 bits) will be refreshed just after the instruction is executed (output refreshing).

Note 1. The execution times for immediate-refreshing variations are longer than the regular variations of instructions, so the cycle time will be longer. Refer to *10-5 Instruction Execution Times and Number of Steps* in the *Operation Manual* for details.

2. CS1D CPU Units for Duplex-CPU Systems do not support immediate refreshing.

■ IORF(097): I/O REFRESH

IORF(097) can be used to refresh a range of I/O words upon execution of the instruction. IORF(097) can refresh data allocated to Basic I/O Units and Special I/O Units.

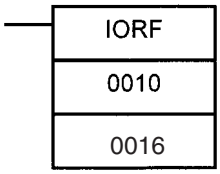


St: Start word

E: End word

The data in all words from St through E are refreshed when IORF(097) is executed.

The following example shows IORF(097) used to refresh 8 words of I/O data.



The 7 words from CIO 0010 through CIO 0016 are refreshed when IORF(097) is executed.

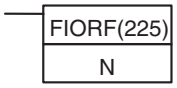
When a high-speed response is needed from a calculation that uses input data from a Basic I/O Unit or outputs data to a Basic I/O Unit, use IORF(097) just before and just after the calculation instruction.

Note IORF(097) has a relatively long instruction execution time and that execution time increases proportionally with the number of words being refreshed, so it can significantly increase the cycle time. Refer to *10-5 Instruction Execution Times and Number of Steps* in the *Operation Manual* for more details.

■ FIORF(225): SPECIAL I/O UNIT I/O REFRESH (CJ1-H-R CPU Units Only)

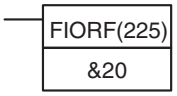
FIORF(225) can be used to refresh the following data in a Special I/O Unit with the specified unit number, only when necessary.

- Allocated CIO Area words
- Allocated DM Area words



N: Special I/O Unit's unit number

Example:



Refreshes the CIO words and DM words allocated to the Special I/O Unit with unit number &20.

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 can prevent the PLC's cycle time from increasing when a Special I/O Unit is mounted.

Note IORF(097) can immediately refresh a specified range of I/O words (in the CIO Area only) allocated to Basic I/O Units or Special I/O Units. In contrast, FIORF(225) immediately refreshes all of the CIO Area words and DM Area words allocated to a specified Special I/O Unit.

■ Using DLNK(226): CPU Bus Unit I/O Refresh

DLNK(226) is used to refresh data for a CPU Bus Unit of a specified unit number. The following data is refreshed.

- Words allocated to the Unit in the CIO Area
- Words allocated to the Unit in the DM Area
- Data specific to the Unit (See note.)

Note Data specific to a CPU Bus Unit would include data links for Controller Link Unit or SYSMAC LINK Units, as well as remote I/O for DeviceNet Units.

N: Unit number of CPU Bus Unit

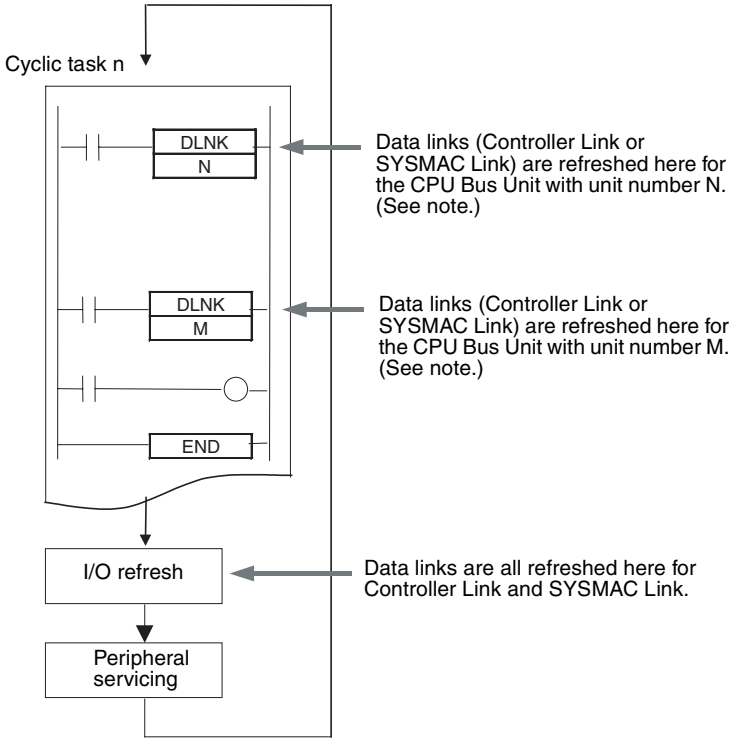
Example:

The instruction on the left would refresh the words allocated to the Unit in the CIO Area and DM Area, and data specific to the CPU Bus Unit with unit number 1.

Application Example: With a long cycle time, the refresh interval for Controller Link data links can be very long. This interval can be shortened by executing DLNK(226) for the Controller Link Unit to increase the frequency of data link refreshing.

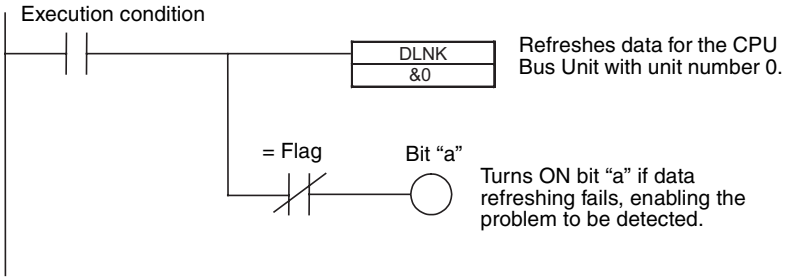
Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support this function.

Note 1. Longer cycle times (e.g., 100 ms) will increase the interval between when data links are refreshed. DLNK(226) can be used in this case, as shown in the following example.



Note If DLNK(226) is executed for a CPU Bus Unit that is busy refreshing data, data will not be refreshed and the Equals Flag will turn OFF. Normally, the Equals Flag should be programmed as shown below to

be sure that refreshing has been completed normally.



2. IORF(097) is used to refresh data for Basic I/O Units and Special I/O Units. DLNK(226) is used to refresh CPU Bus Units (CIO and DM Area words allocated to the Units and special data for the Units).

Disabling Special I/O Unit Cyclic Refreshing

Ten words in the Special I/O Unit Area (CIO 2000 to CIO 2959) are allocated to each Special I/O Unit based on the unit number set on the front of the Unit. Data is refreshed between this area and the CPU Unit each cycle during I/O refreshing, after all of the instructions in the executable tasks have been executed.

If many Special I/O Units are installed, the cycle time may become too long due to the time required for I/O refreshing of the Special I/O Units. To reduce the I/O refreshing time, cyclic refreshing can be disabled for individual Special I/O Units in the PLC Setup.

Note If the I/O refreshing interval is too short, the Unit's internal processing may not be able to keep pace, the Special I/O Unit Error Flag (A40206) will be turned ON, and the Special I/O Unit will not operate properly.
In this case, the cycle time can be extended by setting a minimum cycle time in the PLC Setup, or the Special I/O Unit's cyclic I/O refreshing can be disabled in the PLC Setup.

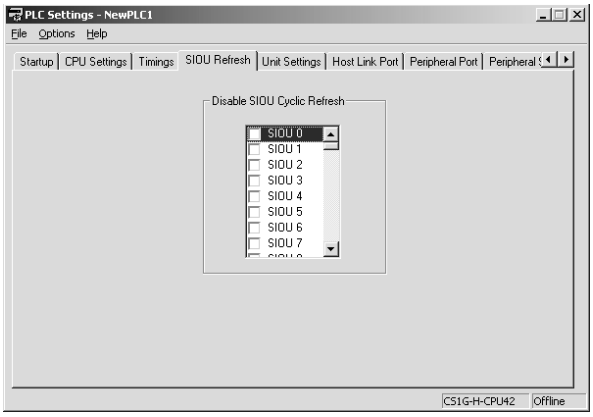
When a Special I/O Unit's cyclic refreshing has been disabled, the Unit's data will not be refreshed unless the IORF(097) or FIORF(225) (CJ1-H-R only) instruction is executed for the Unit in the ladder program.

Note

1. If an IORF(097), FIORF(225) (CJ1-H-R only), IORD(222), or IOWR(223) instruction is being executed for a Special I/O Unit in an interrupt task, always disable cyclic refreshing for that Special I/O Unit in the PLC Setup.
   An interrupt task error will occur if a Special I/O Unit is being refreshed by cyclic refreshing, and an IORF(097), FIORF(225) (CJ1-H-R only), IORD(222), or IOWR(223) instruction is executed for that Special I/O Unit.
2. When you want to disable a Special I/O Unit's cyclic refreshing and refresh the Unit's I/O data only when necessary, execute the FIORF(225) instruction (CJ1-H-R only) for that Unit.

PLC Setup

When using the CX-Programmer, make the settings on the SIOU Refresh Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address	Name	Setting	Default
+226 to 231, bits 0 to 15	Cyclic Refreshing Disable Bit for Special I/O Unit 0	0: Enabled 1: Disabled	0 (Enabled)
	:	:	:
	Cyclic Refreshing Disable Bit for Special I/O Unit 95	0: Enabled 1: Disabled	0 (Enabled)

Improving Refresh Response for CPU Bus Unit Data

This function is supported only by CS1-H, CJ1-H, CJ1M, or CS1D CPU Units.

Normally, data links and other special data for CPU Bus Units are refreshed along with the CIO and DM Area words allocated to the Units during the I/O refresh period following program execution.

The following table lists some example of special data for CPU Bus Units.

Units	Special data
Controller Link Units and SYSMAC LINK Units	Controller Link and SYSMAC LINK data links (including automatically and user-set links)
CS/CJ-series DeviceNet Units	DeviceNet remote I/O communications (including fixed allocations and user-set allocations)

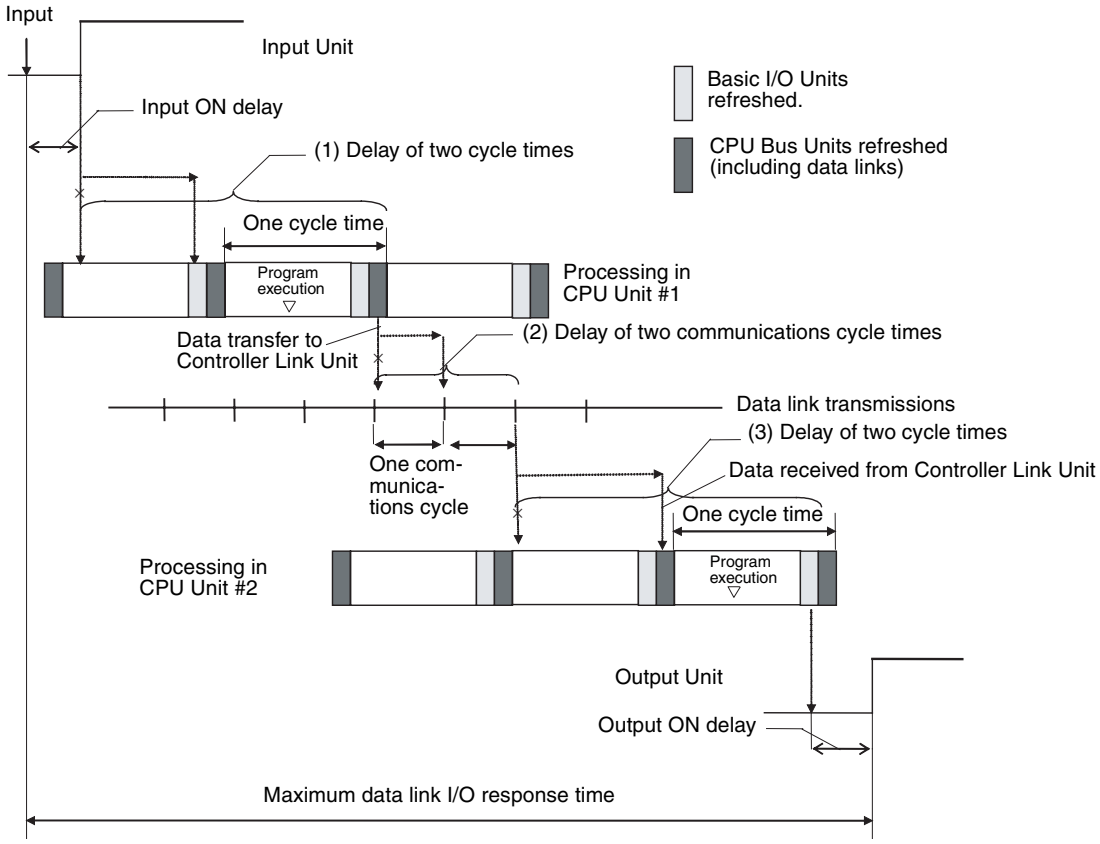
The following functions can be used to improve the refresh response for special CPU Bus Unit data

- Reducing the cycle time by using parallel processing mode or high-speed instructions (Parallel processing mode is not supported by CS1D CPU Units for Duplex-CPU Systems.)
- Executing DLNK(226) to refresh specific CPU Bus Units by specifying their unit numbers (DLNK(226) can be used more than once in the program.)

Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support this function.

Maximum Data Link I/O Response Time

Normal Processing (Not Using DLNK(226))



There are three points shown in the diagram above where processing is delayed, increasing the data link I/O response time.

1,2,3... 1. The input arrives in the PLC (CPU Unit #1) just after I/O refreshing, causing a delay of one cycle before the input is read into the PLC. CPU Bus Units are refreshed after program execution, causing a total delay of two cycle times.

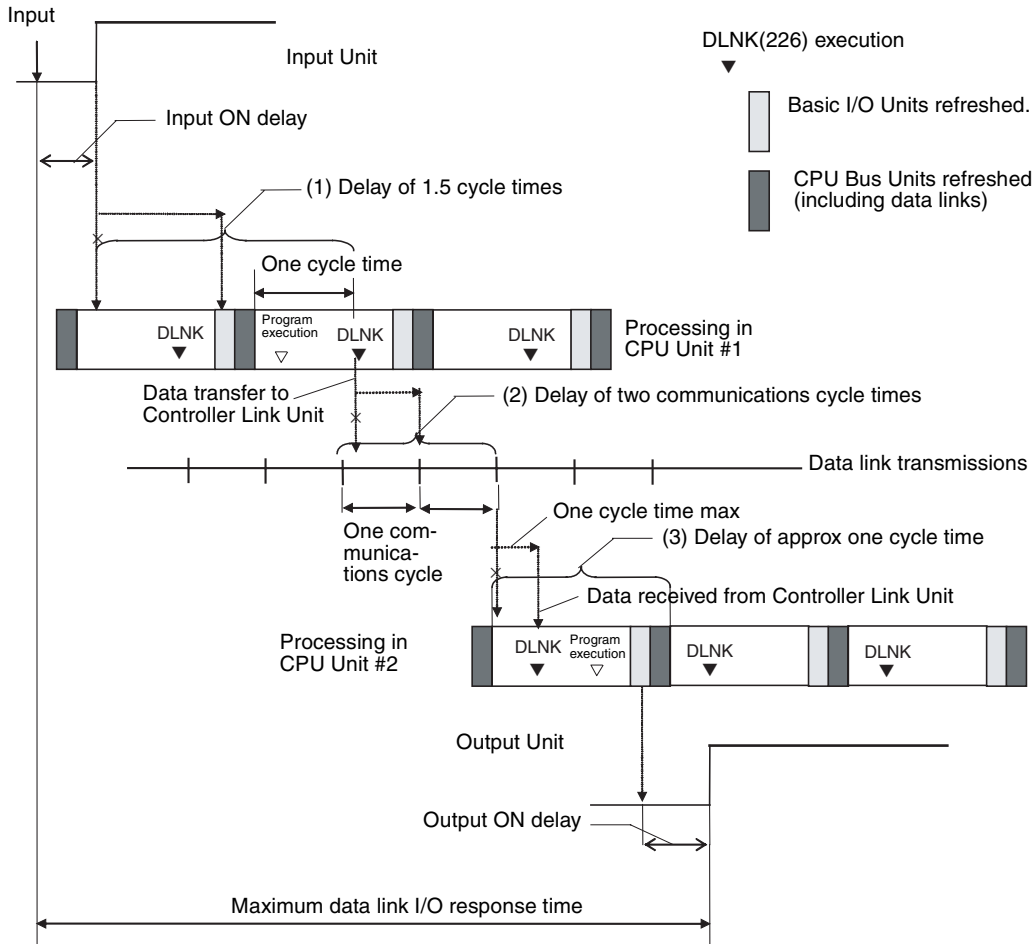
2. Data exchange occurs just after the PLC passes the token that makes it the polling node, causing a delay of up to one communications cycle time before the data is transferred in data link processing. There will also be a delay of up to one communications cycle time after receiving the token, causing a total delay of up to two communications cycle times.

3. The data transferred in data link processing arrives at the PLC (CPU Unit #2) after data exchange, so the data will not be read into the PLC until the next data exchange, causing a delay of up to one cycle. CPU Bus Units are refreshed after program execution, causing a total delay of two cycle times.

The equation for maximum data link I/O response time is as follows:

Input ON delay	1.5 ms
Cycle time of PLC at CPU Unit #1 × 2	25 ms × 2
Communications cycle time × 2	10 ms × 2
Cycle time of PLC at CPU Unit #2 × 2	20 ms × 2
Output ON delay	15 ms
Total (data link I/O response time)	126.5 ms

Example Using DLNK(226)



There are three points shown in the diagram above where processing is delayed, increasing the data link I/O response time.

Note In this example, it is assumed that DNLK(226) is placed after other instructions in the program in both CPU Units

1,2,3... 1. The input arrives in the PLC (CPU Unit #1) just after I/O refreshing, causing a delay of one cycle before the input is read into the PLC. CPU Bus Units are refreshed during program execution, reducing the total delay to approximately 1.5 cycle times.

2. Data exchange occurs just after the PLC passes the token that makes it the polling node, causing a delay of up to one communications cycle time before the data is transferred in data link processing. There will also be a delay of up to one communications cycle time after receiving the token, causing a total delay of up to two communications cycle times.

3. The data transferred in data link processing arrives at the PLC (CPU Unit #2) after the I/O refresh, but DLNK(226) refreshes the data, so the data will be read into the PLC without causing a delay of up to one cycle. The Basic I/O Units are refreshed after program execution, causing a total delay of approximately one cycle time.

The equation for maximum data link I/O response time is as follows:

Input ON delay	1.5 ms	---
Cycle time of PLC at CPU Unit #1 × 1.5	25 ms × 1.5	Faster by 12.5 ms (25 ms × 0.5)

Communications cycle time × 2	10 ms × 2	---
Cycle time of PLC at CPU Unit #2 × 1	20 ms × 1	Faster by 20 ms (20 ms × 1)
Output ON delay	15 ms	---
Total (data link I/O response time)	94 ms	Faster by 32.5 ms (26% faster)

6-1-7 Background Execution

Background execution can be used to reduce fluctuations in the cycle time. Background execution is supported only by CS1-H, CJ1-H, or CJ1M CPU Units, or CS1D CPU Units for Single-CPU Systems.

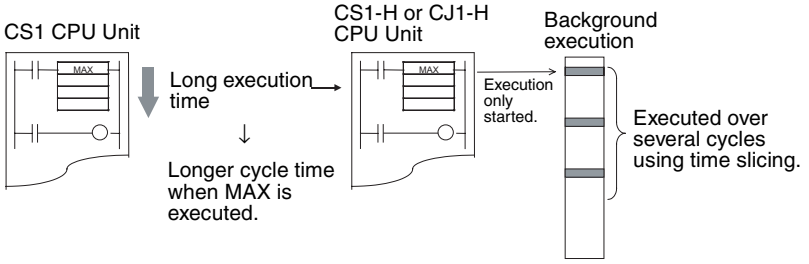
Table data processing (such as data searches) and text string processing (such as text string searches), require time to execute, and can create large fluctuations in the cycle time due to the extended amount of time required to execute them.

Background execution (time slicing) can be used to execute the following instructions over several cycles to help control fluctuations in the cycle time. The PLC Setup enables setting background execution for each type of instruction.

Note The CS1G/H-CPU□□ (-V1), CJ1□-CPU□□, and CS1D (for Duplex-CPU Systems), CPU Units do not this function.

- Table data processing instructions
- Text string processing instructions
- Data shift instructions (ASYNCHRONOUS SHIFT REGISTER only)

Setting background execution for the above instructions can help control temporary increases in the cycle time.



Applications

Background execution can be used for large quantities of data processing, such as data compilation or processing, that is required only at special times (e.g., once a day) when reducing the effect on the cycle time is more important than the speed of the data processing.

Procedure

1,2,3...

1. Set the PLC Setup to enable background execution for the required instructions.
2. Set the communications port number (logical port number) to be used for background execution in the PLC Setup. This port number will be used for all instructions processed in the background.

   Note One port is used for all background execution. Background execution for an instruction can thus not be started if background execution is already being performed for another instruction. Use the Communications Port Enabled Flag to control instructions specified for back-

ground execution so that no more than one instruction is executed at the same time.

3. If an instruction for which background execution has been specified is executed, execution will only be started in the cycle in which the execution condition was met and execution will not be completed in the same cycle.
4. When background execution is started, the Communications Port Enabled Flag for that port will be turned OFF.
5. Background execution will be continued over several cycles.
6. When processing has been completed, the Communications Port Enabled Flag for that port will be turned ON. This will enable another instruction to be executed in the background.

Applicable Instructions

Background processing will not be performed for the following instructions when they are used in function blocks. They will be executed using normal processing.

■ **Table Data Processing Instructions**

Instruction	Mnemonic
DATA SEARCH	SRCH
SWAP BYTES	SWAP
FIND MAXIMUM	MAX
FIND MINIMUM	MIN
SUM	SUM
FRAME CHECKSUM	FCS

■ **Text String Processing Instructions**

Instruction	Mnemonic
MOVE STRING	MOV$
CONCATENATE STRING	+$
GET STRING LEFT	LEFT$
GET STRING RIGHT	RIGHT$
GET STRING MIDDLE	MID$
FIND IN STRING	FIND$
STRING LENGTH	LEN$
REPLACE IN STRING	RPLC$
DELETE STRING	DEL$
EXCHANGE STRING	XCHG$
CLEAR STRING	CLR$
INSERT INTO STRING	INS$

■ **Data Shift Instructions**

Instruction	Mnemonic
ASYNCHRONOUS SHIFT REGISTER	ASFT

Differences between Instructions Executed Normally and in the Background

The differences between normal instruction execution and execution in the background are listed below.

■ **Outputting to Index Registers (IR)**

If MAX(182) or MIN(183) is executed to output the I/O memory map address of the word containing the minimum or maximum value to an index register,

the address will not be output to the index register and will be output to A595 and A596 instead. To store the address in an index register, use a Data Move instruction (e.g., MOVL(498)) to copy the address in A595 and A596 to an index register.

■ **Conditions Flags**

Conditions Flags will not be updated following execution of instructions processed in the background. To access the Conditions Flag status, execute an instruction that affects the Conditions Flags in the same way, as shown in the following example, and then access the Conditions Flags.

■ **Outputting to Index Register IR00**

If SRCH(181) is executed to output the I/O memory map address of the word containing the matching value (the first word if there is more than one) to an index register, the address will not be output to the index register and will be output to A595 and A596 instead.

■ **Outputting to Data Registers (DR) for SRCH(181)**

If SRCH(181) is executed to output the matching data to a data register, the data will not be output to the data register and will be output to A597 instead.

■ **Matching Text Strings**

If SRCH(181) finds matching data, it will not turn ON the Equals Flag, but will turn on A59801 instead.

■ **Instruction Errors**

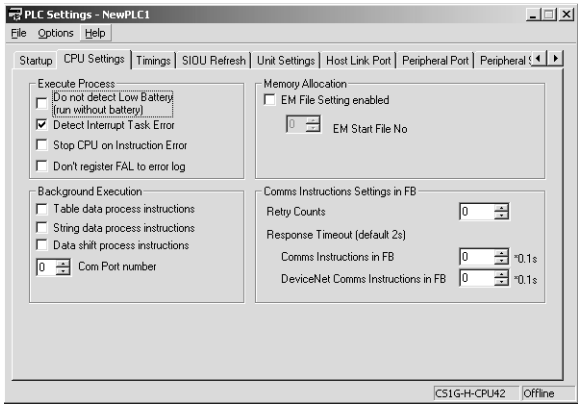
If an instruction execution error or illegal access error occurs for an instruction being processed in the background, the ER or AER Flags will not be turned ON and A39510 will be turned ON instead. A39510 will remain ON until the next time an instruction is processed in the background.

■ **Outputting to Data Registers (DR) for MAX(182) or MIN(183)**

If MAX(182) or MIN(183) is executed with a data register specified as the output word for the minimum or maximum value, an instruction execution error will occur and the ER Flag will turn ON.

PLC Setup

When using the CX-Programmer, make the settings on the CPU Settings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Word	Bits	Name	Setting	Default and update timing
+198	15	Table Data Instruction Background Execution	0: Not processed in background 1: Processed in background	0: Not processed in background Start of operation
	14	Text String Instruction Background Execution	0: Not processed in background 1: Processed in background	
	13	Data Shift Instruction Background Execution	0: Not processed in background 1: Processed in background	
	00 to 03	Communications Port Number for Background Execution	0 to 7 Hex: Communications ports 0 to 7 (internal logical ports)	0 Hex: Port 0 Start of operation

Auxiliary Area Flags and Words

Name	Address	Description
Communications Port Enabled Flags	A20200 to A20207	Turns ON when a network instruction can be executed with the corresponding port number or background execution can be executed with the corresponding port number. Bits 00 to 07 correspond to communications ports 0 to 7 **Note** When the simple backup operation is used to performed a write or compare operation for a Memory Card, a communications port will be automatically allocated, and the corresponding flag will be turned ON during the operation and turned OFF when the operation has been completed.
Communications Port Error Flags	A21900 to A21907	Turns ON when an error occurred during execution of a network instruction. Bits 00 to 07 correspond to communications ports 0 to 7. **Note** When the simple backup operation is used to performed a write or compare operation for a Memory Card, a communications port will be automatically allocated. The corresponding flag will be turned ON if an error occurs and will be turned OFF if the simple backup operation ends normally.
Communications Port Completion Codes	A203 to A210	These words contain the completion codes for the corresponding port numbers when network instructions have been executed. The contents will be cleared when background execution has been completed. Words A203 to A210 correspond to communications ports 0 to 7. **Note** When the simple backup operation is used to performed a write or compare operation for a Memory Card, a communications port will be automatically allocated, and a completion code will be stored in the corresponding word.
Background Execution ER/AER Flag	A39510	Turns ON when an instruction execution error or illegal access error occurs in an instruction being executed in the background. Turns OFF when power is turned ON or operation is started.

Name	Address	Description
Background Execution IR00 Output	A595 and A596	These words receives the output when the output of an instruction executed in the background is specified for an index register. No output will be made to IR00. Range: 0000 0000 to FFFF FFFF Hex Lower 4 digits: A595, Upper 4 digits: A596
Background Execution DR00 Output	A597	This word receives the output when the output of an instruction executed in the background is specified for a data register. No output will be made to DR00. Range: 0000 to FFFF Hex
Background Execution Equals Flag Output	A59801	This flag is turned ON when matching data is found for a SRCH(181) executed in the background.

Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support this function.

Note The communications ports (internal logical ports) in the CPU Unit are used both for background execution and the following instructions

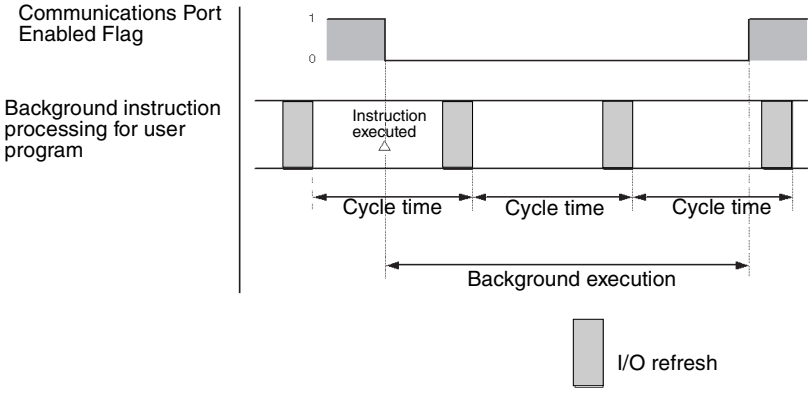
- SEND(090), RECV(098), and CMND(490) (Network Communications Instructions)
- PMCR(260) (PROTOCOL MACRO)
- TXDU(256) and RXDU(255) (the no-protocol communications instructions used with Serial Communications Units)

Background instructions and the above instructions cannot be executed simultaneously on the same port. Use the Communications Port Enabled Flags to be sure that only one instruction is executed on each port at any one time.

Note If an instruction is specified for execution in the background for a port for which the Communications Port Enabled Flag is OFF, the ER Flag will turn ON and the background instruction will not be executed.

Communications Port Enabled Flags

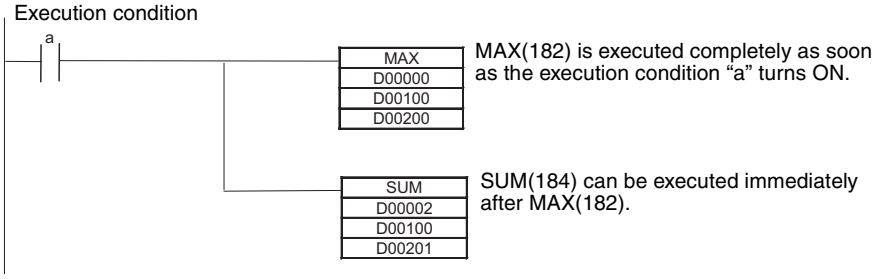
The Communications Port Enabled Flags are ON when the port is not being used and OFF when processing is being performed on the port.



Programming Example 1

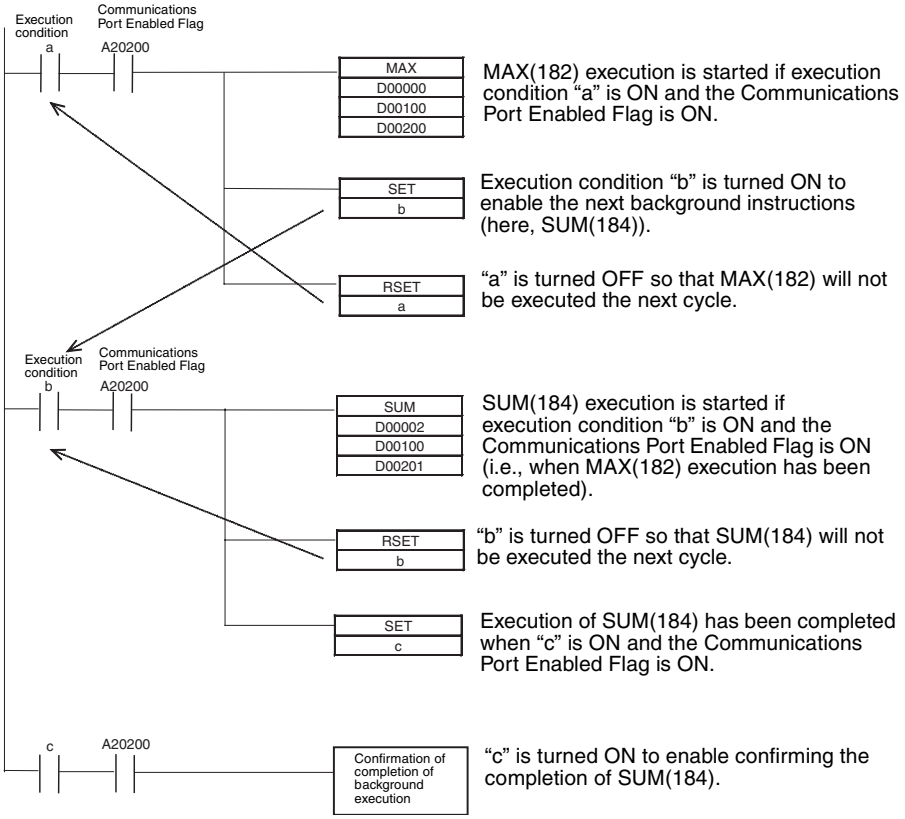
■ Traditional Programming without Background Execution

As shown below, processing is completed when the instruction is executed.



■ Programming with Background Execution

With background execution, the program is changed so that MAX(182) is executed only when the specified Communications Port Enabled Flag is ON (i.e., only when the port is not already being used for background execution or network communications). Also, input conditions are controlled with SET and RESET instructions to ensure that processing is performed in the correct order. (Communications port 0 is used for background execution in the following example.)

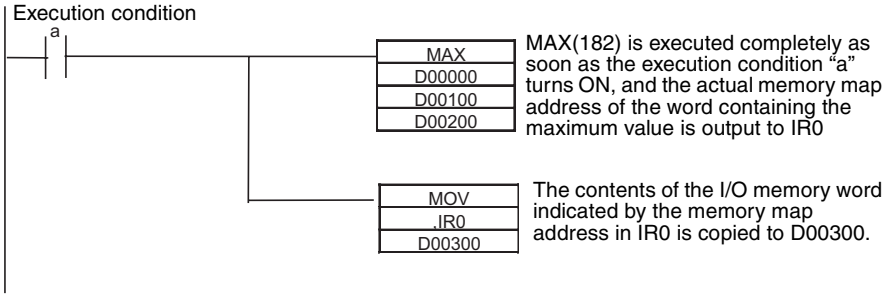


Programming Example 2

This examples show background execution when index register output is specified, as is possible for MAX(182), MIN(183), and SRCH(181).

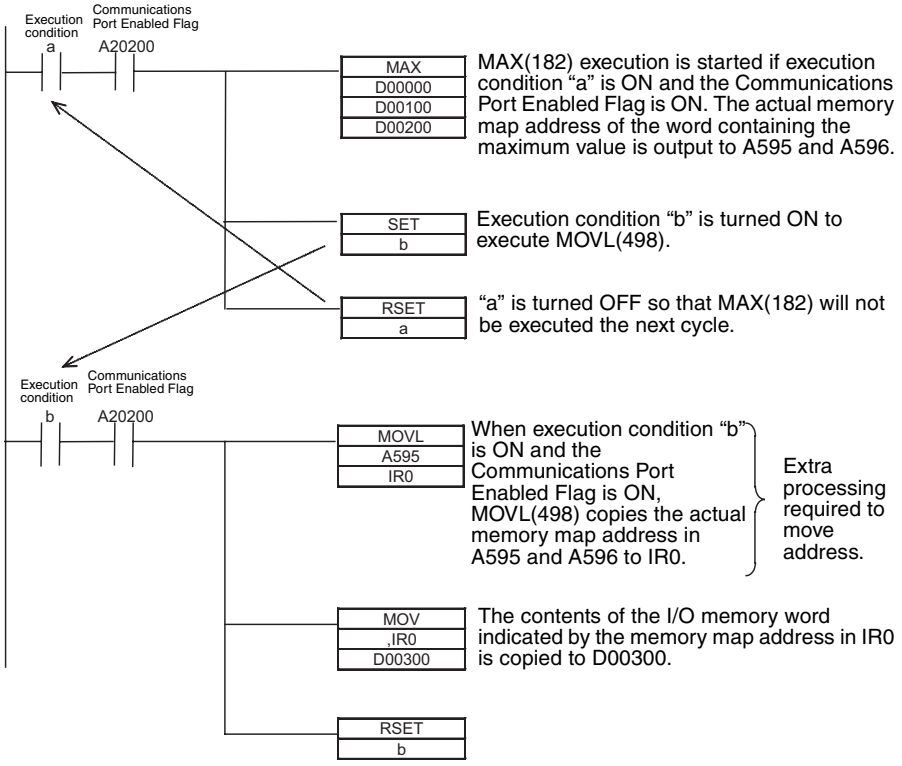
■ Traditional Programming without Background Execution

As shown below, the actual memory map address of the word containing the maximum value is output to an index register.



■ Programming with Background Execution

With background execution, the actual memory map address of the word containing the maximum value is output to A595 and A596. MOVL(498) is then used the actual memory map address to the index register.

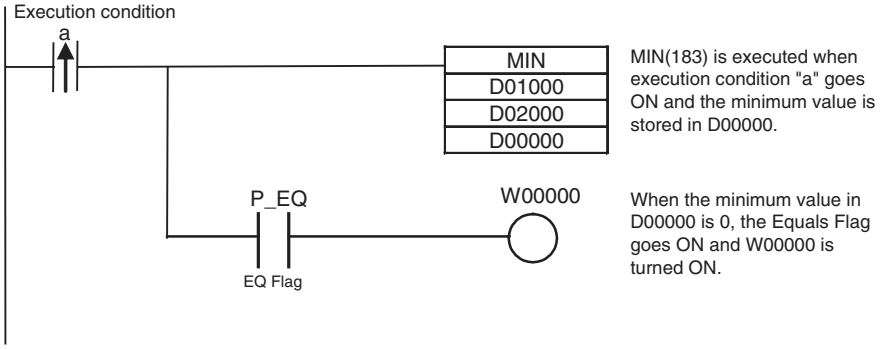


Programming Example 3

This example shows background execution when referencing Condition Flags.

■ Traditional Programming without Background Execution

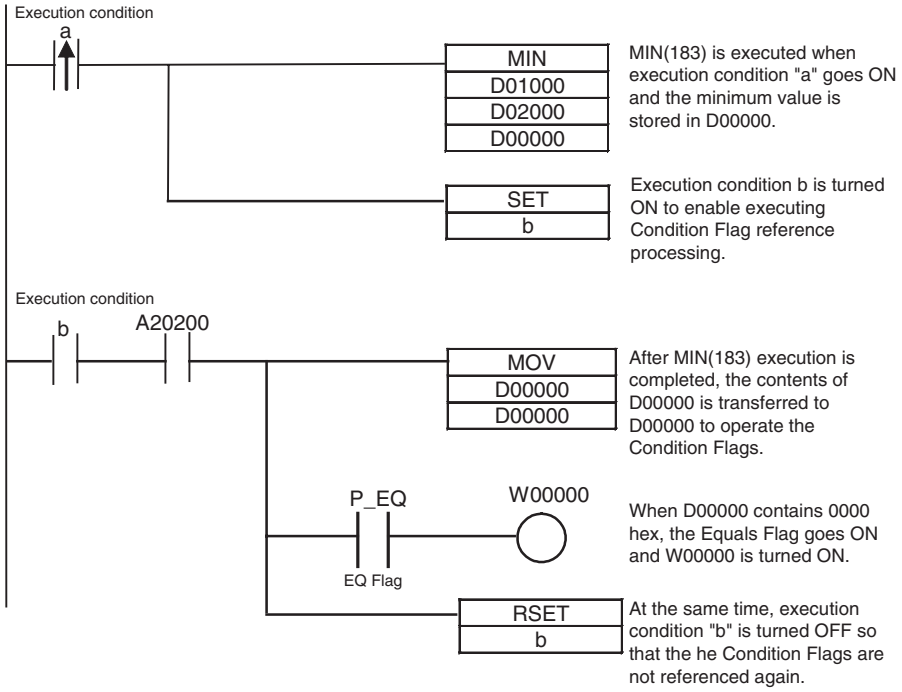
To check whether the minimum value found by MIN(183) is 0, the status of the Equals Flag is checked just after execution of MIN(183).



■ Programming with Background Execution

MOV(021) is used to check whether the minimum value found by MIN(183) is 0, because the Equals Flag in MOV(021) (Equals Flag = ON when the source data in S is 0) operates in the same way as in MIN(183) (Equals Flag = ON when the minimum value is 0).

The minimum value found by MIN(183) is transferred to the same address by MOV(021). By referencing the Equals Flag after the MOV(021) transfer, it is possible to reference the status of the Equals Flag when MIN(183) was executed.



6-1-8 Sharing Index and Data Registers between Tasks

Index and Data Registers (IR/DR) can be shared between tasks.The normal setting is for separate registers for each task. The current setting can be confirmed in A09914.

Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support this function.

Note

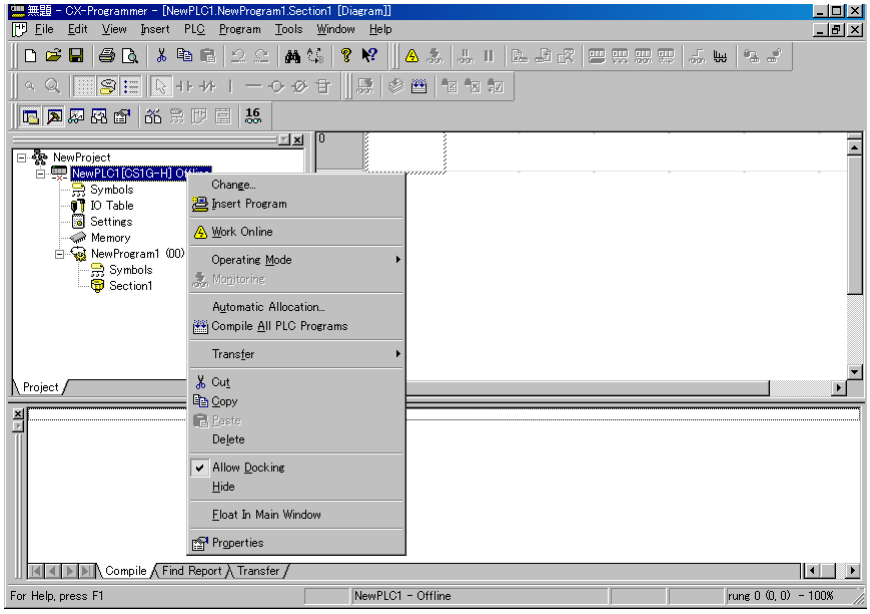
1. Shared Index and Data Registers can be used to eliminate the need to store and load register contents between tasks when the same contents is needed in two or more tasks. Refer to the section on index registers in the *CS Series Operation Manual* (W339) or the *CJ Series Operation Manual* (W393) for information on storing and loading index register contents.
2. The switching time between tasks will be somewhat faster when index and data registers are shared. It is recommended to set shared registers if the registers are not being used or if there is no particular need for separate registers in each task.

Setting Method

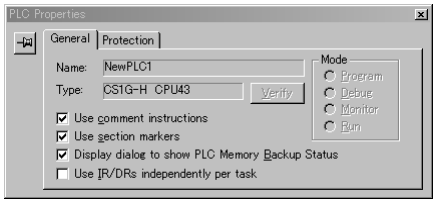
Use the CX-Programmer to set shared index and data registers. This setting cannot be made from a Programming Console.

1,2,3...

1. Select a PLC (PLC) in the CX-Programmer project tree and click the right mouse button.



2. Select ***Properties.*** The following dialog box will be displayed.



3. Leave the checkmark for using IR/DR independently per task if separate index and data registers are required for each task. Remove the checkmark to use shared index and data registers for all tasks.

Auxiliary Area Flags and Words

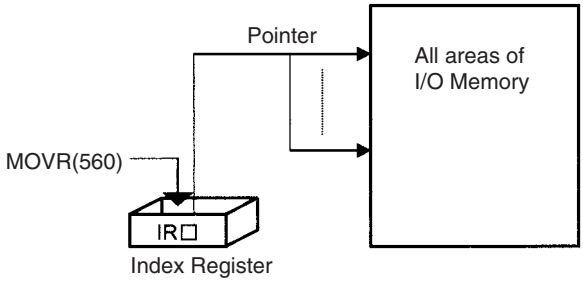
Name	Address	Description
IR/DR Operation between Tasks	A09914	Indicates whether or not index and data registers are shared between tasks. 0: Separate registers for each task (default) 1: Shared registers for all tasks

6-2 Index Registers

6-2-1 What Are Index Registers?

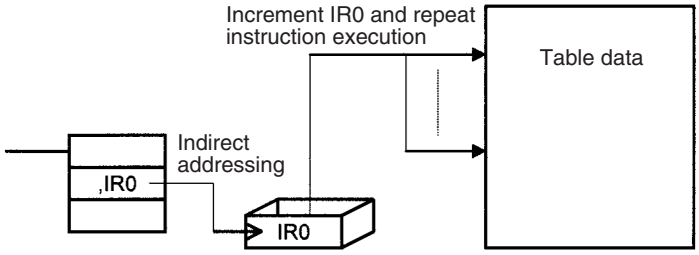
Index Registers function as pointers to specify PLC memory addresses, which are absolute memory addresses in I/O memory. After storing a PLC memory address in an Index Register with MOVR(560) or MOVRW(561), input the Index Register as an operand in other instructions to indirectly address the stored PLC memory address.

The advantage of Index Registers is that they can specify any bit or word in I/O memory, including timer and counter PVs.



6-2-2 Using Index Registers

Index Registers can be a powerful tool when combined with loops such as FOR-NEXT loops. The contents of Index Registers can be incremented, decremented, and offset very easily, so a few instructions in a loop can process tables of consecutive data very efficiently.



Basic Operation

Basically, Index Registers are used with the following steps:

1,2,3...

1. Use MOVR(560) to store the PLC memory address of the desired bit or word in an Index Register.
2. Specify the Index Register as the operand in almost any instruction to indirectly address the desired bit or word.
3. Offset or increment the original PLC memory address (see below) to redirect the pointer to another address.
4. Continue steps 2 and 3 to execute the instruction on any number of addresses.

Offsetting, Incrementing, and Decrementing Addresses

The following table shows the variations available for indirect addressing.

Variation	Syntax
Indirect addressing	,IR□
Indirect addressing with constant offset	Constant ,IR□ (Include a + or – in the constant.)

Variation	Syntax
Indirect addressing with DR offset	DR□, IR□
Indirect addressing with auto-increment	Increment by 1: ,IR□+ Increment by 2: ,IR□++
Indirect addressing with auto-decrement	Decrement by 1: ,–IR□ Decrement by 2: ,– –IR□

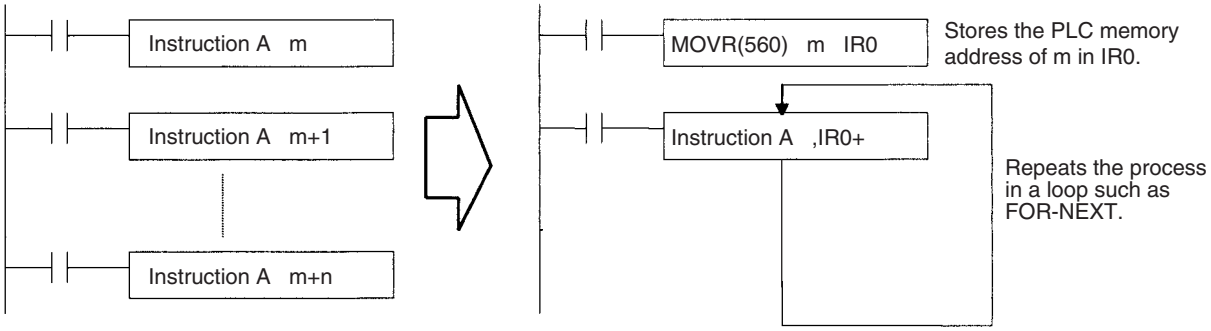
Instructions That Directly Address Index Registers

Index registers can be directly addressed by the following instructions.

DOUBLE SIGNED BINARY ADD WITHOUT CARRY: +L(401), DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY: –L(411), DOUBLE INCREMENT BINARY: ++L(591), and DOUBLE DECREMENT BINARY: – – L(593)

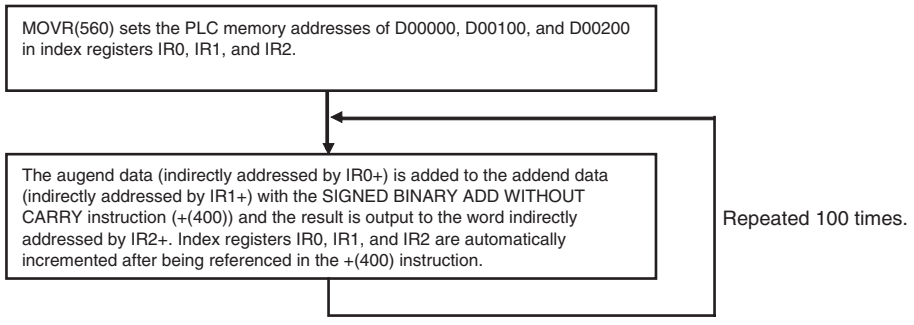
Example 1

The following example shows how an Index Register in a program loop can replace a long series of instructions. In this case, instruction A is repeated n+1 times to perform some operation such as reading and comparing a table of values.

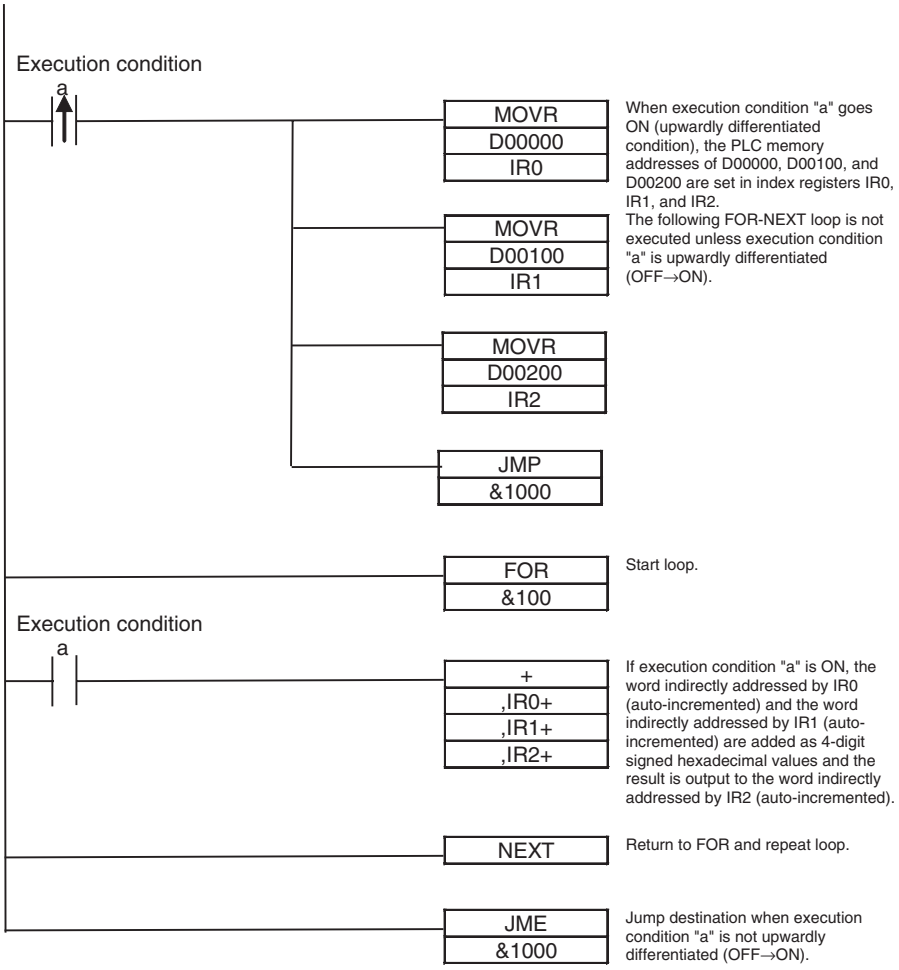


Example 2

The data in D00000 to D00099 (augend data) is added to the data in D00100 to D00199 (addend data) and the addition results are output to D00200 to D00299. The operands of a single addition instruction are specified by index registers and the addition operations are performed by incrementing the index registers and repeatedly executing the addition instruction.



The 11-instruction subroutine on the left is equivalent to the 200-instruction subroutine on the right.



Direct Addressing of Index Registers

Index Registers can be directly addressed only in the instructions shown in the following table.

Instruction group	Instruction name	Mnemonic	Primary function
Data Movement Instructions	MOVE TO REGISTER	MOVR(560)	Stores the PLC memory address of a bit or word in an Index Register.
	MOVE TIMER/COUNTER PV TO REGISTER	MOVRW(561)	
Table Data Processing Instructions	SET RECORD LOCATION	SETR(635)	
	GET RECORD NUMBER	GETR(636)	Outputs the PLC memory address stored in an Index Register.
Data Movement Instructions	DOUBLE MOVE	MOVL(498)	Transfers between Index Registers. Used for exchanges and comparisons.
	DOUBLE DATA EXCHANGE	XCGL(562)	
Comparison Instructions	DOUBLE EQUAL	=L(301)	
	DOUBLE NOT EQUAL	< >L(306)	
	DOUBLE LESS THAN	< L(311)	
	DOUBLE LESS THAN OR EQUAL	< =L(316)	
	DOUBLE GREATER THAN	>L(321)	
	DOUBLE GREATER THAN OR EQUAL	>=L(326)	
	DOUBLE COMPARE	CMPL(060)	

Instruction group	Instruction name	Mnemonic	Primary function
Increment/Decrement Instructions	DOUBLE INCREMENT BINARY	++L(591)	Changes the PLC memory address in the Index Register by incrementing, decrementing, or offsetting its content.
	DOUBLE DECREMENT BINARY	– –L(593)	
Symbol Math Instructions	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L(401)	
	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L(411)	
Special Instructions	CONVERT ADDRESS FROM CV	FRMCV(284)	Convert actual PLC memory addresses between CV-series and CS/CJ-series addresses. (The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not this function.)
	CONVERT ADDRESS TO CV	TOCV(285)	

Note Instructions for double-length operands (i.e., those with "L" at the end) are used for index registers IR0 to IR15 because each register contains two words.

6-2-3 Processing Related to Index Registers

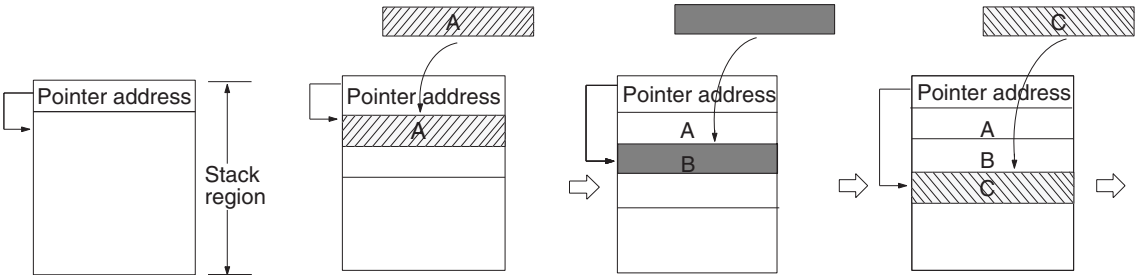
The CS/CJ-series CPU Unit's Table Data Processing instructions complement the functions of the Index Registers. These instructions can be broadly divided into the stack-processing and table-processing instructions

Processing			Purpose	Instructions
Stack processing			Operate FIFO (first-in first-out) or LIFO (last-in first-out) data tables, and read, write, insert, delete, or count data entries in data tables.	SSET(630), PUSH(632), FIFO(633), LIFO(634), SREAD(639), SWRITE(640), SINS(641), SDEL(642), SNUM(638)
Table processing	Tables with one-word records (Range instructions)	Basic processing	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), and SUM(184)
		Special processing	Perform various other table processing such as comparisons or sorting.	Combine Index Registers with instructions such as SRCH(181), MAX(182), MIN(183), and comparison instructions.
	Tables with multiple-word records (Record-table instructions)		Process data in records that are several words long.	Combine Index Registers with instructions such as DIM(631), SETR(635), GETR(636), and comparison instructions.

Stack Processing

Stack instructions act on specially defined data tables called stacks. Data can be drawn from a stack on a first-in first-out (FIFO) or last-in first-out (LIFO) basis.

A particular region of I/O memory must be defined as a stack. The first words of the stack indicate the length of the stack and contain the stack pointer. The stack pointer is incremented each time that data is written to the stack to indicate the next address where data should be stored.

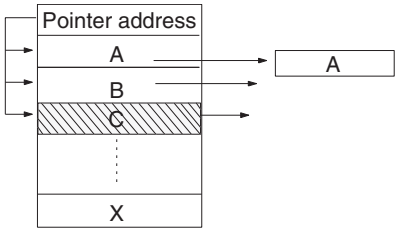


(The above diagram shows the status of the pointer data before data is added.)

Note Actually, the first two words of the stack contain the PLC memory address of the last word in the stack and the next word contains the stack pointer.

FIFO (First-in First-out) Processing

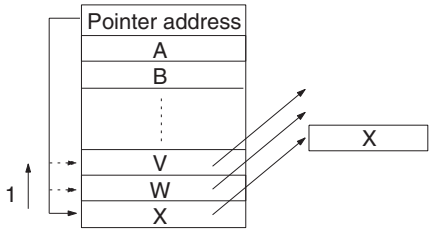
The following diagram shows the operation of a first-in first-out (FIFO) stack.



Reads the oldest word of data stored in the stack. Each time that a word is read, the pointer is decremented by one to indicate the next address for storage.

LIFO (Last-in First-out) Processing

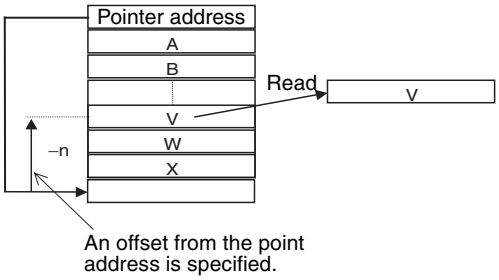
The following diagram shows the operation of a last-in first-out (LIFO) stack.



Reads most recent word of data stored in the stack. Each time that a word is read, the pointer is decremented by one to indicate the next address for storage. Data at the position that was read remains unchanged.

Manipulating Specific Table Data

Individual entries in a table can be read, writing, inserted, or deleted. The following diagram shows an example for reading.

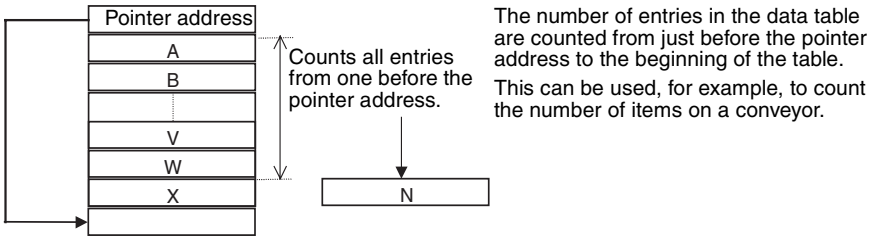


Data is read from a specific offset from the point address in the table.

Manipulating specific table data can be used, for example, in tracing items on a conveyor.

Counting Table Data

The following diagram shows how data can be counted in a data table.



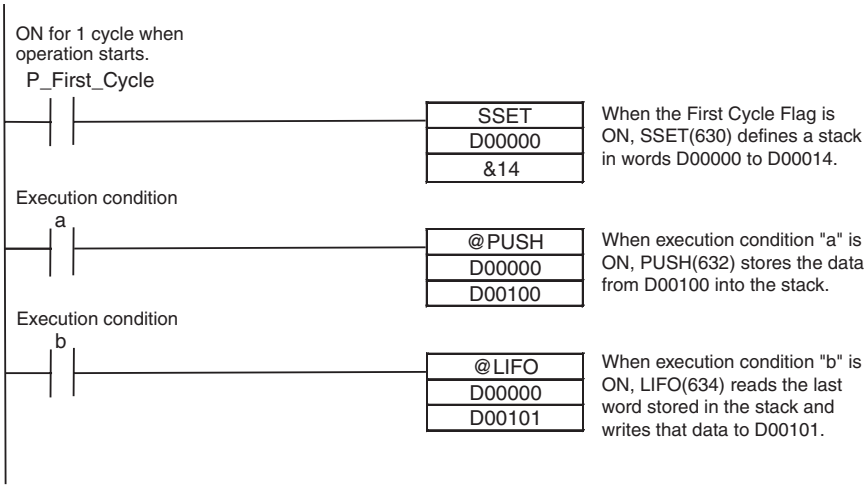
Stack Instructions

The following table lists the stack instructions and their functions. Typical applications for stacks would be processing shelf information for automatic warehousing systems, processing test results, and managing information on workpieces on a conveyor.

Instruction	Function
SSET(630)	Defines a stack region.
PUSH(632)	Stores data in the next available word in the stack.
FIFO(633)	Reads data from the stack on a first-in first-out basis.
LIFO(634)	Reads data from the stack on a last-in first-out basis.
SREAD(639)	Read a specific entry from the table.
SWRITE(640)	Writes a specific entry to the table.
SINS(641)	Inserts a specific entry in the table.
SDEL(642)	Deletes a specific entry from the table.
SNUM(638)	Counts the number of entries in the table.

Example 1:

This example defines a stack from D00000 to D00014 (which can store 10 words of data), stores data in the stack, and reads data from the stack. (LIFO(634) reads the last word stored in the stack.)



Example 2:

This example defines a stack from D00000 to D00014 (which can store 10 words of data), finds the words in the stack that match a specified value, and deletes all of the matching words.

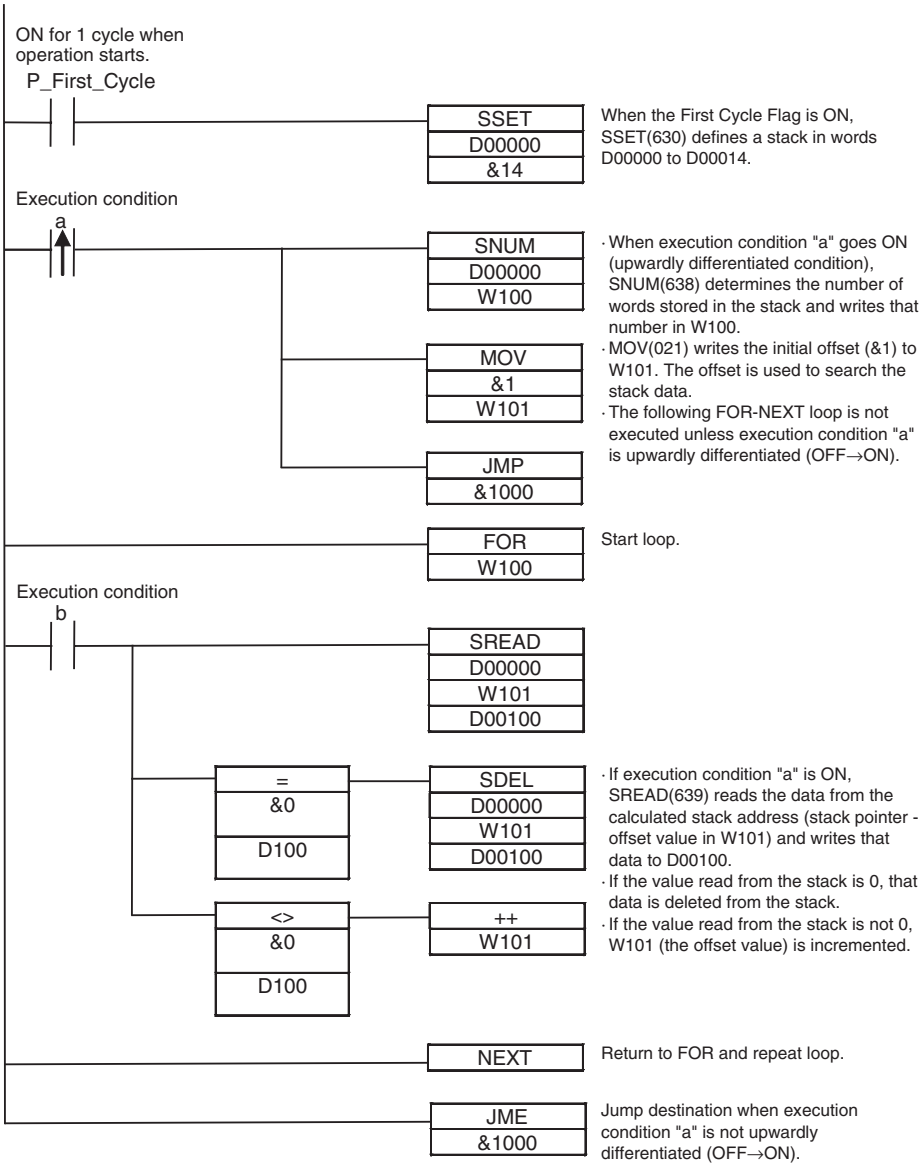
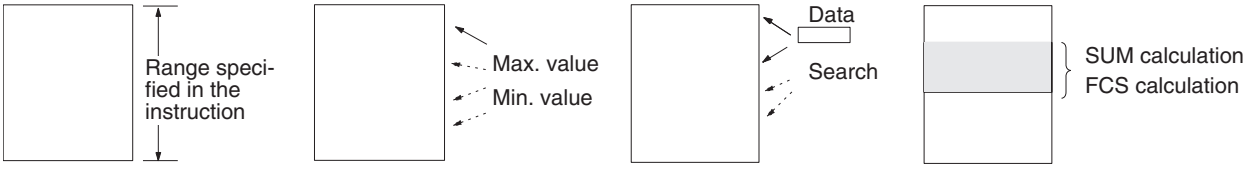


Table Processing (Range Instructions)

The range instructions act on a range of words, which can be considered a table of one-word records. These instructions perform basic operations such as finding the maximum value or minimum value in the range, search for a particular value in the range, or calculating the sum or FCS.

The PLC memory address of the result word (word containing the max. value, min. value, search data, etc.) is automatically stored in IR0. The Index Register (IR0) can be used as an operand in later instructions such as MOV(021) to read the contents of the word or perform other processing.



The following table lists the range instructions and their functions.

Instruction	Function	Description
SRCH(181)	Finds search data.	Finds the search data in the specified range and outputs the PLC memory address of the word containing that value to IR0.
MAX(182)	Finds max. value.	Finds the maximum value in the specified range and outputs the PLC memory address of the word containing that value to IR0.
MIN(183)	Finds min. value.	Finds the minimum value in the specified range and outputs the PLC memory address of the word containing that value to IR0.
SUM(184)	Calculates sum.	Calculates the sum of the data in the specified range.
FCS(180)	Calculates checksum.	Calculates the frame checksum of the data in the specified range.

The Index Registers can be combined with other instructions (such as comparison instructions) in FOR–NEXT loops to perform more complicated operations on ranges of words.

Table Processing (Record-table Instructions)

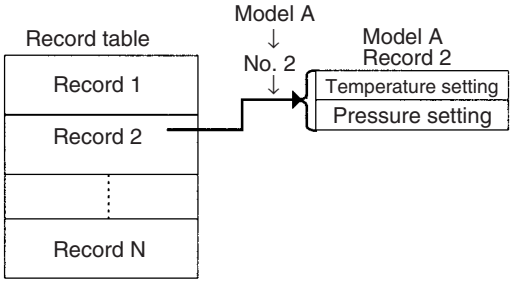
The record-table instructions act on specially defined data tables made up of equal-length records. The records can be accessed by record number for easy processing.

Instruction	Function	Description
DIM(631)	Defines a record table.	Declares the length of each record and the number of records.
SETR(635)	Sets record location.	Writes the location of the specified record (the PLC memory address of the beginning of the record) in the specified Index Register.
GETR(636)	Gets record location.	Returns the record number of the record that contains the PLC memory address in the specified Index Register.

Note Record numbers and word addresses are related through the Index Registers. Specify a record number in SETR(635) to store the PLC memory address of the beginning of that record in an Index Register. When data is required from the record, add the required offset to that Index Register to access any word in the record.

Use the record-table instructions with Index Registers to perform the following kinds of operations: reading/writing record data, searching records, sorting record data, comparing record data, and performing calculations with record data.

A typical application of record tables is storing manufacturing data for different models of a product (such as temperature and pressure settings) in record form and switching from model to model just by changing the record number.



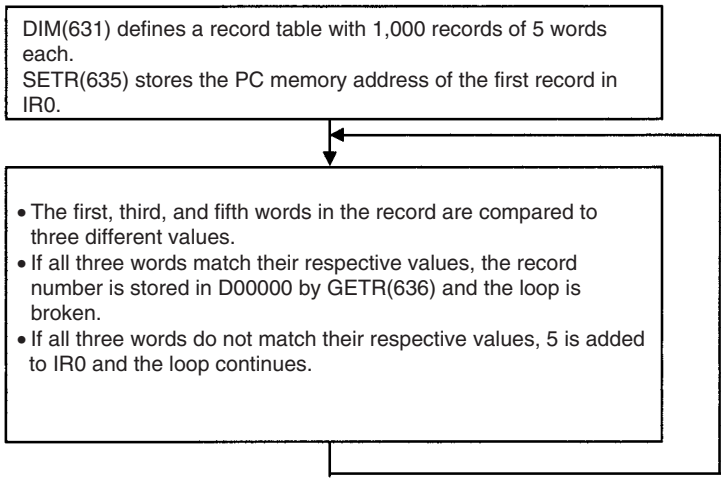
Basically, record tables are used with the following steps:

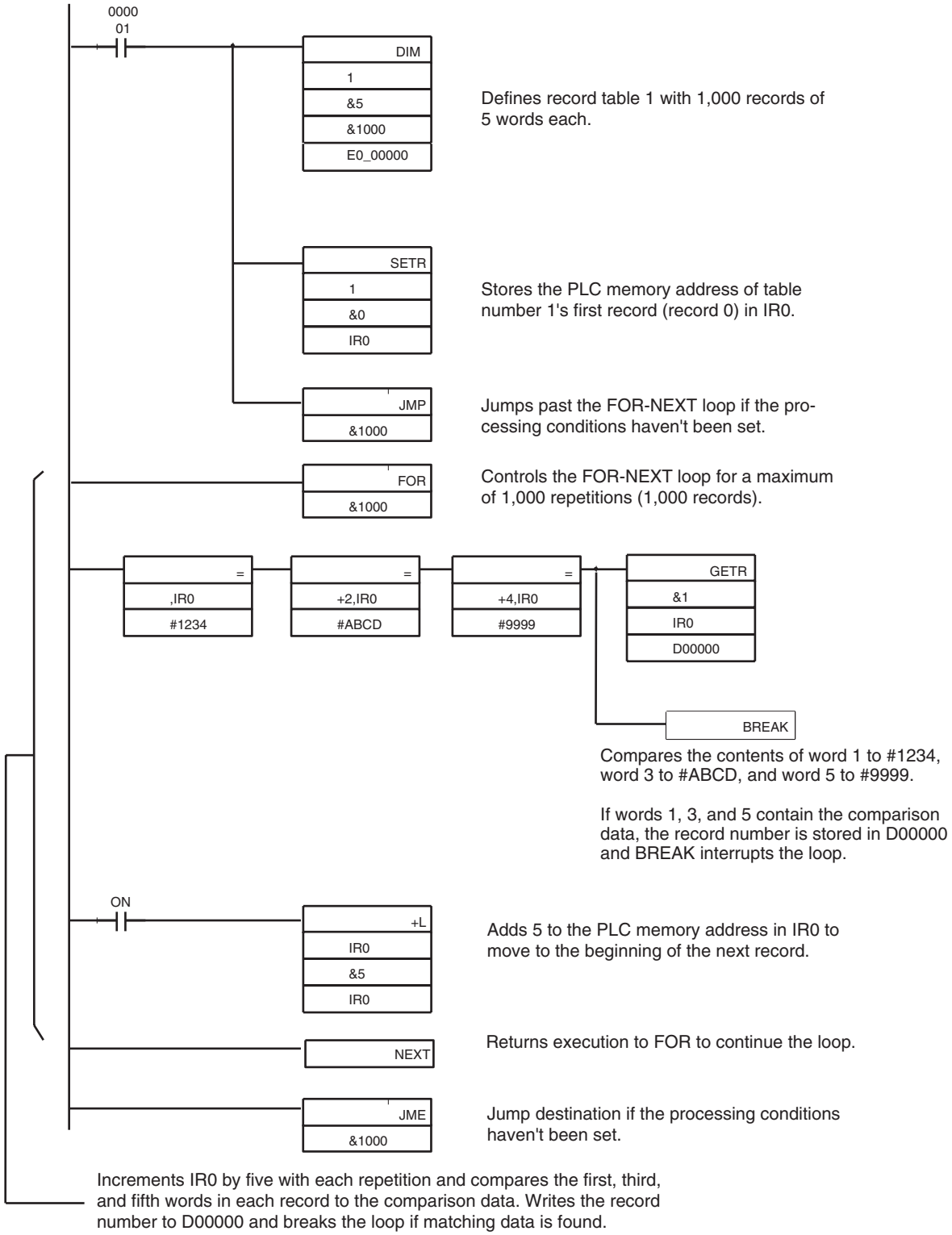
1,2,3...

1. Define the structure of the record table with DIM(631) and set the PLC memory address of a record in an Index Register with SETR(635).
2. Offset or increment the PLC memory address in the Index Register to read or compare words in the record.
3. Offset or increment the PLC memory address in the Index Register to switch to another record.
4. Repeat steps 2 and 3 as required.

Example

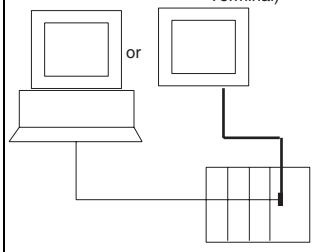
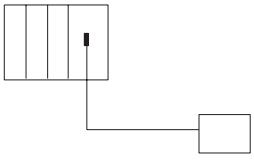
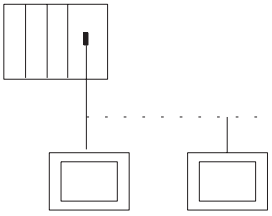
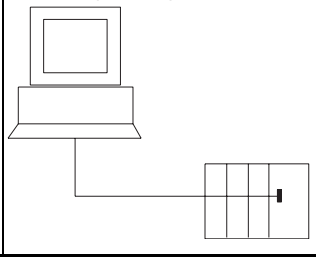
The following example uses Index Registers and the record-table instructions to compare three values to words 1, 3, and 5 in each record. If a match is found, the record number is stored in D00000.

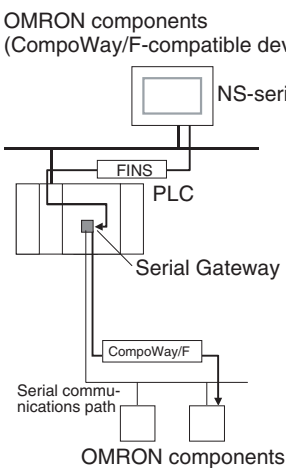
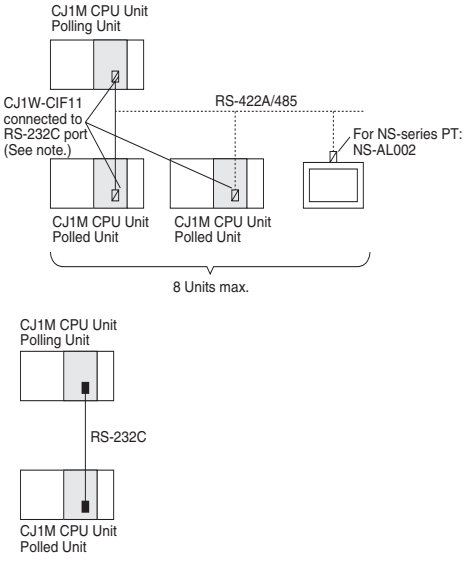



0000
01
DIM
1
&5
&1000
E0_00000
Defines record table 1 with 1,000 records of 5 words each.
SETR
1
&0
IR0
Stores the PLC memory address of table number 1's first record (record 0) in IR0.
JMP
&1000
Jumps past the FOR-NEXT loop if the processing conditions haven't been set.
FOR
&1000
Controls the FOR-NEXT loop for a maximum of 1,000 repetitions (1,000 records).
=
,IR0
#1234
=
+2,IR0
#ABCD
=
+4,IR0
#9999
GETR
&1
IR0
D00000
BREAK
Compares the contents of word 1 to #1234, word 3 to #ABCD, and word 5 to #9999.
If words 1, 3, and 5 contain the comparison data, the record number is stored in D00000 and BREAK interrupts the loop.
ON
+L
IR0
&5
IR0
Adds 5 to the PLC memory address in IR0 to move to the beginning of the next record.
NEXT
Returns execution to FOR to continue the loop.
JME
&1000
Jump destination if the processing conditions haven't been set.
Increments IR0 by five with each repetition and compares the first, third, and fifth words in each record to the comparison data. Writes the record number to D00000 and breaks the loop if matching data is found.

6-3 Serial Communications

The CS/CJ-series CPU Units support the following serial communications functions. Host link communications and no-protocol communications are described in detail later in this section.

Protocol	Connections	Description	Ports	
			Peripheral	RS-232C
Host link	Host computer OMRON PT (Programmable Terminal) or 	1) Various control commands such as reading and writing I/O memory, changing the operating mode, and force-setting/resetting bits can be executed by issuing host link commands or FINS commands from the host computer to the CPU Unit. 2) It is also possible to issue FINS commands from the CPU Unit to the host computer to send data or information. Use host link communications to monitor data such as operating status, error information, and quality data in the PLC or send data such as production planning information to the PLC.	OK	OK
No-protocol	Standard external device 	Communicate with standard devices connected to the RS-232C port without a command–response format. Instead the TXD(236) and RXD(235) instructions are executed from the program to transmit data from the transmission port or read data in the reception port. The frame headers and end codes can be specified.	Not allowed	OK
NT link 1:N or 1:1	OMRON PTs (Programmable Terminals) 	Data can be exchanged with PTs without using a communications program in the CPU Unit.	OK	OK
Peripheral bus	Programming Devices (Not Programming Consoles) 	Provides high-speed communications with Programming Devices other than Programming Consoles. (Remote programming through modems is not supported.)	OK	OK

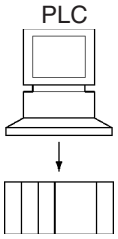
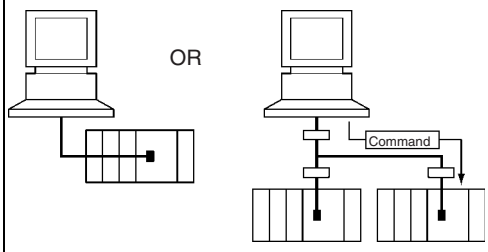
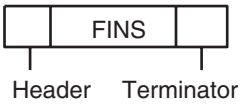
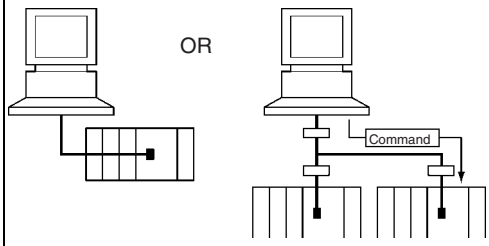
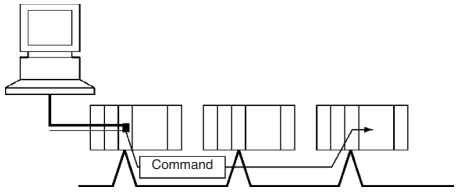
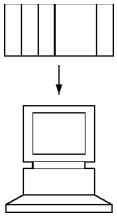
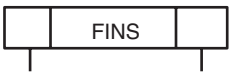
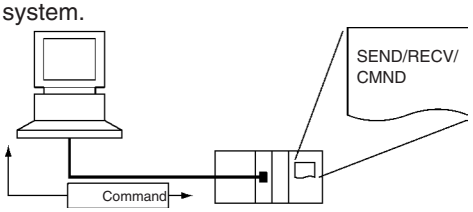
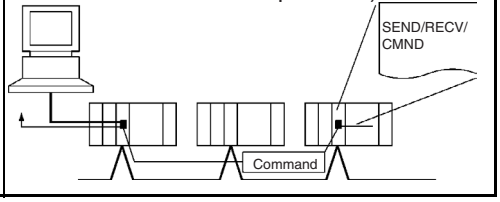
Protocol	Connections	Description	Ports	
			Peripheral	RS-232C
Serial Gateway (conversion to CompoWay/F) (unit version 3.0 or later only)		Converts received FINS commands into CompoWay/F commands and transfers them on the serial communications path.	OK	OK
Serial PLC Links (CJ1M only)		Up to ten words per Unit can be shared by up to nine CPU Units, including one Polling Unit and eight Polled Units. An RS-422A Converter can be connected to the RS-232C port on each CPU Unit to communicate via RS-422A/485, or two CPU Units can communicate via an RS-232C connection. The Serial PLC Links can also include PTs as Polled Units via NT Links (1:N) combined with CJ1M CPU Units.	Not allowed	OK

Here, we will describe Host Link and No-protocol communications.

Note The CJ1W-CIF11 is not insulated and the total transmission distance is 50 meters max. If the total transmission distance is greater than 50 meters, use the insulated NT-AL001 and do not use the CJ1W-CIF11. If only the NT-AL001 is used, the total transmission distance can is 500 meters max.

6-3-1 Host Link Communications

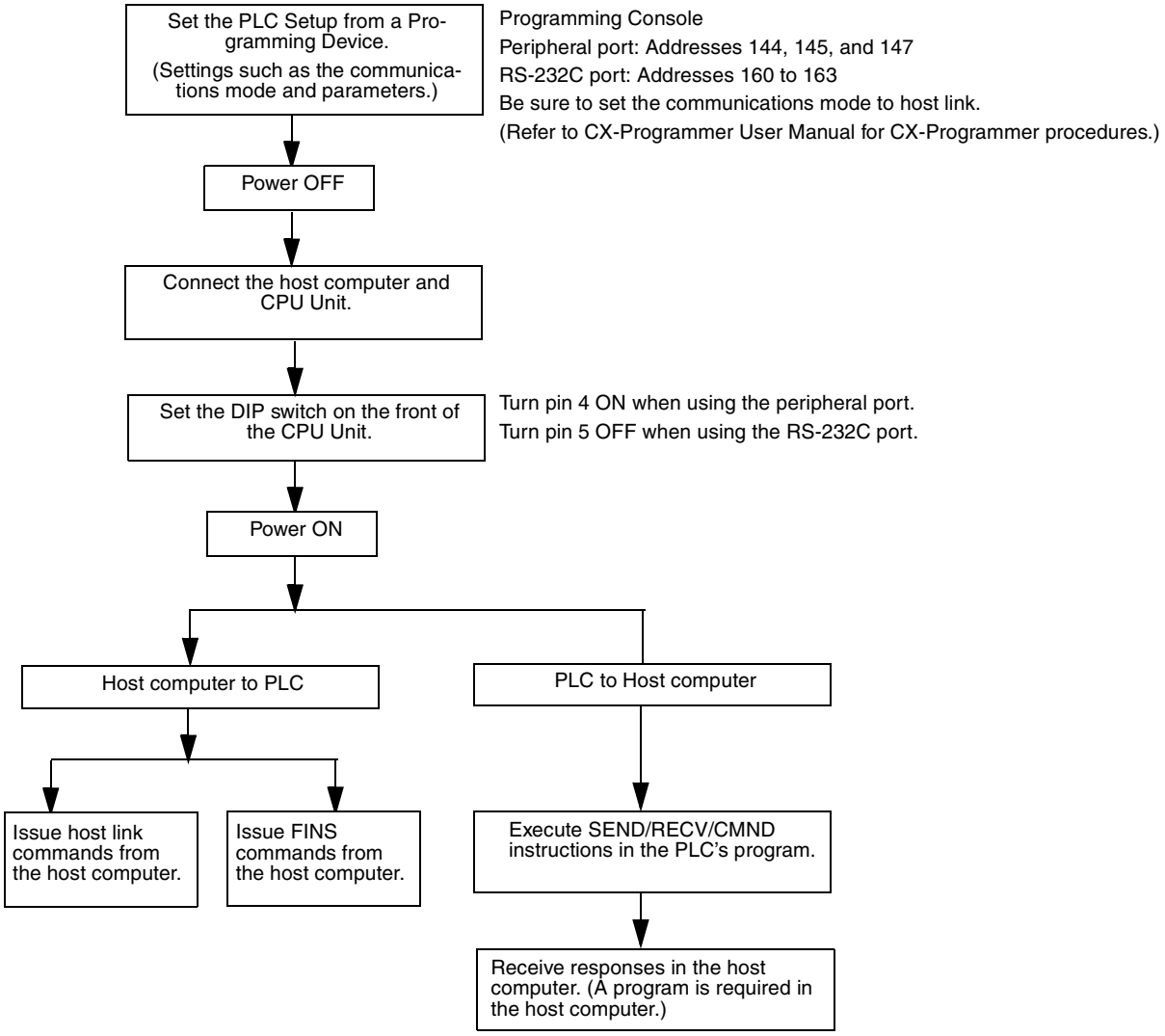
The following table shows the host link communication functions available in CS/CJ PLCs. Select the method that best suits your application.

Command flow	Command type	Communications method	Configuration
Host computer PLC 	Host link command Host link command	Create frame in the host computer and issue command to the PLC. Receive the response from the PLC. Application: Use this method when communicating primarily from the host computer to the PLC.	Directly connect the host computer in a 1:1 or 1:N system. 
	FINS command[1] (with host link header and terminator) 	Create frame in the host computer and issue command to the PLC. Receive the response from the PLC. Application: Use these methods when communicating primarily from the host computer to PLCs in the network.	Directly connect the host computer in a 1:1 or 1:N system. 
			Communicate with other PLCs in the network from the host computer. (Convert from host link to network protocol.) 
PLC Host computer 	FINS command[2] (with host link header and terminator) 	Issue frame with the CPU Unit's SEND/RECV/CMND instructions. Receive response from the host computer. Application: Use this method when communicating primarily from the PLC to the host computer to transmit status information such as error information.	Directly connect the host computer in a 1:1 system. 
			Communicate with the host computer through other PLCs in the network. (Convert from host link to network protocol.) 

Note 1. The FINS command must have a host link header and terminator attached before it is transmitted from the host computer.

2. The FINS command is transmitted from the PLC with a host link header and terminator attached. A program must be prepared in the host computer to analyze the FINS commands and return the proper responses.

Procedure



Host Link Commands

The following table lists the host link commands. Refer to the *C-series Host Link Units System Manual (W143)* for more details.

Header code	Name	Function
RR	CIO AREA READ	Reads the contents of the specified number of CIO Area words, starting from the specified word.
RL	LINK AREA READ	Reads the contents of the specified number of Link Area words, starting from the specified word.
RH	HR AREA READ	Reads the contents of the specified number of Holding Area words, starting from the specified word.
RC	PV READ	Reads the contents of the specified number of timer/counter PVs (present values), starting from the specified timer/counter.
RG	T/C STATUS READ	Reads the status of the Completion Flags of the specified number of timers/counters, starting from the specified timer/counter.
RD	DM AREA READ	Reads the contents of the specified number of DM Area words, starting from the specified word.
RJ	AR AREA READ	Reads the contents of the specified number of Auxiliary Area words, starting from the specified word.

Header code	Name	Function
RE	EM AREA READ	Reads the contents of the specified number of EM Area words, starting from the specified word.
WR	CIO AREA WRITE	Writes the specified data (word units only) to the CIO Area, starting from the specified word.
WL	LINK AREA WRITE	Writes the specified data (word units only) to the Link Area, starting from the specified word.
WH	HR AREA WRITE	Writes the specified data (word units only) to the Holding Area, starting from the specified word.
WC	PV WRITE	Writes the PVs (present values) of the specified number of timers/counters, starting from the specified timer/counter.
WD	DM AREA WRITE	Writes the specified data (word units only) to the DM Area, starting from the specified word.
WJ	AR AREA WRITE	Writes the specified data (word units only) to the Auxiliary Area, starting from the specified word.
WE	EM AREA WRITE	Writes the specified data (word units only) to the EM Area, starting from the specified word.
R#	SV READ 1	Reads the 4-digit BCD constant or word address in the SV of the specified timer/counter instruction.
R$	SV READ 2	Searches for the specified timer/counter instruction beginning at the specified program address and reads the 4-digit constant or word address in the SV.
R%	SV READ 3	Searches for the specified timer/counter instruction beginning at the specified program address and reads the 4-digit BCD constant or word address in the SV.
W#	SV CHANGE 1	Changes the 4-digit BCD constant or word address in the SV of the specified timer/counter instruction.
W$	SV CHANGE 2	Searches for the specified timer/counter instruction beginning at the specified program address and changes the 4-digit constant or word address in the SV.
W%	SV CHANGE 3	Searches for the specified timer/counter instruction beginning at the specified program address and changes the 4-digit constant or word address in the SV.
MS	STATUS READ	Reads the operating status of the CPU Unit (operating mode, force-set/reset status, fatal error status).
SC	STATUS CHANGE	Changes the CPU Unit's operating mode.
MF	ERROR READ	Reads and clears errors in the CPU Unit (non-fatal and fatal).
KS	FORCE SET	Force-sets the specified bit.
KR	FORCE RESET	Force-resets the specified bit.
FK	MULTIPLE FORCE SET/RESET	Force-sets, force-resets, or clears the forced status of the specified bits.
KC	FORCE SET/RESET CANCEL	Cancels the forced status of all force-set and force-reset bits.
MM	PLC MODEL READ	Reads the model type of the PLC.
TS	TEST	Returns, unaltered, one block of data transmitted from the host computer.
RP	PROGRAM READ	Reads the contents of the CPU Unit's user program area in machine language (object code).
WP	PROGRAM WRITE	Writes the machine language (object code) program transmitted from the host computer into the CPU Unit's user program area.
MI	I/O TABLE GENERATE	Creates a registered I/O table with the actual I/O table.
QQMR	COMPOUND COMMAND	Registers the desired bits and words in a table.
QQIR	COMPOUND READ	Reads the registered words and bits from I/O memory.
XZ	ABORT (command only)	Aborts the host link command that is currently being processed.

Header code	Name	Function
**	INITIALIZE (command only)	Initializes the transmission control procedure of all PLCs connected to the host computer.
IC	Undefined command (response only)	This response is returned if the header code of a command was not recognized.

FINS Commands

The following table lists the FINS commands. Refer to the *FINS Commands Reference Manual (W227)* for more details.

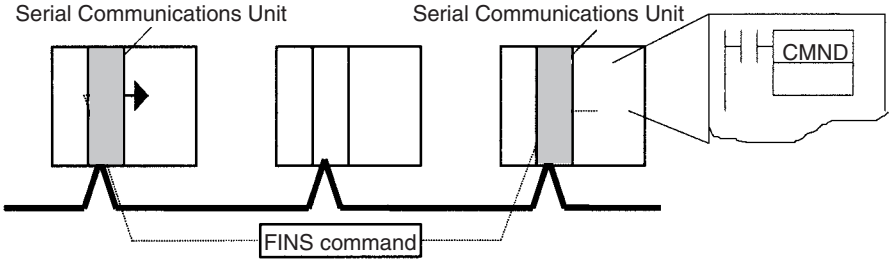
Type	Command code		Name	Function
I/O Memory Area Access	01	01	MEMORY AREA READ	Reads consecutive data from the I/O memory area.
	01	02	MEMORY AREA WRITE	Writes consecutive data to the I/O memory area.
	01	03	MEMORY AREA FILL	Fills the specified range of I/O memory with the same data.
	01	04	MULTIPLE MEMORY AREA READ	Reads non-consecutive data from the I/O memory area.
	01	05	MEMORY AREA TRANSFER	Copies and transfers consecutive data from one part of the I/O memory area to another.
Parameter Area Access	02	01	PARAMETER AREA READ	Reads consecutive data from the parameter area.
	02	02	PARAMETER AREA WRITE	Writes consecutive data to the parameter area.
	02	03	PARAMETER AREA FILL	Fills the specified range of the parameter area with the same data.
Program Area Access	03	06	PROGRAM AREA READ	Reads data from the user program area.
	03	07	PROGRAM AREA WRITE	Writes data to the user program area.
	03	08	PROGRAM AREA CLEAR	Clears the specified range of the user program area.
Execution Control	04	01	RUN	Switches the CPU Unit to RUN, MONITOR, or DEBUG mode.
	04	02	STOP	Switches the CPU Unit to PROGRAM mode.
Configuration Read	05	01	CONTROLLER DATA READ	Reads CPU Unit information.
	05	02	CONNECTION DATA READ	Reads the model numbers of the specified Units.
Status Read	06	01	CONTROLLER STATUS READ	Reads the CPU Unit's status information.
	06	20	CYCLE TIME READ	Reads the average, maximum, and minimum cycle times.
Clock Access	07	01	CLOCK READ	Reads the clock.
	07	02	CLOCK WRITE	Sets the clock.
Message Access	09	20	MESSAGE READ/CLEAR	Reads/clears messages and FAL(S) messages.
Access Right	0C	01	ACCESS RIGHT ACQUIRE	Acquires the access right if no other device holds it.
	0C	02	ACCESS RIGHT FORCED ACQUIRE	Acquires the access right even if another device currently holds it.
	0C	03	ACCESS RIGHT RELEASE	Releases the access right regardless of what device holds it.
Error Access	21	01	ERROR CLEAR	Clears errors and error messages.
	21	02	ERROR LOG READ	Reads the error log.
	21	03	ERROR LOG CLEAR	Clears the error log pointer to zero.

Type	Command code		Name	Function
File Memory	22	01	FILE NAME READ	Reads the file memory's file information.
	22	02	SINGLE FILE READ	Reads the specified amount of data from the specified point in a file.
	22	03	SINGLE FILE WRITE	Writes the specified amount of data from the specified point in a file.
	22	04	FILE MEMORY FORMAT	Formats file memory.
	22	05	FILE DELETE	Deletes the specified files from file memory.
	22	07	FILE COPY	Copies a file within file memory or between two file memory devices in a system.
	22	08	FILE NAME CHANGE	Changes a file name.
	22	0A	I/O MEMORY AREA FILE TRANSFER	Transfers or compares data between the I/O memory area and file memory.
	22	0B	PARAMETER AREA FILE TRANSFER	Transfers or compares data between the parameter area and file memory.
	22	0C	PROGRAM AREA FILE TRANSFER	Transfers or compares data between the program area and file memory.
	22	15	CREATE/DELETE DIRECTORY	Creates or deletes a directory.
Forced Status	23	01	FORCED SET/RESET	Force-sets, force-resets, or clears the forced status of the specified bits.
	23	02	FORCED SET/RESET CANCEL	Cancels the forced status of all force-set and force-reset bits.

Message Communications Functions

The FINS commands listed in the table above can also be transmitted through the network from other PLCs to the CPU Unit. Observe the following points when transmitting FINS commands through the network.

- CPU Bus Units (such as Controller Link Units or Ethernet Units) must be mounted in the local PLC and destination PLC to transmit FINS commands.
- FINS commands are issued with CMND(490) from the CPU Unit's program.
- FINS commands can be transmitted across up to eight networks for the CS/CJ-series CPU Units Ver. 2.0 or later and across up to three networks for other CPU Units. The networks can be the same type or different types.



Refer to the CPU Bus Unit's Operation Manual for more details on the message communications functions.

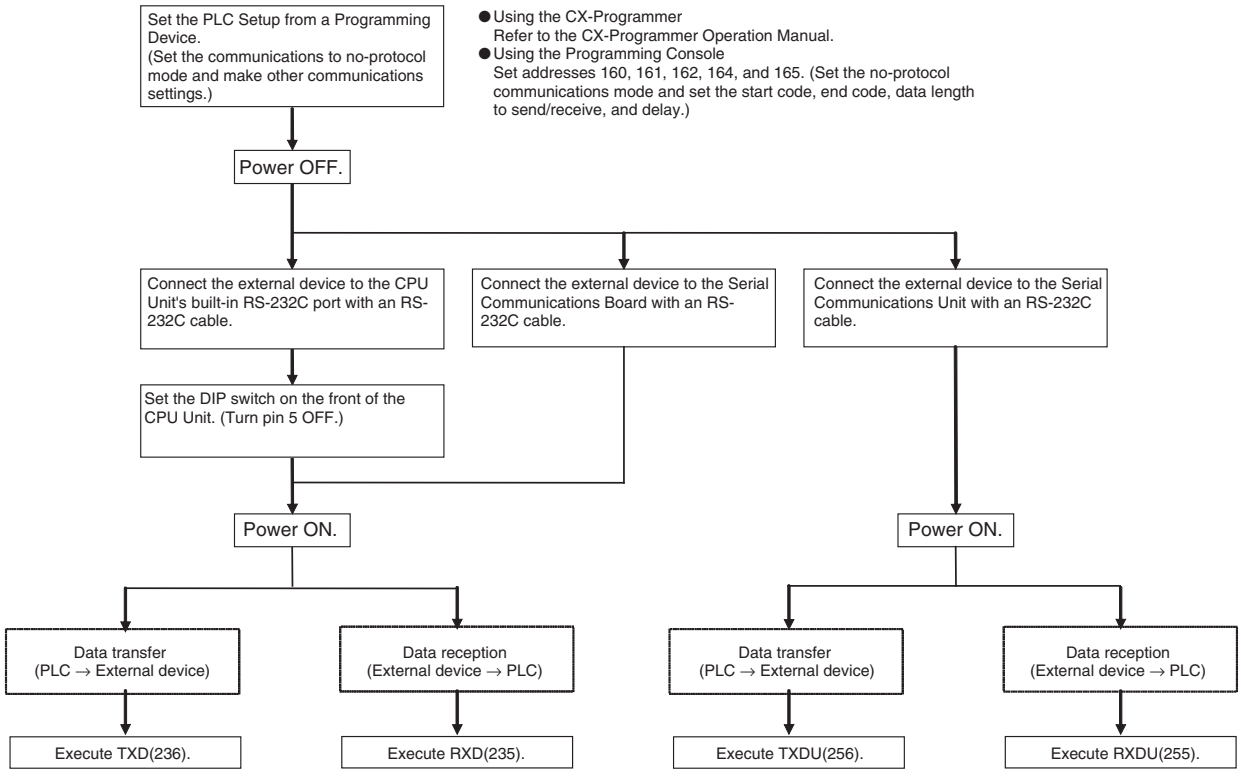
6-3-2 No-protocol Communications

The following table lists the no-protocol communication functions available in CS/CJ PLCs.

Transfer direction	Method	Max. amount of data	Frame format		Other functions
			Start code	End code	
Data transmission (PLC → External device)	Execution of TXD(236)/ TXDU(256) in the program*	256 bytes	Yes: 00 to FF No: None	Yes: 00 to FF or CR+LF No: None	Send delay time (delay between execution and sending data from specified port): 0 to 99,990 ms (unit: 10 ms)
Data reception (External device → PLC)	Execution of RXD(235)/ RXDU(255) in the program	256 bytes			---

Note This function is supported by CPU Units with unit version 3.0 or later or Serial Communications Units/Boards with unit version 1.2 or later.

Procedure



Message Frame Formats

Data can be placed between a start code and end code for transmission by TXD(236) or TXDU(256) and frames with that same format can be received by RXD(235) or RXDU(255). When transmitting with TXD(236) or TXDU(256), just the data from I/O memory is transmitted, and when receiving with RXD(235) or RXDU(255), just the data itself is stored in I/O memory. Up to 256 bytes (including the start and end codes) can be transferred in no-protocol mode.

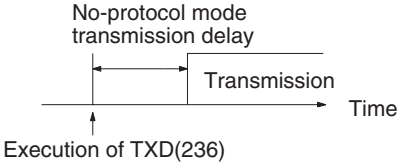
The following table shows the message formats that can be set for transmissions and receptions in no-protocol mode. The format is determined by the start code (ST) and end code (ED) settings in the PLC Setup.

Start code setting	End code setting		
	No	**Yes**	**CR+LF**
No	data (*data*: 256 bytes max.)	*data*+ED (*data*: 255 bytes max.)	*data*+CR+LF (*data*: 254 bytes max.)
Yes	ST+*data* (*data*: 255 bytes max.)	ST+*data*+ED (*data*: 254 bytes max.)	ST+*data*+CR+LF (*data*: 253 bytes max.)

- When more than one start code is used, the first start code will be effective.
- When more than one end code is used, the first end code will be effective.

Note

1. If the data being transferred contains the end code, the data transfer will be stopped midway. In this case, change the end code to CR+LF.
2. There is a setting in the PLC Setup (address 162: no-protocol mode delay) that will delay the transmission of data after the execution of TXD(236)/ TXDU(256).



Refer to the *CJ-series Programmable Controllers Instructions Reference Manual (W340)* for more details

6-3-3 NT Link (1:N Mode)

In the CS/CJ Series, communications are possible with PTs (Programmable Terminals) using NT Links (1:N mode).

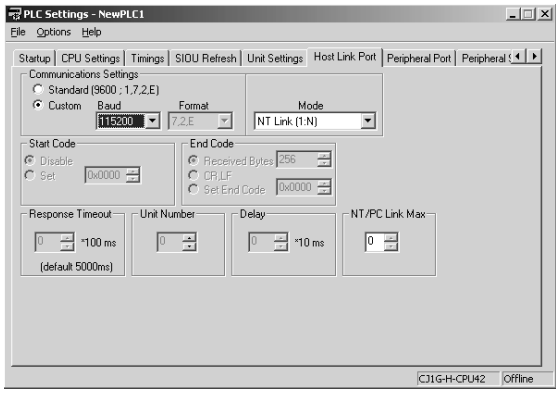
Note Communications are not possible using the 1:1-mode NT Link protocol.

High-speed NT Links are possible in addition to the previous standard NT Links by using the PT system menu and the following PLC Setup settings (not supported by CS-series pre-EV1 CS1 CPU Units). High-speed NT Links are possible, however, only with the NT31(C)-V2 or NT631(C)-V2 PTs.

Note The CS1G/H-CPU□□-V CPU Units do not support this function.

PLC Setup

When making settings with the CX-Programmer, set the baud rate to 115,200 bps for a high-speed NT Link or 38,400 bps for a standard NT Link.



The following table shows the corresponding settings when using a Programming Console.

Communications port	Programming Console address	Name	Settings contents	Default values	Other conditions
Peripheral port	+144, bits 8 to 11	Serial communications mode	02 Hex: NT Link (1:N mode)	00 Hex: Host Link	Turn ON pin 4 on the CPU Unit DIP switch.
	+145, bits 0 to 7	Baud rate	00 to 09 Hex: Standard NT Link 0A Hex: High-speed NT Link	00 Hex: Standard NT Link	
	+150, bits 0 to 3	NT Link mode maximum unit number	0 to 7 Hex	0 Hex (Max. unit No. 0)	---
RS-232C port	+160, bits 8 to 11	Serial communications mode	02 Hex: NT Link (1:N mode)	00 Hex: Host Link	Turn OFF pin 5 on the CPU Unit DIP switch.
	+161, bits 0 to 7	Baud rate	00 to 09 Hex: Standard NT Link 0A Hex: High-speed NT Link	00 Hex: Standard NT Link	
	+166, bits 0 to 3	NT Link mode maximum unit number	0 to 7 Hex	0 Hex (Max. unit No. 0)	---

PT System Menu

Set the PT as follows:

1,2,3...

1. Select NT Link (1:N) from Comm. A Method or Comm. B Method on the Memory Switch Menu under the System Menu on the PT Unit.
2. Press the SET Touch Switch to set the Comm. Speed to High Speed.

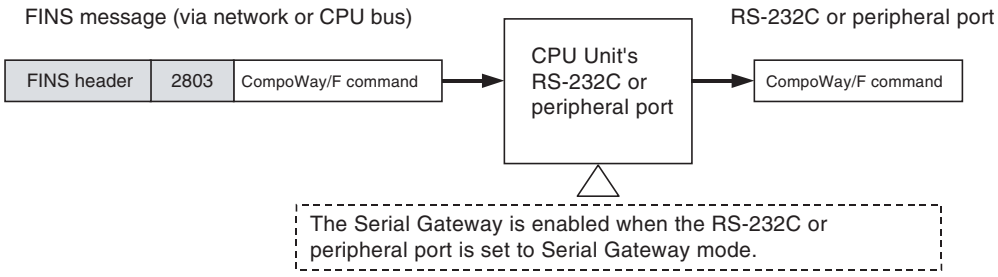
6-3-4 CPU Unit's Serial Gateway

Serial Gateway Overview

FINS messages (commands) that are received are automatically converted into the specified protocol and then sent via serial communications. The responses are also automatically converted. When the CPU Unit's RS-232C port or peripheral port is used, FINS messages can be converted into the following protocol.

- CompoWay/F

The Serial Gateway is enabled when the serial communications mode is set to Serial Gateway.



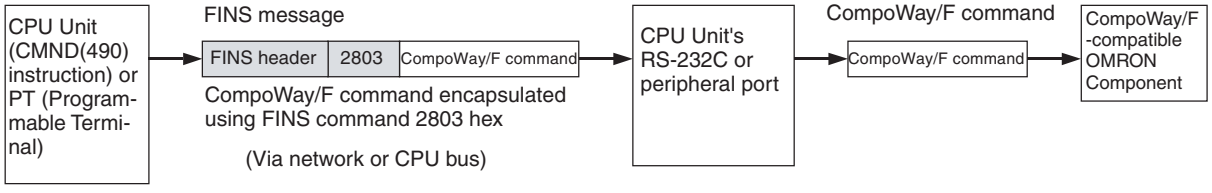
Serial Gateway Specifications

Item	Details
Conversion source	FINS command (received via FINS network, Host Link FINS, peripheral bus, NT links, or CPU bus)
Conversion function	Received FINS commands are converted according to the following values before sending to the serial port (peripheral port or RS-232C port) of the CPU Unit. 2803 hex: Removes FINS header and converts to CompoWay/F command
After conversion	CompoWay/F commands
Serial communications method	1:N half-duplex communications
Maximum number of connected Units	31 slaves
Supported serial communications modes	Serial Gateway mode
Response timeout monitoring	The time is monitored from when a message converted to CompoWay/F protocol using the Serial Gateway is sent until a response is received (enabled in Serial Gateway mode or protocol macro mode). Default: 5 s; Setting range: 0.1 to 25.5 s **Note** If a timeout occurs, the FINS end code is returned to the source of the FINS command (0205 hex: Response timeout).
Send delay	None

Converting FINS to CompoWay/F

OMRON Components connected serially to the CPU Unit's RS-232C port or peripheral port via CompoWay/F can be accessed from the PLC or PT using CompoWay/F commands enclosed in FINS messages.

- Sent FINS message: FINS header + FINS command code 2803 hex + CompoWay/F command
- Message after conversion: CompoWay/F command

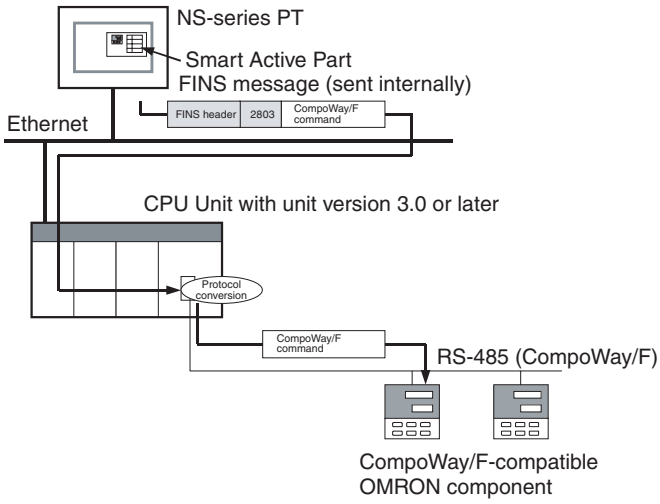


For details on FINS command code 2803 hex, refer to the *Communications Commands Reference Manual* (W342).

System Configuration Patterns

Executing Smart Active Parts Using an NS-series PT (Sending Internal FINS Messages)

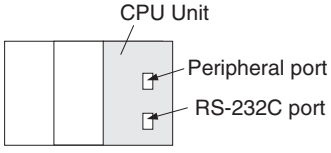
The CPU Unit converts the FINS messages to CompoWay/F protocol for sending in this operation.

Access from PT on Ethernet or serial NT Link	Details	Routing tables to treat serial communications path as network
	Access via serial communications using CompoWay/F is possible from a PT connected to the network by executing a Smart Active Part that is connected serially, which automatically sends an internal FINS command.	Optional

Note When the NS-series PT is connected serially to the PLC using serial communications mode (1:N NT Links), and the NS-series PT sends FINS commands encapsulated in NT Link commands using Smart Active Parts, the CPU Unit removes the NT Link header, etc. from the received command, converting it to a FINS command, and transfers the command to the CPU Unit's serial port. The CPU Unit uses the Serial Gateway to convert the command into the specified protocol. This operation enables serially connected devices to access the CPU Unit's serial port from Smart Active Parts using an NS-series PT.

Note 1. The FINS header contains the following information.

- Remote destination network address (DNA)
  - With routing tables that treat serial communications path as a network: Network address corresponding to serial port in the routing tables.
  - Without routing tables that treat serial communications path as a network: Network address for specifying actual remote PLC.
- Remote destination node address (DA1)
  - With routing tables that treat serial communications path as a network: 00 hex (local PLC's internal communications)
  - Without routing tables that treat serial communications path as a network: Node address for specifying actual remote PLC
- Remote destination unit address (DA2)
  Unit address of serial port



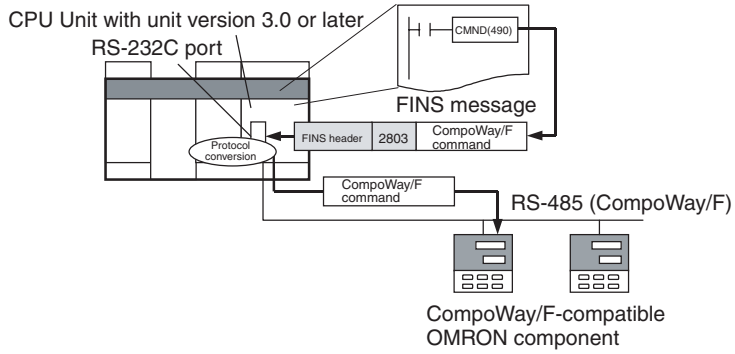
CPU Unit's serial port	Serial port's unit address
Peripheral port	FD hex (253 decimal)
RS-232C port	FC hex (252 decimal)

2. The contents of the CompoWay/F command enclosed in the FINS message that is sent is as follows:
   Node number + subaddress + SID + command text (ASCII must be used.)
   STX, ETX+BCC are not required when sending FINS. They are added automatically for serial communications.

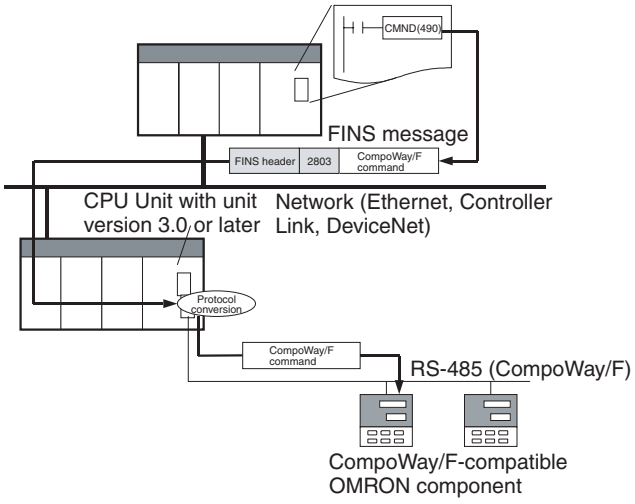
Sending FINS Messages Using CMND(490) in CPU Unit's Ladder Program

The CPU Unit converts the FINS messages to CompoWay/F protocol for sending in this operation.

Access from CPU Unit (on the Same PLC)

Serial conversion	Details	Routing tables to treat serial communications path as network
	OMRON components connected serially to the CPU Unit's RS-232C port using CompoWay/F can be accessed from a CPU Unit in the same PLC.	Optional

Access from CPU Unit (PLC on the Network)

FINS message-to-serial	Details	Routing tables to treat serial communications path as network
	OMRON components connected serially to the CPU Unit's RS-232C port using CompoWay/F can be accessed from a CPU Unit in a PLC connected to the network.	Optional

Communications Frames

Command Frame

Frame before Conversion

FINS header				FINS command		CompoWay/F (See note.)				
Remote network address (DNA)	Remote node address (DA1)	Remote unit address (DA2)	Etc.	MRC	SRC					
Serial port allocated address	00 hex	Serial port unit address		28	03	Node No. ($\times 10^1$) ($\times 10^2$) (2 bytes ASCII)	Sub-address "00" (ASCII 3030 hex) etc.	SID "0" (ASCII 30 hex)	Command (MRC, SRC) (4 bytes ASCII)	Text (ASCII)
or local network address	or local network node address									

Frame after Conversion

CompoWay/F							
STX (02 hex)	Node No. ($\times 10^1$) ($\times 10^2$) (2 bytes ASCII)	Sub-address "00" (ASCII 3030 hex) etc.	SID "0" (ASCII 30 hex)	Command (MRC, SRC) (4 bytes ASCII)	Text (ASCII)	ETX (03 hex)	BCC

Note CompoWay/F commands use ASCII as the transmission code. Therefore, be sure to use ASCII for the CompoWay/F command after the FINS command code 2803 hex (from node number to text) using CMND(490) or other instruction.

Example: If the CompoWay/F command MRC SRC is “01” “02” (where the quotation marks (“”) indicate ASCII characters), 0, 1, 0, 2 must be treated as ASCII characters. Therefore, set “01” as 3031 hex (not 01 hex), and “02” as 3032 hex (not 02 hex).

Further, to write the CompoWay/F command to the command storage area using CMND(490) in frame order (without creating empty bytes), the SID component of the CompoWay/F command requires 1 byte of ASCII as 30 hex, so the subsequent components (s+3 and afterwards) must be set in one byte each.

Frame before Conversion

CompoWay/F								
STX (02 hex)	Node No. ($\times 10^1$) ($\times 10^2$) (2 bytes ASCII)	Sub-address "00" (ASCII 3030 hex) etc.	End code (2 bytes ASCII)	Command (MRC, SRC) (4 bytes ASCII)	Response (MRES, SRES) (4 bytes ASCII)	Text (ASCII)	ETX (03 hex)	BCC

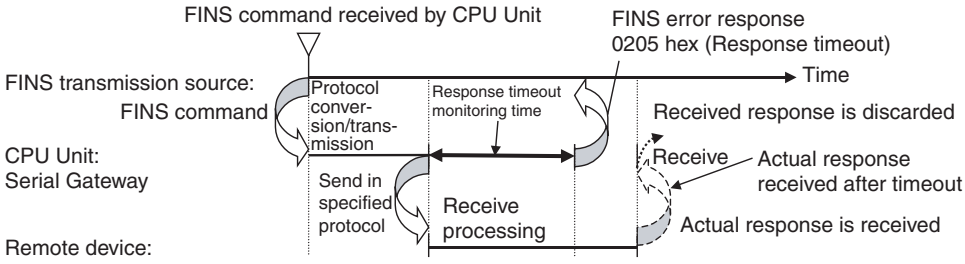
Frame after Conversion

FINS header			FINS command		FINS end code		CompoWay/F (See note.)					
Remote network address (DNA)	Remote node address (DA1)	Remote unit address (DA2) etc.	MRC	SRC	MRES	SRES						
Serial port allocated address	00 hex	Serial port unit address	28	03	User-specified	User-specified	Node No. ($\times 10^1$) ($\times 10^2$) (2 bytes ASCII)	Sub-address "00" (ASCII 3030 hex) etc.	End code (2 bytes ASCII)	Command (MRC, SRC) (4 bytes ASCII)	Response (MRES, SRES) (4 bytes ASCII)	Text (ASCII)

Response Timeout Monitoring (Serial Gateway Mode)

During Serial Gateway mode the time is monitored from when the message converted into the specified protocol by Serial Gateway is sent until a response is received from the remote device. (The default is 5 s. The setting range for a user-specified value is between 0.1 and 25.5 s.)

If a response is not received at the serial port within the set time, a FINS error response is returned to the source of the FINS command (end code: 0205 hex (response timeout)). If a response is received after the timeout has occurred, however, the response received for each protocol is discarded and a FINS response is not returned to the source of the FINS command.



6-3-5 Serial PLC Links (CJ1M CPU Units Only)

Overview

Serial PLC Links are supported by CJ1M CPU Units only. They allow data to be exchanged among CJ1M CPU Units via the built-in RS-232C ports without requiring special programming. Words are allocated in memory in the Serial PLC Link Words (CIO 3100 to CIO 3199). RS-232C connections can be used between CPU Units, or RS-422A/485 connections can be used by connecting RS-232C-to-RS-422A/485 converters to the RS-232C ports. CJ1W-CIF11 RS-422A Converters can be used to convert between RS-232C and RS-422A/485.

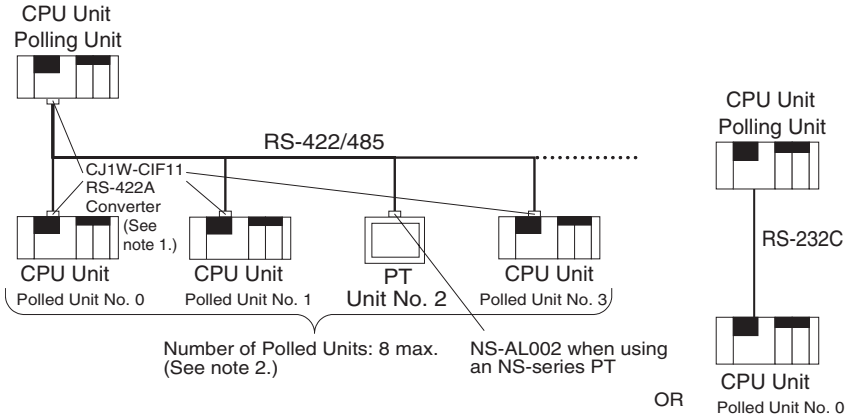
A PT that is set for NT Link (1:N) communications can also be used together on the same network. The polled PT uses the network to communicate in an NT link (1:N) with the polling CPU Unit. When a PT is connected, however, the addresses in the Serial PLC Link Words corresponding to the PT's unit number are undefined.

When using RS-422A/485, 2-wire connections, PTs set for NT Links cannot be used together on the same network.

Specifications

Item	Specifications
Connection method	RS-422A/485 or RS-232C connection via the CPU Unit's RS-232C port.
Allocated data area	Serial PLC Link Words: CIO 3100 to CIO 3199 (Up to 10 words can be allocated for each CPU Unit.)
Number of Units	9 Units max., comprising 1 Polling Unit and 8 Polled Units (A PT can be placed on the same network in an NT Link (1:N), but it must be counted as one of the 8 Polled Units.)

System Configuration



Note

1. The CJ1W-CIF11 is not insulated and the total transmission distance is 50 meters max. If the total transmission distance is greater than 50 meters, use the insulated NT-AL001 and do not use the CJ1W-CIF11. If only the NT-AL001 is used, the total transmission distance can is 500 meters max.
2. Up to 8 Units, including the PT and Polled Units, can be connected to the Polling Unit when a PT set for Serial PLC Link communications is on the same network.

Data Refresh Methods

The following two methods can be used to refresh data.

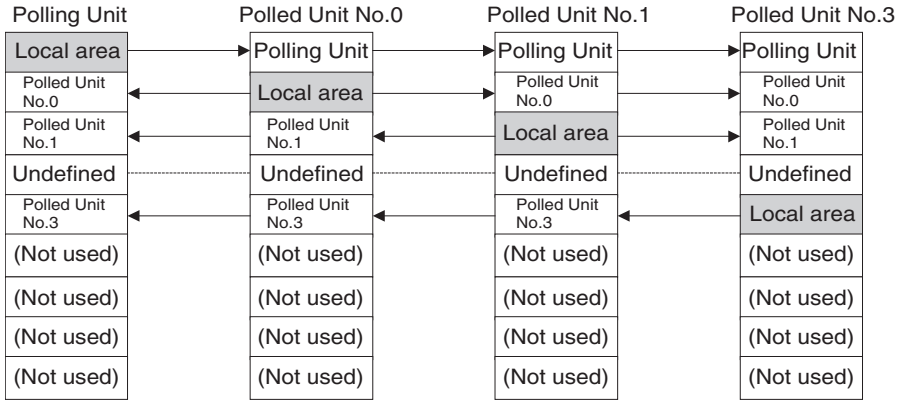
- Complete link method
- Polling Unit link method

Complete Link Method

The data from all nodes in the Serial PLC Links are reflected in both the Polling Unit and the Polled Units. (The only exceptions are the address allocated to the connected PT's unit number and the addresses of Polled Units that are not present in the network. These data areas are undefined in all nodes.)

Example: Complete link method, highest unit number: 3.

In the following diagram, Polled Unit No. 2 is either a PT or is a Unit not present in the network, so the area allocated for Polled Unit No. 2 is undefined in all nodes.



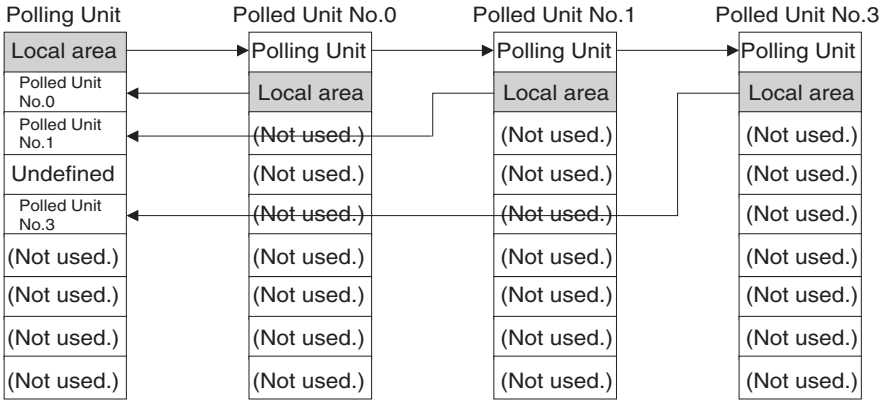
Polling Unit Link Method

The data for all the Polled Units in the Serial PLC Links ar reflected in the Polling Unit only, and each Polled Unit reflects the data of the Polling Unit only. The advantage of the Polling Unit link method is that the address allocated for the local Polled Unit data is the same in each Polled Unit, allowing data to be

accessed using common ladder programming. The areas allocated for the unit numbers of the PT or Polled Units not present in the network are undefined in the Polling Unit only.

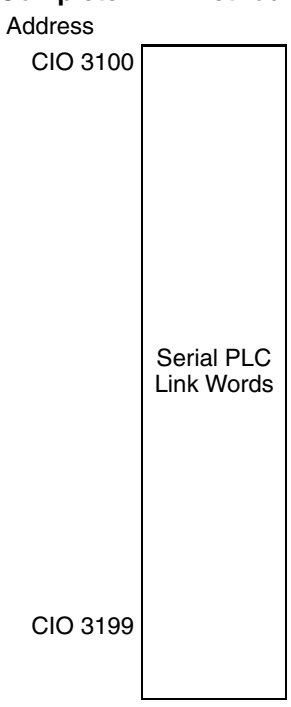
Example: Polling Unit link method, highest unit number: 3.

In the following diagram, Polled Unit No. 2 is a PT or a Unit not participating in the network, so the corresponding area in the Polling Unit is undefined.



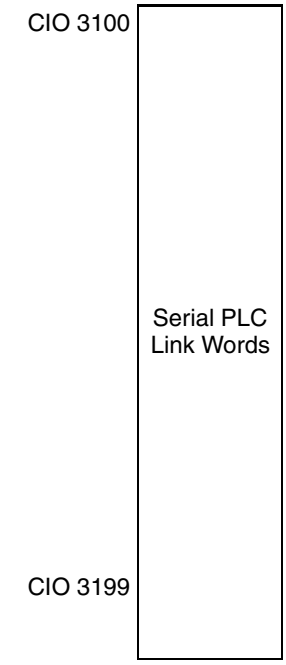
Allocated Words

Complete Link Method



Link words	1 word	2 words	3 words	to	10 words
Polling Unit	CIO 3100	CIO 3100 to CIO 3101	CIO 3100 to CIO 3102		CIO 3100 to CIO 3109
Polled Unit No. 0	CIO 3101	CIO 3102 to CIO 3103	CIO 3103 to CIO 3105		CIO 3110 to CIO 3119
Polled Unit No. 1	CIO 3102	CIO 3104 to CIO 3105	CIO 3106 to CIO 3108		CIO 3120 to CIO 3129
Polled Unit No. 2	CIO 3103	CIO 3106 to CIO 3107	CIO 3109 to CIO 3111		CIO 3130 to CIO 3139
Polled Unit No. 3	CIO 3104	CIO 3108 to CIO 3109	CIO 3112 to CIO 3114		CIO 3140 to CIO 3149
Polled Unit No. 4	CIO 3105	CIO 3110 to CIO 3111	CIO 3115 to CIO 3117		CIO 3150 to CIO 3159
Polled Unit No. 5	CIO 3106	CIO 3112 to CIO 3113	CIO 3118 to CIO 3120		CIO 3160 to CIO 3169
Polled Unit No. 6	CIO 3107	CIO 3114 to CIO 3115	CIO 3121 to CIO 3123		CIO 3170 to CIO 3179
Polled Unit No. 7	CIO 3108	CIO 3116 to CIO 3117	CIO 3124 to CIO 3126		CIO 3180 to CIO 3189
Not used.	CIO 3109 to CIO 3199	CIO 3118 to CIO 3199	CIO 3127 to CIO 3199		CIO 3190 to CIO 3199

Polling Unit Link Method



Link words	1 word	2 words	3 words	to	10 words
Polling Unit	CIO 3100	CIO 3100 to CIO 3101	CIO 3100 to CIO 3102		CIO 3100 to CIO 3109
Polled Unit No. 0	CIO 3101	CIO 3102 to CIO 3103	CIO 3103 to CIO 3105		CIO 3110 to CIO 3119
Polled Unit No. 1	CIO 3101	CIO 3102 to CIO 3103	CIO 3103 to CIO 3105		CIO 3110 to CIO 3119
Polled Unit No. 2	CIO 3101	CIO 3102 to CIO 3103	CIO 3103 to CIO 3105		CIO 3110 to CIO 3119
Polled Unit No. 3	CIO 3101	CIO 3102 to CIO 3103	CIO 3103 to CIO 3105		CIO 3110 to CIO 3119
Polled Unit No. 4	CIO 3101	CIO 3102 to CIO 3103	CIO 3103 to CIO 3105		CIO 3110 to CIO 3119
Polled Unit No. 5	CIO 3101	CIO 3102 to CIO 3103	CIO 3103 to CIO 3105		CIO 3110 to CIO 3119
Polled Unit No. 6	CIO 3101	CIO 3102 to CIO 3103	CIO 3103 to CIO 3105		CIO 3110 to CIO 3119
Polled Unit No. 7	CIO 3101	CIO 3102 to CIO 3103	CIO 3103 to CIO 3105		CIO 3110 to CIO 3119
Not used.	CIO 3102 to CIO 3199	CIO 3104 to CIO 3199	CIO 3106 to CIO 3199		CIO 3120 to CIO 3199

Procedure

The Serial PLC Links operate according to the following settings in the PLC Setup.

Settings at the Polling Unit

1,2,3...

1. Set the serial communications mode of the RS-232C communications port to Serial PLC Links (Polling Unit).
2. Set the link method to the Complete Link Method or Polling Unit Link Method.
3. Set the number of link words (up to 10 words for each Unit).
4. Set the maximum unit number in the Serial PLC Links (0 to 7).

Settings at the Polled Units

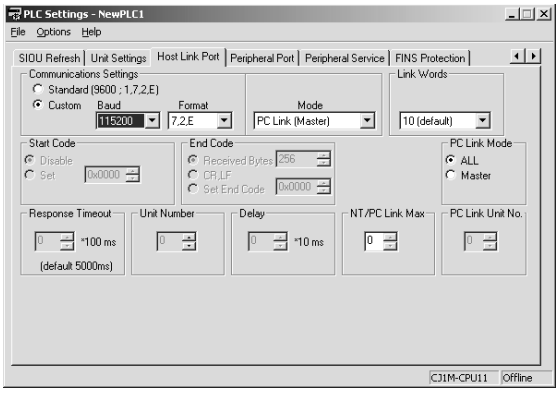
1,2,3...

1. Set the serial communications mode of the RS-232C communications port to Serial PLC Links (Polled Unit).
2. Set the unit number of the Serial PLC Link Polled Unit.

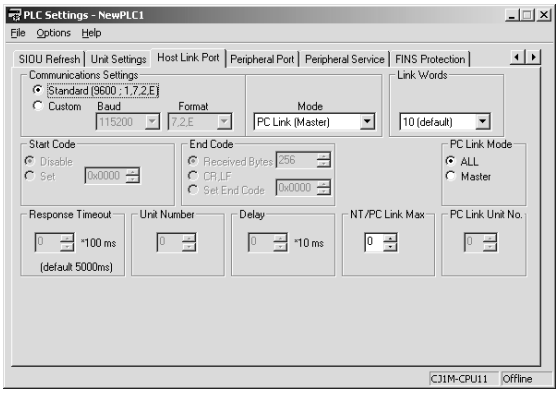
PLC Setup

Settings at the Polling Unit

When using a high-speed serial PLC Link, select the *Custom* Option in the Communications Settings Area and set the baud rate to 115,200 bps.



When using a standard serial PLC Link, select the *Standard* Option in the Communications Settings Area.



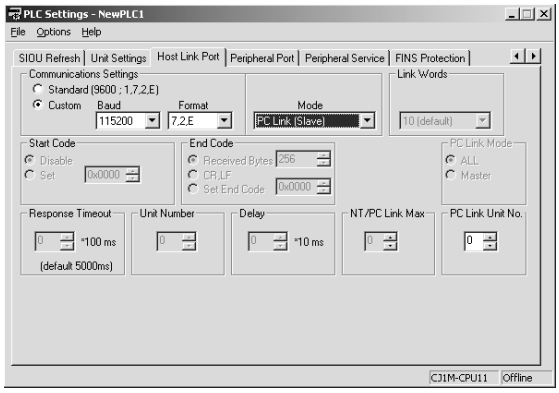
The following table shows the corresponding settings when using a Programming Console.

Item		Programming Console address		Set value	Default	Refresh timing
		Word	Bit			
RS-232C port setting	Serial communications mode	+160	11 to 08	8 hex: Serial PLC Links Polling Unit	0 hex	Every cycle
	Port baud rate	+161	07 to 00	00 to 09 hex: Standard 0A hex: High-speed	00 hex	
	Link method	+166	15	0: Complete links 1: Polling Unit links	0	
	Number of link words		07 to 04	1 to A hex	0 hex (See note.)	
	Highest unit number		03 to 00	0 to 7 hex	0 hex	

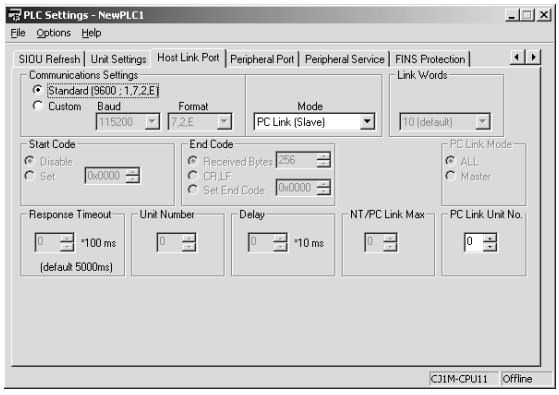
Note Automatically allocates 10 words (A hex) when the default setting of 0 hex is used.

Settings at the Polled Unit

When using a high-speed serial PLC Link, select the *Custom* Option in the Communications Settings Area and set the baud rate to 115,200 bps.



When using a standard serial PLC Link, select the *Standard* Option in the Communications Settings Area.



The following table shows the corresponding settings when using a Programming Console.

Item		Programming Console address		Set value	Default	Refresh timing
		Word	Bit			
RS-232C port settings	Serial communications mode	+160	11 to 08	7 hex: Serial PLC Link Polled Unit	0 hex	Every cycle
	Port baud rate	+161	07 to 00	00 to 09 hex: Standard 0A hex: High-speed	00 hex	
	Polled Unit unit number	+167	03 to 00	0 to 7 hex	0 hex	

Related Auxiliary Area Flags

Name	Address	Details	Read/write	Refresh timing
RS-232C Port Communications Error Flag	A39204	Turns ON when a communications error occurs at the RS-232C port. 1: Error 0: Normal	Read	• Cleared when power is turned ON. • Turns ON when a communications error occurs at the RS-232C port. • Turns OFF when the port is restarted. • Disabled in Peripheral Bus Mode, NT Link Mode, and Serial PLC Link Polling/ Polled Unit Mode.

Name	Address	Details	Read/write	Refresh timing
RS-232C Port Communicating with PT Flag (See note.)	A39300 to A39307	When the RS-232C port is being used in NT link mode, the bit corresponding to the Unit performing communications will be ON. Bits 00 to 07 correspond to unit numbers 0 to 7, respectively. 1: Communicating 0: Not communicating	Read	• Cleared when power is turned ON. • Turns ON the bit corresponding to the unit number of the PT/Polled Unit that is communicating via the RS-232C port in NT Link Mode or Serial PLC Link Mode. • Bits 00 to 07 correspond to unit numbers 0 to 7, respectively.
RS-232C Port Restart Bit	A52600	Turn ON this bit to restart the RS-232C port.	Read/write	• Cleared when power is turned ON. • Turned ON when restarting the RS-232C port, (except when communicating in peripheral bus mode). Note: Depending on the system, the bit may automatically turn OFF when restart processing is completed.
RS-232C Port Error Flag	A52800 to A52807	When an error occurs at the RS-232C port, the corresponding error code is stored. Bit 00: Not used. Bit 01: Not used. Bit 02: Parity error Bit 03: Framing error Bit 04: Overrun error Bit 05: Timeout error Bit 06: Not used. Bit 07: Not used.	Read/write	• Cleared when power is turned ON. • When an error occurs at the RS-232C port, the corresponding error code is stored. • Depending on the system, the flag may be cleared when the RS-232C port is restarted. • Disabled during Peripheral Bus Mode, 1:N NT Link Mode, and Serial PLC Polling/Polled Unit Mode.
RS-232C Port Settings Changed Flag	A61902	Turns ON when the communications conditions of the RS-232C port are being changed. 1: Changed 0: No change	Read/write	• Cleared when power is turned ON. • Turns ON while communications conditions settings for the RS-232C port are being changed. • Turns ON when the CHANGE SERIAL PORT SETUP instruction (STUP(237)) is executed. • Turns OFF again when the changes to settings are completed.

Note In the same way as for the NT Link (1:N), the status (communicating/not communicating) of PTs in the Serial PLC Link can be checked from the Polling Unit (CPU Unit) by reading the RS-232C Port Communicating with PT Flag (A393 bits 00 to 07 for unit numbers 0 to 7).

6-4 Changing the Timer/Counter PV Refresh Mode

6-4-1 Overview

The timer/counter PV refresh mode can be set to either BCD mode or binary mode (see notes).

When binary mode is used, the timer/counter setting time of 0 to 9999 can be expanded to 0 to 65535. Binary data calculated using other instructions can also be used for the timer/counter set values. The timer/counter PV refresh mode can also be specified when the timer/counter set value is specified as an address (indirect specification). (The setting of the mode as BCD mode or

binary mode will determine whether the contents of the addressed word are taken as a BCD or binary value.)

There are differences in the instruction operands for BCD mode and binary mode, however, so check and understand the differences between the BCD and binary modes before changing the timer/counter PV refresh mode.

Note

1. The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units support only the BCD mode.
2. When the mnemonic is monitored from the Programming Console for CS1-H, CJ1-H, CJ1M, or CS1D CPU Units manufactured on or before 31 May 2002 with the timer/counter PV refresh mode set to binary mode, the mnemonic of the binary is displayed as the mnemonic or the BCD instruction (example: TIMX #0000 &16 is displayed as TIM #0000 &16), but operations are performed in binary mode.
3. The PV refresh mode can be selected with CX-Programmer Ver 3.0 only. Mode selection is not supported by CX-Programmer Ver 2.1 or lower, or the Programming Consoles.
4. CX-Programmer Ver. 2.1 or lower cannot read user programs for the CPU Unit containing binary-mode instructions, but it can read those set using BCD-mode instructions.

6-4-2 Functional Specifications

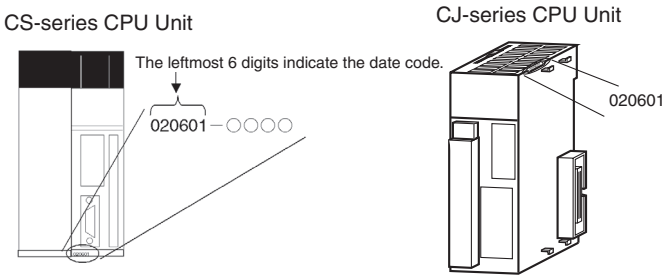
Item	Details		
Timer/counter PV refresh mode setting method	Must be set using CX-Programmer Ver.3.0 (not supported by CX-Programmer Ver 2.1 or lower). Set in the PLC properties of CX-Programmer Ver.3.0.		
Supported CPU Units	CS1-H/CJ1-H CPU Units from Lot No. 020601 (manufactured on 1 June 2002) or later (see note 1), and CJ1M and CS1D CPU Units.		
Mode	**BCD mode**	**Binary mode**	
Mnemonic	Same as previous models Example: TIM	X added to BCD mode mnemonic Example: TIMX	
Function code	Same as previous models	New codes	
PV/SV range	#0000 to #9999	&0 to &65536	#0000 to #FFFF
PV display on Programming Device (CX-Programmer Ver.3.0 or Programming Console)	BCD Example: #0100	Decimal Example: &100	Hexadecimal Example: #64

Note When the mnemonic is monitored from the Programming Console for CS1-H/CJ1-H CPU Units manufactured on or before 31 May 2002 with the timer/counter PV refresh mode set to binary mode, the mnemonic of the binary is displayed as the mnemonic or the BCD instruction (example: TIMX #0000 &16 is displayed as TIM #0000 &16), but operations are performed in binary mode.

Checking the CPU Unit Lot Number

1,2,3...

1. The lot number is printed on the bottom of the front panel (CS Series) or the right corner of the top of the Unit (CJ Series), and is comprised of the last two digits of the year, the month, and the day, in that order, as shown in the following diagram.
   Example: 020601 (Manufactured on 1 June 2002.)

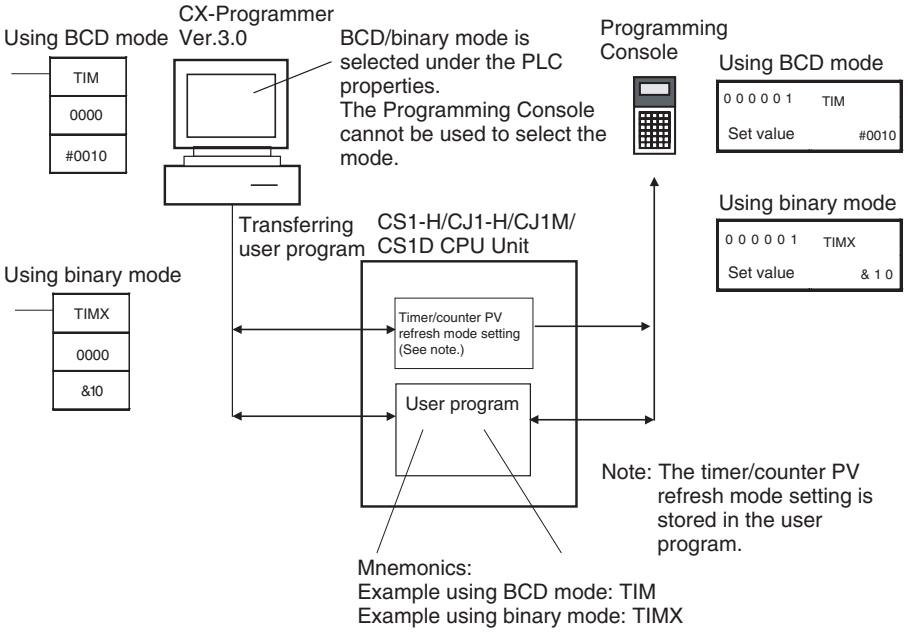


2. Check which mode is selected by putting the CX-Programmer online, opening the I/O Table Window, and selecting ***Unit Information*** - ***CPU Unit***. The lot No. will be displayed in the same format as shown in the above diagram, i.e., comprised of the last two digits of the year, the month, and the day, in that order.

6-4-3 BCD Mode/Binary Mode Selection and Confirmation

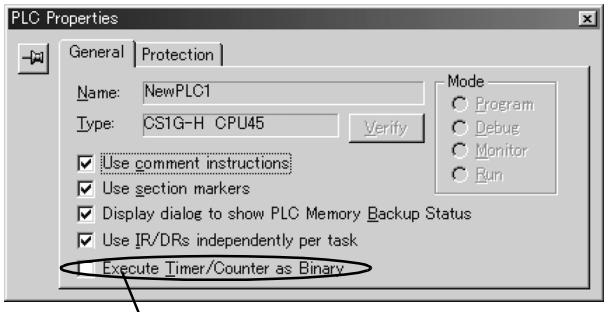
When writing a new program, the BCD mode/binary mode is selected in the PLC property settings in CX-Programmer.

Note The BCD mode/binary mode selection is supported by CX-Programmer Ver 3.0 or higher only. CX-Programmer Ver 2.1 or lower versions do not allow mode selection.



BCD Mode/Binary Mode Selection

1,2,3... 1. Select the PLC name, click the right mouse button, and select ***PLC Properties***.

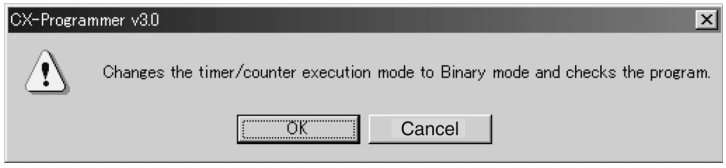


Select this check box to enable the setting.

2. Click the **General** Tab, and select ***Execute Timers/Counters as Binary***.
   - Not selected (default): BCD mode
   - Selected: Binary mode

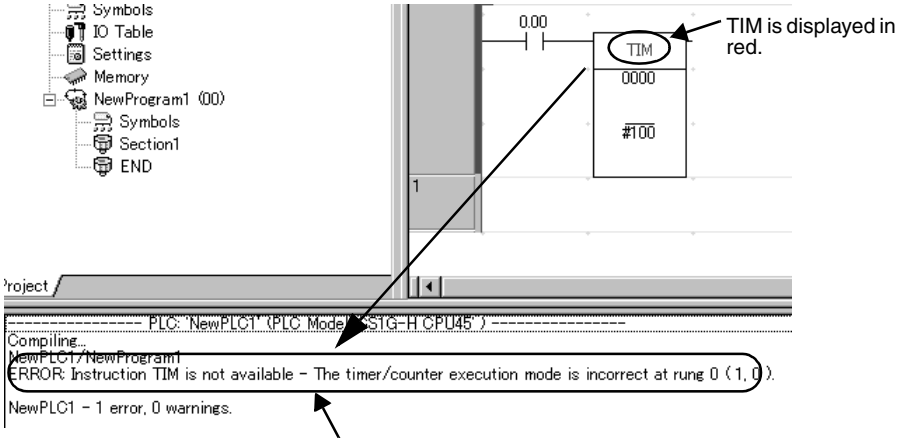
   The timer/counter PV refresh mode set value set under the PLC properties will be stored in the CPU Unit's user memory when the user program is transferred from the CX-Programmer to the CPU Unit.

   When the setting is changed, the following dialog box will be displayed automatically.



Click the **OK** Button to execute the program check. The program check results will be displayed in the output window.

Example: The TIM instruction has been used even though the mode has been changed to binary mode.



The program check results are displayed in the output window.

Example: The timer/counter operation mode is different, so TIM cannot be used.

BCD Mode/ Binary Mode Confirmation

A09915 in the Auxiliary Area (Timer/Counter PV Refresh Mode Flag) can be used to check whether a CPU Unit is operating in BCD mode or binary mode.

Name	Address	Details
Timer/Counter PV Refresh Mode Flag	A09915	0: BCD mode 1: Binary mode

6-4-4 BCD Mode/Binary Mode Mnemonics and Data

BCD Mode/Binary Mode Mnemonics

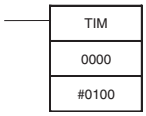
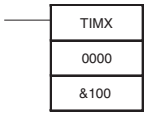
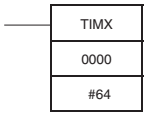
Binary mode mnemonics are indicated by the suffix X added to the BCD mnemonic.

Example: Mnemonics for the HUNDRED-MS TIMER instruction

BCD mode: TIM

Binary mode: TIMX

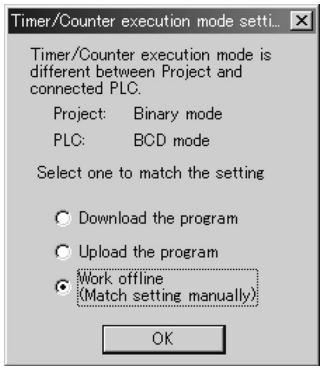
BCD Mode/Binary Mode Data Display

PLC property	Meaning of input and display symbols	Setting range	Example: Timer number: 0000, Set value: 10 s
BCD mode	The # symbol indicates the instruction value (a BCD value when BCD mode is used)	#0000 to #9999 or #00000000 to #99999999	
Binary mode	The & symbol indicates a decimal value.	&0 to &65535 or &0 to &4294967295	
	The # symbol indicates the instruction value (a hexadecimal value when BCD mode is used.)	#0000 to #FFFF or #0000 to #FFFFFFFF	

Note When using the CX-Programmer in either BCD or binary mode, if the numerical value is input without including the input/display symbol # or & indicating the constant, (e.g., TIM 0000 0010), the timer/counter set value will be input as an address (e.g., the value in CIO word 0010 will be used as the set value).

6-4-5 Restrictions

- BCD mode and binary mode cannot be used together in the same CPU Unit.
- When the Programming Console is used to create a new user program, or to clear memory, the timer/counter PV refresh mode is fixed in BCD mode.
- When CX-Programmer Ver. 3.0 is used to place the CPU Unit online, the set value that is stored in the CPU Unit's user memory for the timer/counter PV refresh mode will be automatically used. If the CPU setting is different from the setting for the CX-Programmer project, an error will occur, and the online connection will not be possible. The following message will be displayed.



Select whether to change the CPU Unit setting to that for the CX-Programmer project or change the CX-Programmer project property setting to that for the CPU Unit.

- CX-Programmer Ver. 2.1 or lower cannot read user programs in the CPU Unit that are set using binary mode, but can read those set using BCD mode.
- The differences between the CX-Programmer and Programming Console operations when an incorrect timer/counter PV refresh mode instruction is input are as follows:
  - CX-Programmer:
    An error will occur if an instruction is input for a different mode than that set as the timer/counter PV refresh mode under *PLC properties*.
    Example: When the PLC in the project is set to binary mode, an error will occur if TIM is input as the mnemonic. When BCD mode is set, an error will occur if TIMX is input as the mnemonic.
  - Programming Console:
    When a function code is input for an instruction for a different mode that for the timer/counter PV refresh mode set in the CPU Unit, the mnemonic will automatically be changed to that for the timer/counter PV refresh mode set in the CPU Unit.

6-4-6 Instructions and Operands

Instructions

Instruction type	Name	Mnemonic	
		BCD mode	Binary mode
Timer and Counter Instructions	HUNDRED-MS TIMER	TIM	TIMX
	TEN-MS TIMER	TIMH	TIMHX
	ONE-MS TIMER	TMHH	TMHHX
	TENTH-MS TIMER (CJ1-H-R CPU Units only)	TIMU	TIMUX
	HUNDREDTH-MS TIMER (CJ1-H-R CPU Units only)	TMUH	TMUHX
	ACCUMULATIVE TIMER (100 ms)	TTIM	TTIMX
	LONG TIMER (100 ms)	TIML	TIMLX
	MULTI-OUTPUT TIMER (100 ms)	MTIM	MTIMX
	COUNTER	CNT	CNTX
	REVERSIBLE COUNTER	CNTR	CNTRX
	RESET TIMER/COUNTER	CNR	CNRX
Block program instructions	TIMER WAIT (100 ms)	TIMW	TIMWX
	HIGH-SPEED TIMER WAIT (10 ms)	TMHW	TMHWX
	COUNTER WAIT	CNTW	CNTWX

Instructions and Operands

Timer and Counter Instructions

HUNDRED-MS TIMER

Instruction name	BCD mode	Binary mode
Mnemonic	TIM	TIM**X**
S (timer set value)	#0000 to #9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting time (unit: 0.1 s)	0 to 999.9 s	0 to 6,553.5 s

TEN-MS TIMER

Instruction name	BCD mode	Binary mode
Mnemonic	TIMH	TIMH**X**
S (timer set value)	#0000 to #9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting time (unit: 0.01 s)	0 to 99.99 s	0 to 655.35 s

ONE-MS TIMER

Instruction name	BCD mode	Binary mode
Mnemonic	TMHH	TMHH**X**
S (timer set value)	#0000 to #9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting time (unit: 0.001 s)	0 to 9.999 s	0 to 65.535 s

TENTH-MS TIMER (CJ1-H-R CPU Units Only)

Instruction name	BCD mode	Binary mode
Mnemonic	TIMU	TIMU**X**
S (timer set value)	#0000 to #9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting time (unit: 0.0001 s)	0 to 0.9999 s	0 to 6.5535 s

HUNDREDTH-MS TIMER (CJ1-H-R CPU Units Only)

Instruction name	BCD mode	Binary mode
Mnemonic	TMUH	TMUH**X**
S (timer set value)	#0000 to #9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting time (unit: 0.00001 s)	0 to 0.09999 s	0 to 0.65535 s

ACCUMULATIVE TIMER (100 ms)

Instruction name	BCD mode	Binary mode
Mnemonic	TTIM	TTIM**X**
S (timer set value)	#0000 to #9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting time (unit: 0.1 s)	0 to 999.9 s	0 to 6,553.5 s

LONG TIMER (100 ms)

Instruction name	BCD mode	Binary mode
Mnemonic	TIML	TIML**X**
S, S+1 (timer set values)	#00000000 to #99999999 (BCD)	&0 to &4294967295 (decimal) or #0000 to #FFFFFFFF (hexadecimal)
Setting time (unit: 0.1 s)	0 to 999.9 s	0 to 6,553.5 s

MULTI-OUTPUT TIMER (100 ms)

Instruction name	BCD mode	Binary mode
Mnemonic	MTIM	MTIM**X**
S to S-7 (each timer set value)	#0000 to #9999 (BCD)	&0 to &65535 or #0000 to #FFFF (hexadecimal)
Setting time (unit: 0.1 s)	0 to 999.9 s	0 to 6,553.5 s

COUNTER

Instruction name	BCD mode	Binary mode
Mnemonic	CNT	CNT**X**
S (counter set value)	#0000 to #9999 (BCD)	&0 to& 65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting	0 to 9,999 times	0 to 65,535 times

REVERSIBLE COUNTER

Instruction name	BCD mode	Binary mode
Mnemonic	CNTR	CNTR**X**
S (counter set value)	#0000 to #9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting	0 to 9,999 times	0 to 65,535 times

RESET TIMER/COUNTER

Instruction name	BCD mode	Binary mode
Mnemonic	CNR	CNR**X**

Block Program Instructions

TIMER WAIT (100 ms)

Instruction name	BCD mode	Binary mode
Mnemonic	TIMW	TIMW**X**
S (timer set value)	#0000 or# 9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting time (unit: 0.1 s)	0 to 999.9 s	0 to 6,553.5 s

HIGH-SPEED TIMER WAIT (10 ms)

Instruction name	BCD mode	Binary mode
Mnemonic	TMHW	TMHW**X**
S (timer set value) Unit: 0.01 s	#0000 to #9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting time (unit: 0.01 s)	0 to 999.9 s	0 to 655.35 s

COUNTER WAIT

Instruction name	BCD mode	Binary mode
Mnemonic	CNTW	CNTW**X**
S (counter set value)	#0000 to #9999 (BCD)	&0 to &65535 (decimal) or #0000 to #FFFF (hexadecimal)
Setting	0 to 9,999 times	0 to 65,535 times

6-5 Using a Scheduled Interrupt as a High-precision Timer (CJ1-H-R and CJ1M Only)

When using a CJ1-H-R or CJ1M CPU Unit, the following functions allow a scheduled interrupt to be used as a high-precision timer.

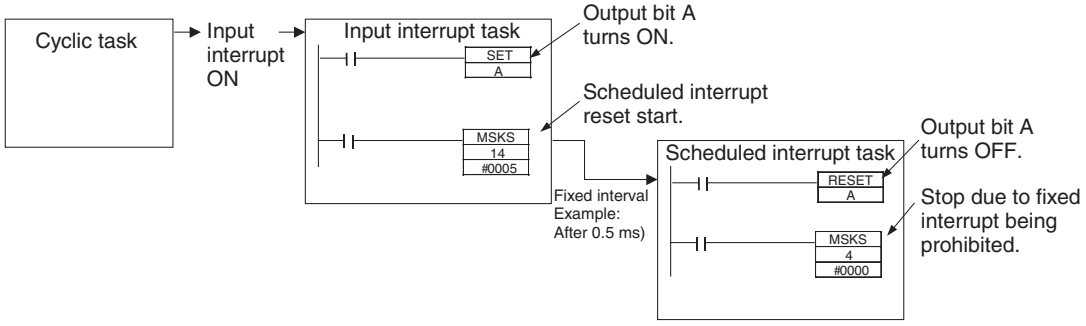
- The scheduled interrupt timer can be input in units of 0.1 ms (high-precision interval timer).
- Resetting (i.e., restart) is possible using the MSKS(690) instruction (fixed time to first interrupt).
- Internal timer PVs can be read using the MSKR(692) instruction (interval timer PV reading)

These functions allow applications such as that shown in the following example of a high-precision one-shot timer, where the input bit turning ON acts as a trigger, causing the output bit to turn ON, and then turn OFF again after a fixed interval.

Example:

1,2,3...

1. Input interrupt task starts when the built-in input bit turns ON.
2. Output bit A turns ON in the input interrupt task, and the MSKS(690) instruction is executed to perform a scheduled interrupt reset start.
3. After a fixed interval, the scheduled interrupt task starts, and output bit A in the scheduled interrupt task turns OFF, and the MSKS(690) instruction is executed to prohibit a scheduled interrupt.



6-5-1 Setting the Scheduled Interrupt to Units of 0.1 ms

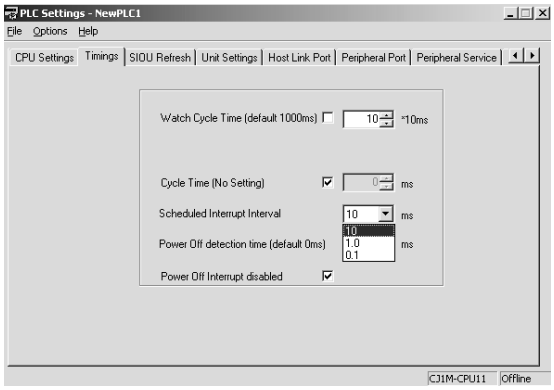
The scheduled interrupt time is set using the PLC Setup's scheduled interrupt unit time setting and the MSKS(690) instruction.

With CJ1-H-R and CJ1M CPU Units, the scheduled interrupt time can be set in units of 0.1 ms.

- CJ1H-R CPU Units: When the units are set to 0.1 ms, the scheduled interrupt time can be set between 0.2 ms and 999.9 ms.
- CJ1M CPU Units: When the units are set to 0.1 ms, the scheduled interrupt time can be set between 0.5 ms and 999.9 ms.

PLC Setup

When using the CX-Programmer, make the settings on the Timings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Item	Programming Console address		Set value	Default	Refresh timing
	Word	Bit			
Scheduled interrupt unit time setting	+195	00 to 03	0 hex: 10-ms unit 1 hex: 1-ms unit 2 hex: 0.1-ms unit (CJ1-H-R and CJ1M CPU Units only)	0 hex	When operation starts.

6-5-2 Specifying a Reset Start with MSKS(690)

When CJ1M CPU Units are used and the MSKS(690) instruction is used to start the scheduled interrupt, the internal timer can be reset before starting the interrupt (this is called a reset start).

This method can be used to specify the time to the first interrupt without using the CLI(691) instruction.

Scheduled interrupts are started by using the MSKS(690) instruction to set the scheduled interrupt time (interval between two interrupts). After executing the MSKS(690) instruction, however, the time required before the first scheduled interrupt task starts (first interrupt start time) is fixed only if the CLI(691) instruction is specified. Therefore, CJ1M CPU Units provide an internal timer reset start, allowing the time to the first interrupt to be set without using the CLI(691) instruction.

MSKS(690) Instruction Operand (Only when Scheduled Interrupt Is Specified)

Operand	Set value
N (Interrupt number)	4: Scheduled interrupt 0, normal setting (internal timer not reset) 5: Scheduled interrupt 1, normal setting (internal timer not reset) 14: Scheduled interrupt 0, specifies reset start (CJ1M CPU Units only) 15: Scheduled interrupt 1, specifies reset start (CJ1M CPU Units only)

6-5-3 Reading the Internal Timer PV with MSKR(692)

CJ1M CPU Units allow reading the PV of the internal timer that measures the scheduled interrupt time. The time is read from either the scheduled interrupt start point or the previous scheduled interrupt point. The internal timer PV is read by executing the MSKR(692) instruction. The unit of time depends on the scheduled interrupt unit time setting in the PLC Setup, in the same way as for the scheduled interrupt time.

MSKR(692) Operands (Only when Scheduled Interrupt Is Specified)

Operand	Set value
N (Interrupt number)	4: Scheduled interrupt 0, reads scheduled interrupt time (set value) 5: Scheduled interrupt 1, reads scheduled interrupt time (set value) 14: Scheduled interrupt 0, reads internal timer PV (CJ1M CPU Units only) 15: Scheduled interrupt 1, reads internal timer PV (CJ1M CPU Units only)

6-6 Startup Settings and Maintenance

This section describes the following functions related to startup and maintenance.

- Hot Start/Hot Stop Functions
- Startup Mode Setting
- Power OFF Detection Delay Setting
- Disabling Power OFF Interrupts
- RUN Output
- Clock
- Program Protection

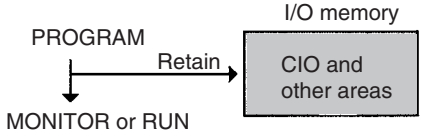
- Remote Programming and Monitoring
- Flash Memory
- Setting Startup Conditions

6-6-1 Hot Start/Hot Stop Functions

Operating Mode Change

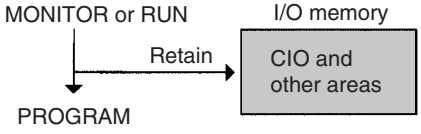
Hot Start

Turn ON the IOM Hold Bit (A50012) to retain all data* in I/O memory when the CPU Unit is switched from PROGRAM mode to RUN/MONITOR mode to start program execution.



Hot Stop

When the IOM Hold Bit (A50012) is ON, all data* in I/O memory will also be retained when the CPU Unit is switched from RUN/MONITOR mode to PROGRAM mode to stop program execution.



Note *The following areas of I/O memory will be cleared during mode changes (PROGRAM ↔ RUN/MONITOR) unless the IOM Hold Bit is ON: the CIO Area (I/O Area, Data Link Area, CPU Bus Unit Area, Special I/O Unit Area, Inner Board Area, SYSMAC BUS Area, I/O Terminal Area, DeviceNet (CompoBus/D) Area, and Internal I/O Areas), Work Area, Timer Completion Flags, and Timer PVs. (The Inner Board, SYSMAC BUS, and I/O Terminal Areas are supported by CS-series CPU Units only.)

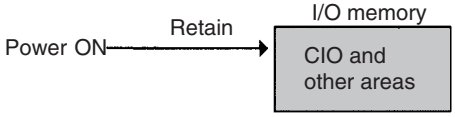
Auxiliary Area Flags and Words

Name	Address	Description
IOM Hold Bit	A50012	When this bit is ON, all of I/O memory will be retained when the operating mode is changed (PROGRAM ↔ RUN/MONITOR).

When the IOM Hold Bit is ON, all outputs from Output Units will be maintained when program execution stops. When the program starts again, outputs will have the same status that they had before the program was stopped.
(When the IOM Hold Bit is OFF, instructions will be executed after the outputs have been cleared.)

PLC Power ON

In order for all data* in I/O memory to be retained when the PLC is turned on (OFF → ON), the IOM Hold Bit must be ON and it must be protected in the PLC Setup (address 80, IOM Hold Bit Status at Startup).

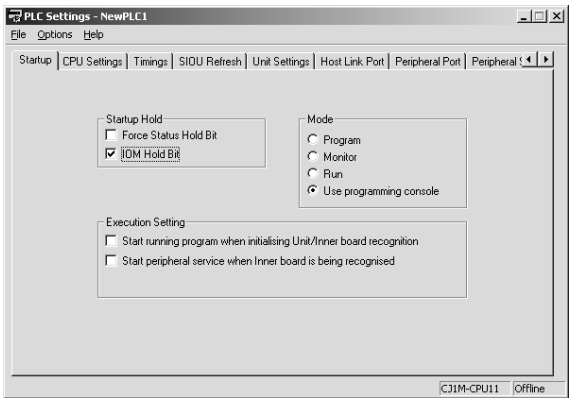


Auxiliary Area Flags and Words

Name	Address	Description
IOM Hold Bit	A50012	When this bit is ON, all of I/O memory will be retained when the operating mode is changed (PROGRAM ↔ RUN/MONITOR).

PLC Setup

When using the CX-Programmer, make the settings on the Startup Tab Page.

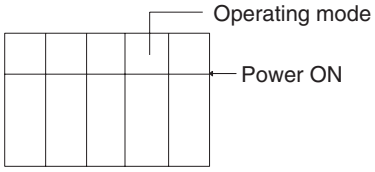


The following table shows the corresponding settings when using a Programming Console.

Programming Console address	Name	Setting	Default
+80, bit 15	IOM Hold Bit Status at Startup	0: The IOM Hold Bit is cleared to 0 when power is turned on. 1: The IOM Hold Bit is retained when power is turned on.	0 (Cleared)

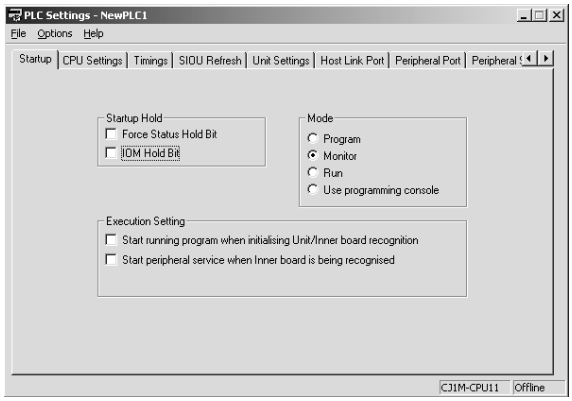
6-6-2 Startup Mode Setting

The CPU Unit's initial operating mode (when the power is turned on) can be set in the PLC Setup.



PLC Setup

When using the CX-Programmer, make the settings on the Startup Tab Page.



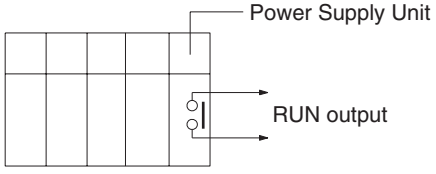
The following table shows the corresponding settings when using a Programming Console.

Programming Console address	Name	Meaning	Setting	Default
+81	Startup Mode	Specifies operating mode to use at startup	PRCN: Programming Console's mode switch PRG: PROGRAM mode MON: MONITOR mode RUN: RUN mode	PRCN: Programming Console's mode switch

Note If the Startup Mode is set to PRCN (Programming Console's mode switch) but a Programming Console isn't connected, the CPU Unit will start in RUN mode. Change the PLC Setup from the default value to start in MONITOR mode or PROGRAM mode when the power is turned ON. (The CS-series CS1 CPU Units, however, will start in PROGRAM mode under the same conditions.)

6-6-3 RUN Output

Some of the Power Supply Units (the C200HW-PA204R, C200HW-PA209R, CJ1W-PA205R, and CS1D-PA207R) are equipped with a RUN output. This output point is ON (closed) when the CPU Unit is operating in RUN or MONITOR mode and OFF (open) when the CPU Unit is in PROGRAM mode.



This RUN output can be used to create an external safety circuits, such as an emergency stop circuit that prevents an Output Unit's external power supply from providing power unless the PLC is on.

Note When a Power Supply Unit without a RUN output is used, an equivalent output can be created by programming the Always ON Flag (A1) as the execution condition for an output point from an Output Unit.

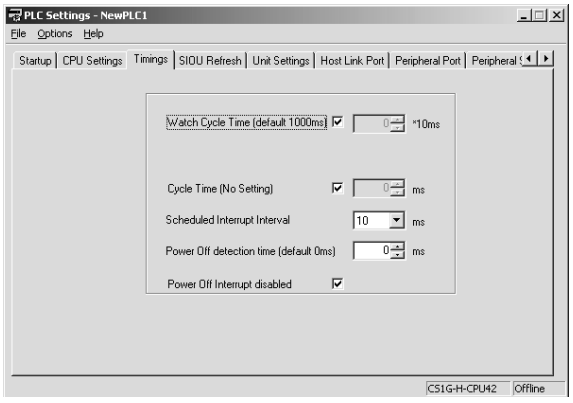
⚠ **Caution** If Output Unit's external power supply goes on before the PLC's power supply, the Output Unit may malfunction momentarily when the PLC first goes on. To prevent any malfunction, add an external circuit that prevents the Output Unit's external power supply from going on before the power supply to the PLC itself. Create a fail-safe circuit like the one described above to ensure that power is supplied by an external power supply only when the PLC is operating in RUN or MONITOR mode.

6-6-4 Power OFF Detection Delay Setting

Normally a power interruption will be detected about 10 to 25 ms (2 to 5 ms for DC power supplies) after the power supply voltage drops below 85% of the minimum rated value(80% for DC power supplies). There is a setting in the PLC Setup (address 225 bits 0 to 7, Power OFF Detection Delay Time) that can extend this time by up to 10 ms (up to 2 ms for DC power supplies).
When the power OFF interrupt task is enabled, it will be executed when the power interruption is confirmed, otherwise the CPU will be reset and operation will be stopped.

PLC Setup

When using the CX-Programmer, make the settings on the Timings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address	Name	Meaning	Setting	Default
+256, bits 00 to 07	Power OFF Detection Delay	Set the time to delay before detecting a power interruption.	00 to 0A (Hex): 0 to 10 ms	00 (Hex): 0 ms

6-6-5 Disabling Power OFF Interrupts

Areas of the program can be protected from power OFF interrupts so that they will be executed before the CPU Unit even if the power supply is interrupted. This is achieved by using the DISABLE
INTERRUPTS (DI(693)) and ENABLE INTERRUPTS (EI(694)) instructions.

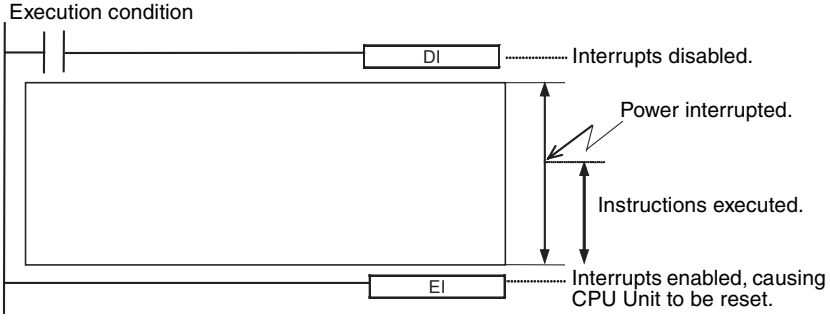
This function can be used with sets of instructions that must be executed as a group, e.g., so that execution does not start with intermediate stored data the next time power is turned ON.

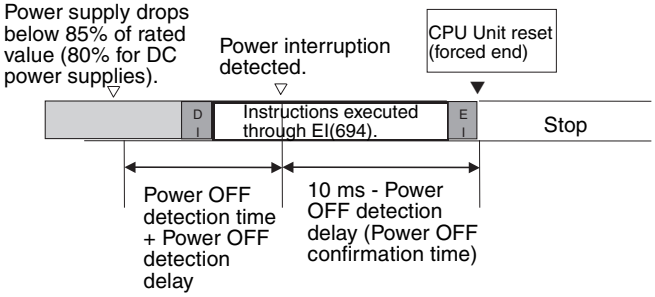
Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support this function.

Procedure

1,2,3...

1. Set the Disable Setting for Power OFF Interrupts in A530 to A5A5 Hex to enable disabling Power OFF Interrupts.
2. Enable disabling Power OFF Interrupts in the PLC Setup (this is the default setting).
3. Use DI(693) to disable interrupts before the program section to be protected and then use EI(694) to enable interrupts after the section. All instructions between DI(693) and EI(694) will be completed before the Power OFF Interrupt is executed even if the power interruption occurs while executing the instructions between DI(693) and EI(694).





Related Settings

Name	Address	Meaning
Disable Setting for Power OFF Interrupts	A530	Enables using DI(693) to disable power OFF interrupt processing (except for execution of the Power OFF Interrupt Task) until EI(694) is executed. A5A5 Hex: Enables using DI(693) to disable power OFF interrupt processing Any other value: Disables using DI(693) to disable power OFF interrupt processing

6-6-6 Clock Functions

The CS/CJ-series PLCs have the following clock functions:

- Monitoring of the time that power interruptions occurred
- Monitoring of the time that the PLC was turned on
- Monitoring of the total time that the PLC has been on

Note The CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units are shipped without the backup battery installed, and the CPU Unit's internal clock will be read

00/01/01 00:00:00 or possibly another value when the battery is connected. To use the clock functions, connect the battery, turn the power ON, and set the time and date with a Programming Device (Programming Console or CX-Programmer) or the FINS command (07 02, CLOCK WRITE). The CPU Unit's internal clock will begin operating once it has been set.

Auxiliary Area Flags and Words

Name	Addresses	Function
Clock data	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 the month: 00 to 31 (BCD)
	A35300 to A35307	Month: 00 to 12 (BCD)
	A35308 to A35315	Year: 00 to 99 (BCD)
	A35400 to A35407	Day of the week: 00: Sunday, 01: Monday, 02: Tuesday, 03: Wednesday, 04: Thursday, 05: Friday, 06: Saturday
Start-up Time	A510 and A511	Contain the time at which the power was turned on (day, hour, minutes, and seconds).
Power Interruption Time	A512 and A513	Contain the time at which the power was last interrupted.
Total Power ON Time	A523	Contains the total time (in binary) that the PLC has been on in 10-hour units.
Power ON Clock Data	A720 to A749 (See note 1.)	These words contain the date and time (year, month, day, hour, and seconds) when the PLC power was turned ON for the last 10 times that the power was turned ON. A720 to A722: Power ON Clock Data 1 (most recent clock data, see note 2.) A723 to A725: Power ON Clock Data 2 A726 to A728: Power ON Clock Data 3 A729 to A731: Power ON Clock Data 4 A732 to A734: Power ON Clock Data 5 A735 to A737: Power ON Clock Data 6 A738 to A740: Power ON Clock Data 7 A741 to A743: Power ON Clock Data 8 A744 to A746: Power ON Clock Data 9 A747 to A749: Power ON Clock Data 10 (old est clock data)

Note
1. Supported only in CPU Units with unit version 3.0 or later.
2. The day, hour, minute, and seconds data in Power ON Clock Data 1 (A720 to A722) is the same as the time data in the Startup Time words (A510 to A511).

Related Instructions

Instruction	Name	Function
SEC(065)	HOURS TO SECONDS	Converts time data in hours/minutes/seconds format to an equivalent time in seconds only.
HMS(066)	SECONDS TO HOURS	Converts seconds data to an equivalent time in hours/minutes/seconds format.
CADD(730)	CALENDAR ADD	Adds time to the calendar data in the specified words.
CSUB(731)	CALENDAR SUBTRACT	Subtracts time from the calendar data in the specified words.
DATE(735)	CLOCK ADJUSTMENT	Changes the internal clock setting to the setting in the specified source words.

6-6-7 Program Protection

The CS/CJ-series user program can be write-protected and completely protected (read/write protection).

Write-protection Using the DIP Switch

The user program can be write-protected by turning ON pin 1 of the CPU Unit's DIP switch. When this pin is ON, it won't be possible to change the user program from a Programming Device (including Programming Consoles). This function can prevent the program from being overwritten inadvertently at the work site.

It is still possible to read and display the program when it is write-protected.

Note For CS/CJ-series CPU Units Ver. 2.0 or later, the above DIP switch setting can be used or, if CX-Programmer version 4.0 or higher is used, program read/write-protection can set as an option when a password is set for the entire program or for one or more tasks. For details, refer to *Enabling/Disabling Creating File Memory Program Files* under *1-4-2 Improved Read Protection Using Passwords* in the *CS Series PLC Operation Manual* or the *CJ Series PLC Operation Manual*.

Read/Write-protection Using Passwords

Both read and write access to the user program area can be blocked from the CX-Programmer. Either the entire area or individual tasks can be protected. Protecting the program will prevent unauthorized copying of the program and loss of intellectual property. A password is set for program protection from a Programming Device (e.g., CX-Programmer) and access is prevented to the whole program.

When using CX-Programmer version 6.1 or higher, you can prohibit the display or editing of selected function block (FB) definitions.

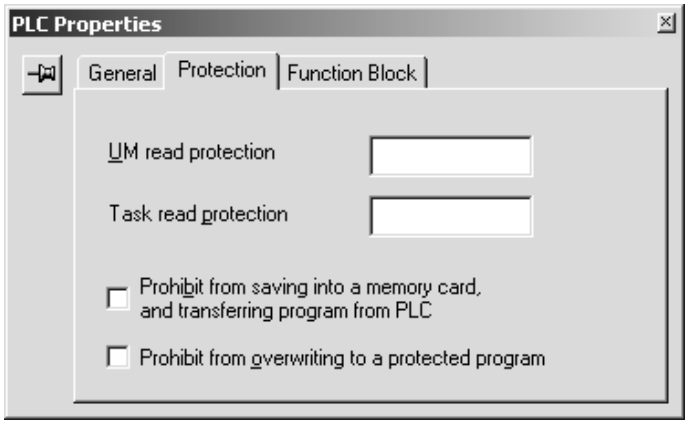
Note

1. If you forget the password, the program within the PLC cannot be transferred to the computer. Make a note of the password, and store it in a safe place.
2. If you forget the password, programs cannot be transferred from the computer to the PLC. Programs can be transferred from the computer to the PLC even if the password protection has not been released.

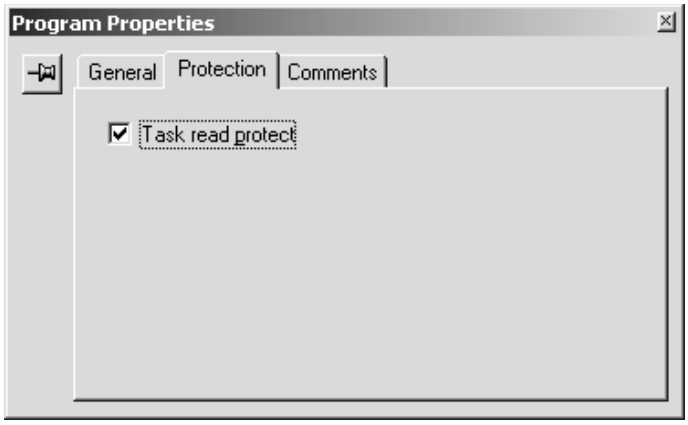
Password Protection

1,2,3...

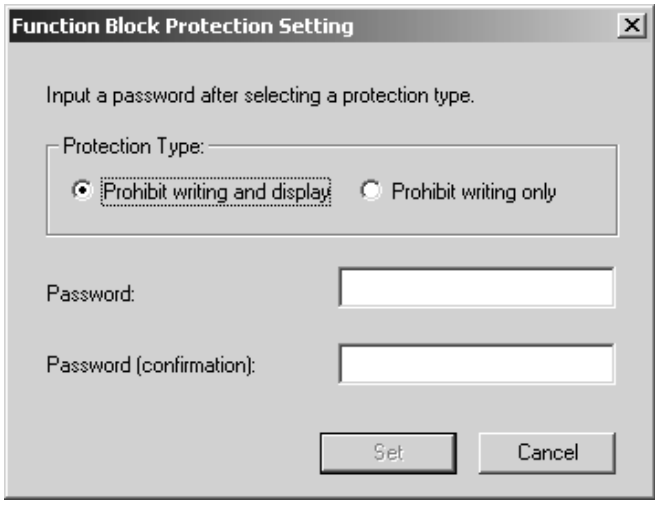
1. Register a password either online or offline as follows:
   a) Select the PLC and select ***Properties*** from the View Menu.
   b) Select ***Protection*** from the PLC Properties Dialog Box and input the password.



To use task read protection, open the Program Properties Dialog Box and select the *Task read protect* Option.



To protect a function block definition, open the Function Block Protection Settings Dialog Box and input the password.



c) Transfer the program.

Note For CS/CJ-series CPU Units Ver. 2.0 or later, read-protection can be set not only for the entire program, but also for specific tasks. For details, refer to *Read Protection for Individual Tasks Using Passwords* under *1-4-2 Improved Read Protection Using Passwords* in the *CS Series PLC Operation Manual* or the *CJ Series PLC Operation Manual*.

Confirming the User Program Date

The dates the program and parameters were created can be confirmed by checking the contents of A090 to A097.

Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support this function.

Auxiliary Area Words

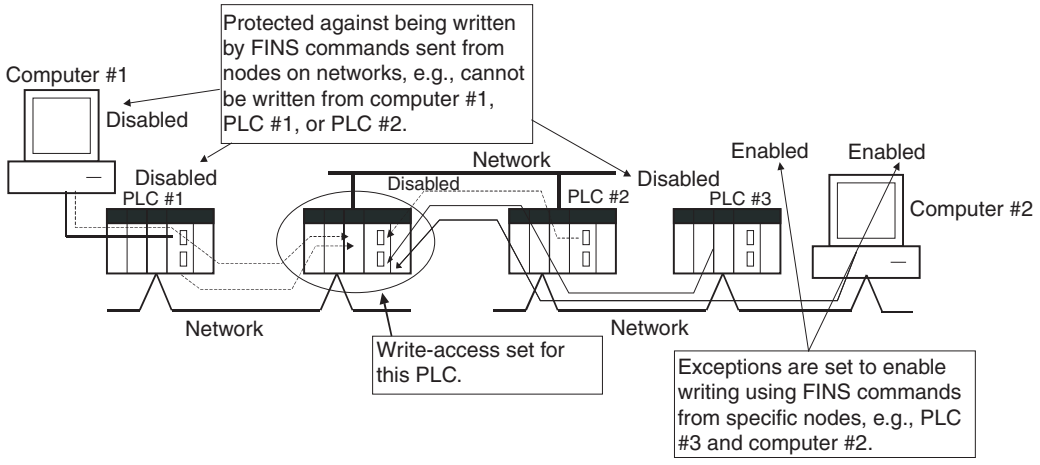
Name	Address	Description	
User Program Date	A090 to A093	The time and date the user program was last overwritten in memory is given in BCD.	
		A09000 to A09007	Seconds (00 to 59 BCD)
		A09008 to A09015	Minutes (00 to 59 BCD)
		A09100 to A09107	Hour (00 to 23 BCD)
		A09108 to A09115	Day of month (01 to 31 BCD)
		A09200 to A09207	Month (01 to 12 BCD)
		A09208 to A09215	Year (00 to 99 BCD)
		A09300 to A09307	Day (00 to 06 BCD) Day of the week: 00: Sunday, 01: Monday, 02: Tuesday, 03: Wednesday, 04: Thursday, 05: Friday, 06: Saturday
Parameter Date	A094 to A097	The time and date the parameters were last overwritten in memory is given in BCD. The format is the same as that for the User Program Date given above.	

6-6-8 Write-protection from FINS Commands Sent to CPU Units via Networks

For CS/CJ-series CPU Units Ver. 2.0 or later, protection can be set to prevent writing to and otherwise controlling CPU Units by using FINS commands via networks (i.e., connections other than direct serial connections). This includes writing from applications using FinsGateway as well as from the CX-Programmer, CX-Protocol, and CX-Process. Reading will still be possible in this case.

The following operations will be prohibited if write-protection is set: Downloading the user program, PLC Setup, I/O memory, or other data; changing the operating mode; online editing; and any other write or control operations.

Even if write-protection is set, specific nodes can be set as exception to enable executing write/control operations from them.



Note This function prevents writing/control operations only for FINS commands. It does not affect writing and control operations performed by any other method, such as for data links.

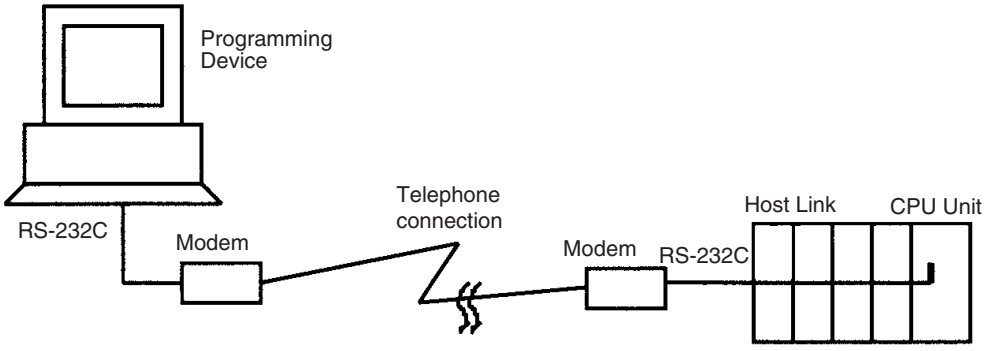
For details, refer to *1-4-3 Write-protection from FINS Commands Sent to CPU Units via Networks* in the *CS Series PLC Operation Manual* or the *CJ Series PLC Operation Manual.*

6-6-9 Remote Programming and Monitoring

CS/CJ-series PLCs can be programmed and monitored remotely through a modem or Controller Link network.

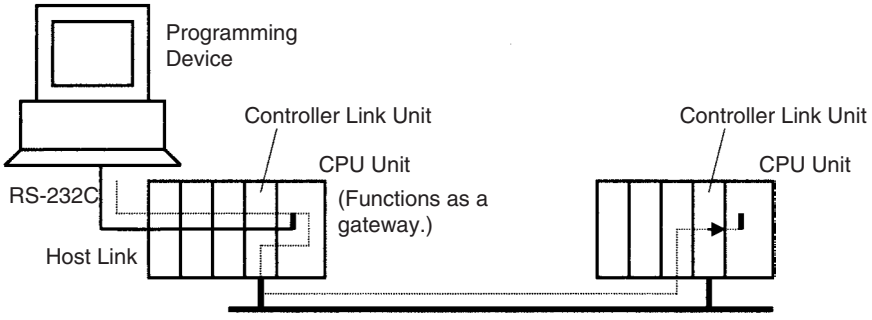
1,2,3... 1. Modem Connections

The host link function can operate through a modem, which allows monitoring of a distant PLC's operation, data transfers, or even online editing of a distant PLC's program by phone. All of the Programming Device's online operations are supported in these connections.



2. Controller Link Network Connections

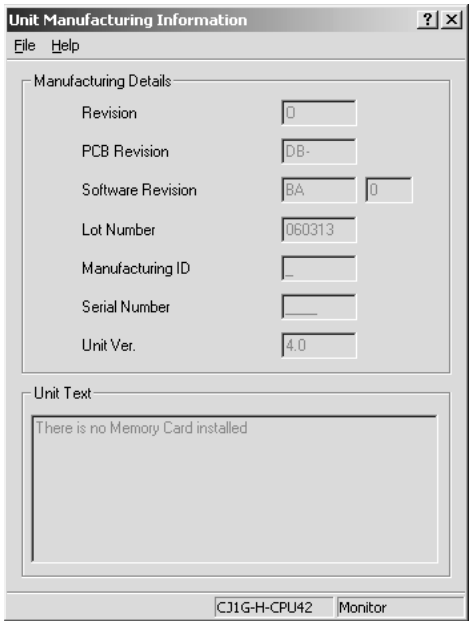
PLCs in a Controller Link or Ethernet network can be programmed and monitored through the Host Link. All of the Programming Device's online operations are supported in these connections.



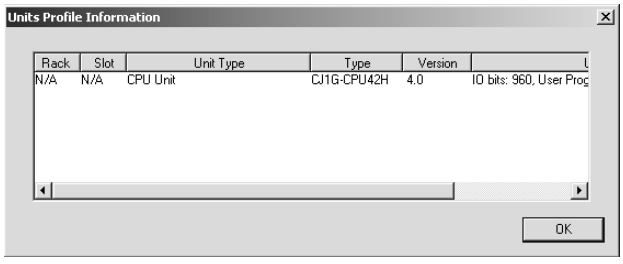
6-6-10 Unit Profiles

The following information can be read for CS/CJ-series Units from the CX-Programer.

- Manufacturing information (lot number, serial number, etc.): Facilitates providing information to OMRON when problems occur with Units.



- Unit information (type, model number, correct rack/slot position): Provides an easy way to obtain mounting information.

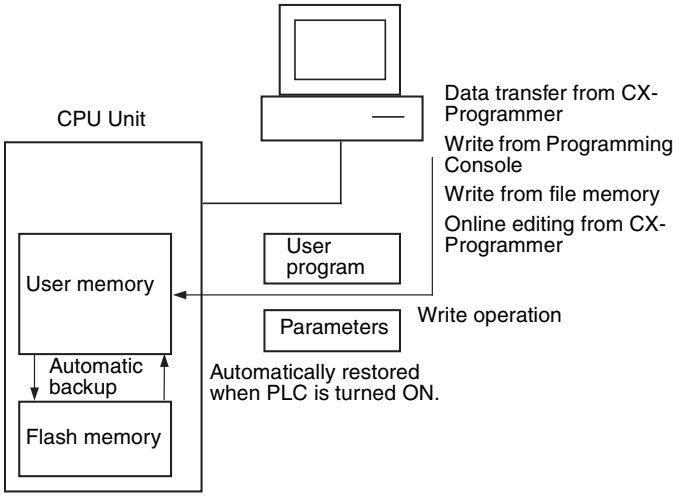


- User-defined text (256 characters max.): Enables recording information necessary for maintenance (Unit inspection history, manufacturing line numbers, and other application information) in Memory Cards.

6-6-11 Flash Memory

The user program and parameters are automatically backed up in flash memory whenever they are written to or altered in the CPU Unit.

- The following data is backed up automatically: User program, parameters (including the PLC Setup, registered I/O tables, routing tables, and CPU Bus Unit data, such as the data link tables).
- The data is backed up automatically whenever the user program or parameters are written in the CPU Unit, including for data transfer operations from the CX-Programmer, writing data from a Programming Console, online editing, data transfers from a Memory Card or EM file memory, etc.
- The user program and parameter data written to flash memory is automatically transferred to user memory in the CPU Unit at startup.



Note

1. The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support this function.
2. The BKUP indicator on the front of the CPU Unit will light while data is being written to flash memory. Do not turn OFF the power supply to the CPU Unit until the backup operation has been completed (i.e., until the BKUP indicator goes out) after transferring data from the a Programming Device or file memory, or performing online editing. The data may be lost.

   Only for online editing and only when there is a Battery in the CPU Unit, the CPU Unit will restart in the previous condition (e.g., with the BKUP indicator lit) even if the power supply is turned OFF after the backup operation has been completed. Note the following precautions:

   a) More time will be required to start the CPU Unit.

   b) Even if there is a Battery in the CPU Unit, always be sure that the backup operation has been completed before turning OFF the power supply if the CPU Unit will be left unpowered for an extended period of time.

The amount of time required to back up data (the time the BKUP indicator will be lit) will depend on the size of the user program, as shown in the following table.

User program size	Backup processing time		
	MONITOR mode		PROGRAM mode
	Cycle time of 0.4 ms (example)	Cycle time of 10.0 ms (example)	
10 Ksteps	2 s	8 s	1 s
60 Ksteps	11 s	42 s	6 s
250 Ksteps	42 s	170 s	22 s

Note

1. The BKUP indicator will be lit when power is supplied to the CPU Unit.
2. Depending on the type of online editing that was performed, up to 1 minute may be required to backup data.

⚠**Caution** Automatically back up the user program and parameter data to flash memory when they 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.

Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support the flash memory functions.

Note A backup status will be displayed in a Memory Backup Status Window by the CX-Programmer when backing up data from the CX-Programmer for transfer operations other than normal data transfers (***PLC/Transfer***). To obtain this window, setting to display the backup status dialog box must be checked in the PLC properties and the window must be selected from the View Menu. For normal transfer operations, the backup status will be displayed in the transfer window after the transfer status for the program and other data.

Auxiliary Area Flags

Name	Address	Meaning
Flash Memory Error Flag	A40310	Turns ON when the flash memory fails.

6-6-12 Startup Condition Settings

Some Units and Inner Boards require extensive time to start up after the power supply is turned ON, affecting the startup time of the CPU Unit. The PLC Setup can be set so that the CPU Unit will start without waiting for these Units to be initialized.

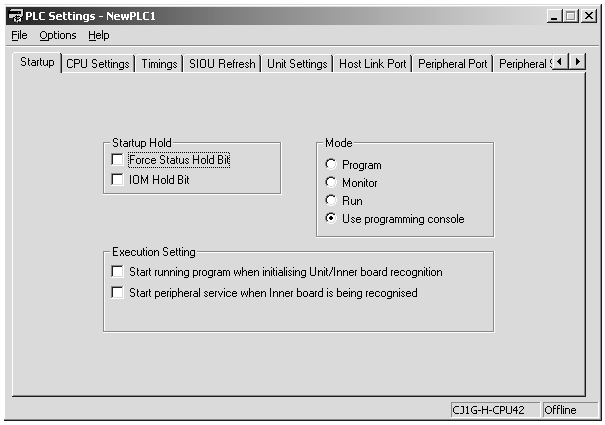
Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support the flash memory functions.

This setting applies to the ITNC-EIS01-CST and ITNC-EIX01-CST Open Network Controller-CS1 Bus Interface Units. (There are currently no Inner Boards that are applicable as of October 2001.)

This function is controller by setting the Startup Condition and Inner Board Setting described in the following table.

PLC Setup

When using the CX-Programmer, make the settings on the Startup Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Startup conditions	PLC Setup	
	Startup Condition (Programming Console address 83, bit 15)	**Inner Board Setting (Programming Console address 84, bit 15)**
To start without waiting for all Units and Boards	1: Enable operation without waiting.	1: Do not wait for specific Inner Boards.
To start without waiting for all Units (wait for Boards)	1: Enable operation without waiting.	0: Wait for all Boards before starting.
To wait for all Units and Boards before starting	0: Always wait for all Units/ Boards	Any

Note With CS1G/H-CPU□□(-V1) and CJ1□CPU□□ CPU Units, the CPU Unit will not start until all Units and Boards have completed startup processing.

Startup Condition

0: If there is one or more of the specific Boards or Units that has not completed startup processing, the CPU Unit will go on standby in MONITOR or PROGRAM mode and wait for all Units and Boards.

1: Even if there is one or more of the specific Boards or Units that has not completed startup processing, the CPU Unit will go ahead and start in MONITOR or PROGRAM mode. The operation for Inner boards, however, also depends on the following setting.

Inner Board Setting

This setting is used only if the Startup Condition is set to 1 to enable starting without waiting for specific Units and Boards. This setting is ignored if the Startup Condition is set to 0.

0: If there is one or more of the specific Boards that has not completed startup processing, the CPU Unit will go on standby in MONITOR or PROGRAM mode and wait for all Boards.

1: Even if there is one or more of the specific Boards that has not completed startup processing, the CPU Unit will go ahead and start in MONITOR or PROGRAM mode.

6-7 Diagnostic Functions

This section provides a brief overview of the following diagnostic and debugging functions.

- Error Log
- Output OFF Function
- Failure Alarm Functions (FAL(006) and FALS(007))
- Failure Point Detection (FPD(269)) Function

6-7-1 Error Log

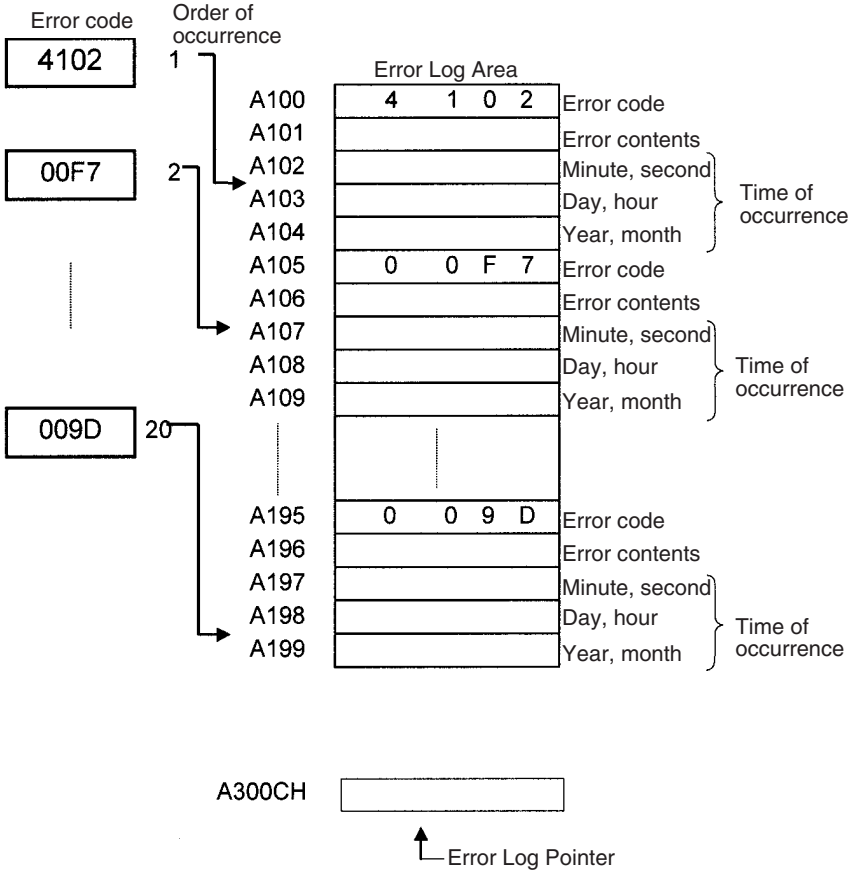
Each time that an error occurs in a CS/CJ-series PLC, the CPU Unit stores error information in the Error Log Area. The error information includes the error code (stored in A400), error contents, and time that the error occurred. Up to 20 records can be stored in the Error Log.

In addition to system-generated errors, the PLC records user-defined FAL(006) and FALS(007) errors, making it easier to track the operating status of the system.

Refer to the section on troubleshooting in the *CS/CJ Series Operation Manual* for details.

Note A user-defined error is generated when FAL(006) or FALS(007) is executed in the program. The execution conditions of these instructions constitute the user-defined error conditions. FAL(006) generates a non-fatal error and FALS(007) generates a fatal error that stops program execution.

When more than 20 errors occur, the oldest error data (in A100 to A104) is deleted, the remaining 19 records are shifted down by one record, and the newest record is stored in A195 to A199.



The number of records is stored in binary in the Error Log Pointer (A300). The pointer is not incremented when more than 20 errors have occurred.

6-7-2 Output OFF Function

As an emergency measure when an error occurs, all outputs from Output Units can be turned OFF by turning ON the Output OFF Bit (A50015). The operating mode will remain in RUN or MONITOR mode, but all outputs will be turned OFF.

Note Normally (when IOM Hold Bit = OFF), all outputs from Output Units are turned OFF when the operating mode is changed from RUN/MONITOR mode to PROGRAM mode. The Output OFF Bit can be used to turn OFF all outputs without switching to PROGRAM mode and stopping program execution.

Application Precaution for DeviceNet

When the master function is used with the CS1W-DRM21 or CJ1W-DRM21, all slave outputs will be turned OFF. When the slave function is used, all inputs to the master will be OFF. When the C200HW-DRM21-V1 is used, however, slave outputs will not be turned OFF.

6-7-3 Failure Alarm Functions

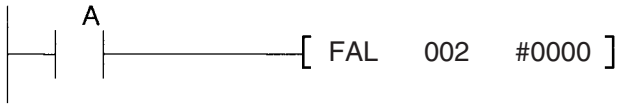
The FAL(006) and FALS(007) instructions generate user-defined errors. FAL(006) generates a non-fatal error and FALS(007) generates a fatal error that stops program execution.

When the user-defined error conditions (execution conditions for FAL(006) or FAL(007)) are met, the Failure Alarm instruction will be executed and the following processing will be performed.

1,2,3...

1. The FAL Error Flag (A40215) or FALS Error Flag (A40106) is turned ON.
2. The corresponding error code is written to A400.
3. The error code and time of occurrence are stored in the Error Log.
4. The error indicator on the front of the CPU Unit will flash or light.
5. If FAL(006) has been executed, the CPU Unit will continue operating.
   If FALS(007) has been executed, the CPU Unit will stop operating. (Program execution will stop.)

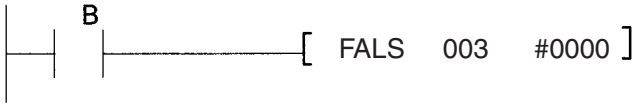
Operation of FAL(006)



When execution condition A goes ON, an error with FAL number 2 is generated, A40215 (FAL Error Flag) is turned ON, and A36002 (FAL Number 2 Flag) is turned ON. Program execution continues.

Errors generated by FAL(006) can be cleared by executing FAL(006) with FAL number 00 or performing the error read/clear operation from a Programming Device (including a Programming Console).

Operation of FALS(007)



When execution condition B goes ON, an error with FALS number 3 is generated, and A40106 (FALS Error Flag) is turned ON. Program execution is stopped.

Errors generated by FAL(006) can be cleared by eliminating the cause of the error and performing the error read/clear operation from a Programming Device (including a Programming Console).

6-7-4 Failure Point Detection

FPD(269) performs time monitoring and logic diagnosis. The time monitoring function generates a non-fatal error if the diagnostic output isn't turned ON within the specified monitoring time. The logic diagnosis function indicates which input is preventing the diagnostic output from being turned ON.

Time Monitoring Function

FPD(269) starts timing when it is executed and turns ON the Carry Flag if the diagnostic output isn't turned ON within the specified monitoring time. The

Carry Flag can be programmed as the execution condition for an error processing block. Also, FPD(269) can be programmed to generate a non-fatal FAL error with the desired FAL number.

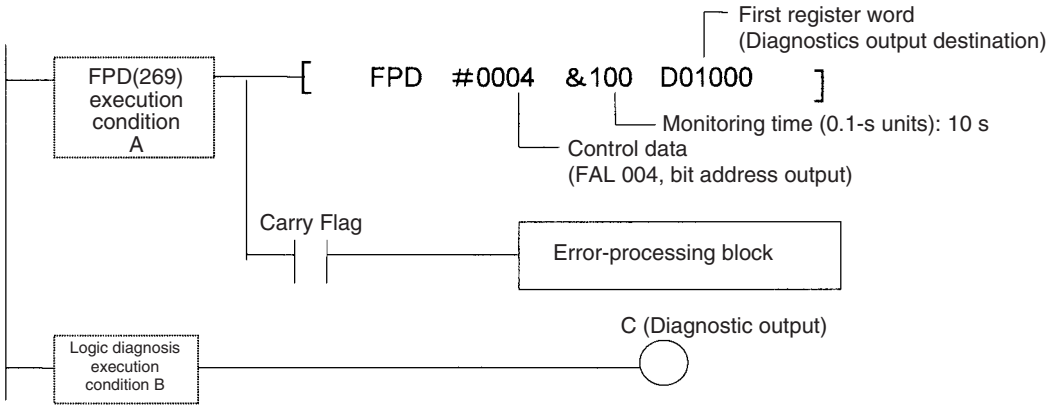
When an FAL error is generated, a preset message will be registered and can be displayed on a Programming Device. FPD(269) can be set to output the results of logic diagnosis (the address of the bit preventing the diagnostic output from being turned ON) just before the message.

The teaching function can be used to automatically determine the actual time required for the diagnostic output to go ON and set the monitoring time.

Logic Diagnosis Function

FPD(269) determines which input bit is causing the diagnostic output to remain OFF and outputs that bit's address. The output can be set to bit address output (PLC memory address) or message output (ASCII).

- If bit address output is selected, the PLC memory address of the bit can be transferred to an Index Register and the Index Register can be indirectly addressed in later processing.
- If the message output is selected, the bit address will be registered in an ASCII message that can be displayed on a Programming Device.



Time Monitoring:

Monitors whether output C goes ON with 10 seconds after input A. If C doesn't go ON within 10 seconds, a failure is detected and the Carry Flag is turned ON. The Carry Flag executes the error-processing block. Also, an FAL error (non-fatal error) with FAL number 004 is generated.

Logic Diagnosis:

FPD(269) determines which input bit in block B is preventing output C from going ON. That bit address is output to D01000 and D01001.

Auxiliary Area Flags and Words

Name	Address	Operation
Error Code	A400	When an error occurs, its error code is stored in A400.
FAL Error Flag	A40215	ON when FAL(006) is executed.
FALS Error Flag	A40106	ON when FALS(007) is executed.
Executed FAL Number Flags	A360 to A391	The corresponding flag is turned ON when an FAL(006) or FALS(007) error occurs.
Error Log Area	A100 to A199	The Error Log Area contains information on the most recent 20 errors.

Name	Address	Operation
Error Log Pointer	A300	When an error occurs, the Error Log Pointer is incremented by 1 to indicate where the next error record will be recorded as an offset from the beginning of the Error Log Area (A100).
Error Log Pointer Reset Bit	A50014	Turn this bit ON to reset the Error Log Pointer (A300) to 00.
FPD Teaching Bit	A59800	Turn this bit ON when you want the monitoring time to be set automatically when FPD(269) is executed.

6-7-5 Simulating System Errors

FAL(006) and FALS(007) can be used to intentionally create fatal and non-fatal system errors. This can be used in system debugging to test display messages on Programmable Terminals (PTs) or other operator interfaces.

Use the following procedure.

Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support this function.

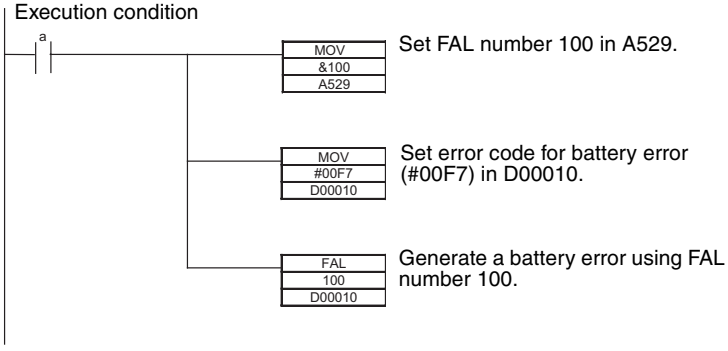
1,2,3... 1. Set the FAL or FALS number to use for simulation in A529. (A529 is used when simulating errors for FAL(006) and FALS(007).
2. Set the FAL or FALS number to use for simulation as the first operand of FAL(006) or FALS(007).
3. Set the error code and error to be simulated as the second operation (two words) of FAL(006) or FALS(007). Indicate a nonfatal error for FAL(006) and a fatal error for FALS(007).

To simulate more than one system error, use more than one FAL(006) or FALS(007) instruction as described above.

Auxiliary Area Flags and Words

Name	Address	Operation
FAL/FALS Number for System Error Simulation	A529	Set a dummy FAL/FALS number to use to simulate the system error. 0001 to 01FF Hex: FAL/FALS numbers 1 to 511 0000 or 0200 to FFFF Hex: No FAL/FALS number for system error simulation.

Example for a Battery Error



Note Use the same methods as for actual system errors to clear the simulated system errors. Refer to the *CS-series Operation Manual* or the *CJ-series Operation Manual* for details. All system errors simulated with FAL(006) and FALS(007) can be cleared by cycling the power supply.

6-7-6 Disabling Error Log Storage of User-defined FAL Errors

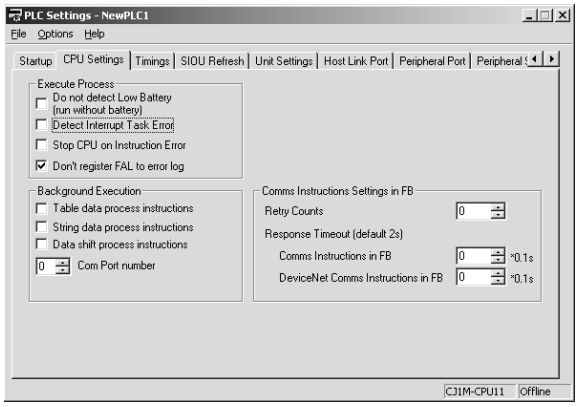
The PLC Setup provides a setting that will prevent user-defined FAL errors created with FAL(006) and time monitoring for FPD(269) from being recorded in the error log (A100 to A199). The FAL error will still be generated even if this setting is used and the following information will also be output: A40215 (FAL Error Flag), A360 to A391 (Executed FAL Numbers), and A400 (Error Code.

This function can be used when only system FAL errors need to be stored in the error log, e.g., when there are many user-defined errors generated by the program using FAL(006) and these fill up the error log too quickly.

Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support this function.

PLC Setup

When using the CX-Programmer, make the settings on the CPU Settings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address		Name	Setting	Default	CPU Unit refresh timing
Word	Bit				
+129	15	User FAL Storage Setting	0: Record user-defined FAL errors in error log. 1: Don't record user-defined FAL errors in error log.	0: Record	Whenever FAL(006) is executed (every cycle)

Note The following items will be stored in the error log even if the above setting is used to prevent user-defined FAL errors from being recorded.

- User-defined fatal errors (FALS(007))
- Non-fatal system errors
- Fatal system errors
- User-simulated nonfatal system errors (FAL(006))
- User-simulated fatal system errors (FALS(007))

6-8 CPU Processing Modes

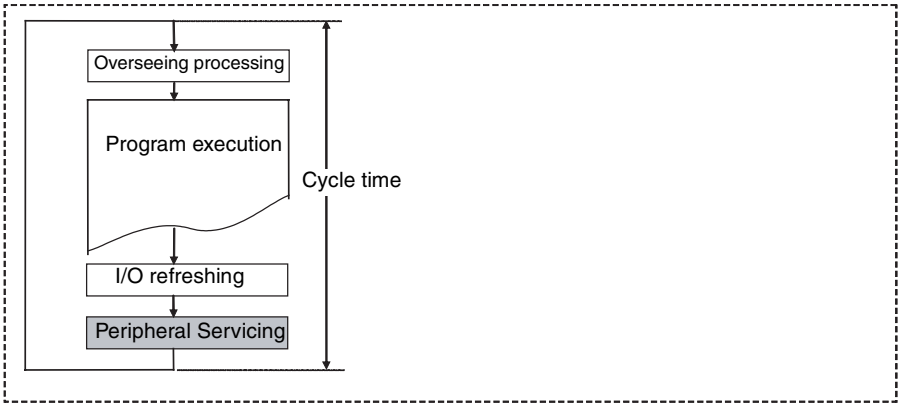
6-8-1 CPU Processing Modes

Normally, peripheral servicing (see note) is performed once at the end of each cycle (following I/O refresh) either for 4% of the cycle or a user-set time for each service. This makes it impossible to service peripheral devices at a rate faster than the cycle time, and the cycle time is increased by the time required for peripheral servicing.

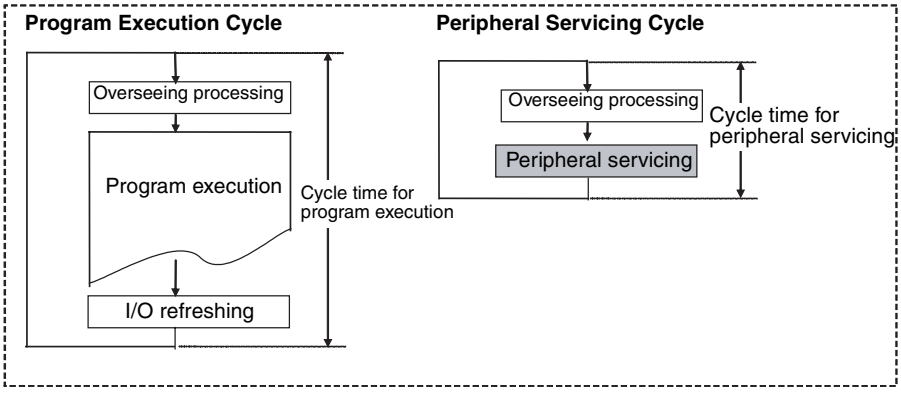
With CS1D CPU Units for Single-CPU Systems or with CS1-H or CJ1-H CPU Units, however, Parallel Processing Modes are supported that enable processing program execution in parallel with peripheral servicing. These modes enable faster peripheral servicing and shorter cycle times, especially when there is extensive peripheral servicing required. (CS1D CPU Units for Duplex-CPU Systems and CJ1M CPU Units do not support the Parallel Processing Modes.)

Note Peripheral servicing includes non-schedule services required by external devices, such as event servicing (e.g., communications for FINS commands) for Special I/O Units, CPU Bus Units, and Inner Boards (CS Series only), as well as communications port servicing for the peripheral and RS-232C ports (but not including data links and other special I/O refreshing for CPU Bus Units).

Normal Mode



Parallel Processing Modes



Parallel Processing Modes

There are two different Parallel Processing Modes: Parallel Processing with Synchronous Memory Access and Parallel Processing with Asynchronous Memory Access.

■ Parallel Processing with Asynchronous Memory Access

In this mode, I/O memory access for peripheral servicing is not synchronized with I/O memory access for program execution. In other words, all peripheral servicing is executed in parallel with program execution, including memory access. This mode will provide the fastest execution (compared to the other modes) for both program execution and event processing when there is a heavy peripheral servicing load.

■ Parallel Processing with Synchronous Memory Access

In this mode, I/O memory access for peripheral servicing is not executed in parallel with program execution, but rather is executed following program execution, just like it is in the normal execution mode, i.e., following the I/O refresh period. All other peripheral servicing is executed in parallel with program execution.

This mode will provide faster execution that the normal execution mode for both program execution and event processing. The program execution cycle time will be longer than that for Parallel Processing with Asynchronous Memory Access by the time required to refresh I/O for peripheral servicing.

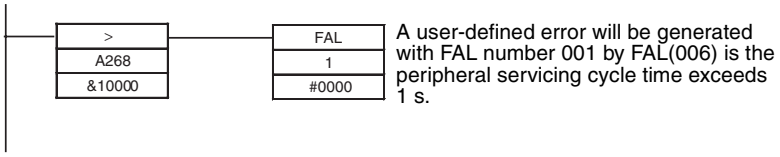
The cycle times and peripheral servicing responses for Normal, Parallel Processing with Asynchronous Memory Access, and Parallel Processing with Synchronous Memory Access are listed in the following table. (These values are for a program consisting of basic instructions with a cycle time of 10 ms and with one Ethernet Unit. These values are provided for reference only and will vary with the system.)

Item	Normal Mode	Parallel Processing with Asynchronous Memory Access	Parallel Processing with Synchronous Memory Access
Cycle time	Arbitrarily set to 1	0.9	0.9
Peripheral servicing	Arbitrarily set to 1	0.4	1.0

Note

1. Peripheral servicing includes event servicing (e.g., communications for FINS commands) for Special I/O Units, CPU Bus Units, and Inner Boards (CS Series only), as well as communications port servicing for the peripheral and RS-232C ports (but not including data links and other special I/O refreshing for CPU Bus Units).
2. The CS1G/H-CPU□□-V1, CS1-H, and CJ-series CPU Units also support a Peripheral Servicing Priority Mode that will perform peripheral servicing at a fixed cycle during program execution. It will provide faster peripheral servicing than the normal processing mode, but program execution will be slower. Event response, however, will not be as fast as the Parallel Processing Modes. Parallel Processing with Asynchronous Memory Access should thus be used whenever response to events is to be given priority in processing.
3. Peripheral servicing cycle time over errors can occur in the CPU Unit as described in *a)* and *b)*, below, when parallel processing is used. If this error occurs, the display on the Programming Device will indicate that the cycle time is too long, A40515 (Peripheral Servicing Cycle Time Over) will turn ON, and operation will stop (fatal error)

a) If the peripheral servicing cycle time exceeds 2.0 s, a cycle time over error will occur. The peripheral servicing cycle time can be monitored in A268 to detect possible errors before they occur. For example, a user-defined error can be generated using FAL number 001 if the peripheral servicing cycle time exceeds 1 s (i.e., if the contents of A268 exceeds 2710 Hex (10000 decimal)).

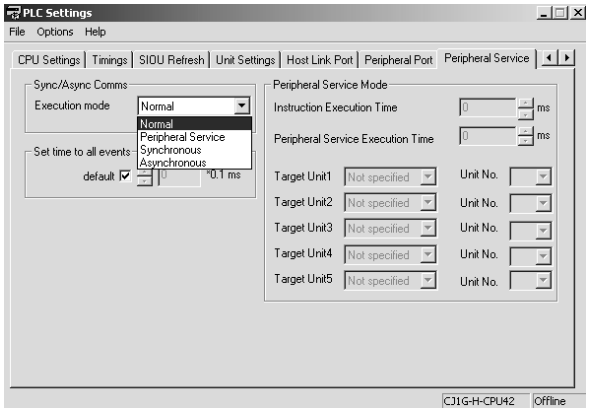


b) A peripheral servicing cycle time over error can also occur if the processing time for the instruction execution cycle (i.e., the instruction execution time) is too short. This time is stored in A266 and A267 in normal execution mode. As a guideline, if the instruction execution time is 2 ms or less, a peripheral servicing cycle time over error will occur and the parallel processing mode cannot be used. When debugging only sections of the program (which can cause a very short instruction execution time), use normal mode to prevent this error from occurring.

The Programming Console should be not be used when user applications are being run in a parallel processing mode. The Programming Console will be allocated servicing time to increase the response to Programming Console keys, and this will increase the peripheral servicing time and reduce the effectiveness of parallel processing.

PLC Setup

When using the CX-Programmer, make the settings on the Peripheral Service Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address		Name	Setting	Default	CPU Unit refresh timing
Word	Bit				
+219	08 to 15	CPU Processing Mode	00 Hex: Normal Mode 01 Hex: Parallel Processing with Synchronous Memory Access 02 Hex: Parallel Processing with Asynchronous Memory Access 05 to FF Hex: Time slice program execution time for Peripheral Servicing Priority Mode (5 to 255 ms in 1-ms increments) Settings of 03 and 04 Hex are not defined (illegal) and will cause PLC Setup errors (nonfatal).	00 Hex: Normal Mode	Start of operation

Auxiliary Area Flags and Words

Name	Address	Operation
Peripheral Servicing Cycle Time Over	A40515	Turns ON when the peripheral servicing cycle time exceeds 2 s. Operation will be stopped.
Peripheral Servicing Cycle Time	A268	Contains the peripheral servicing cycle time when one of the Parallel Processing Modes (synchronous or asynchronous memory access) is used and the PLC is in RUN or MONITOR mode. The time will be in binary between 0.0 and 2000.0 (in 0.1-ms increments).
Instruction Execution Time (Total of all slice times for program execution and all slice times for peripheral servicing)	A266 and A267	In normal mode, only the instruction execution time is included. The time is stored as a 32-bit binary value. 00000000 to FFFFFFFF Hex (unit: 0.1 ms) (0 to 429,496,729.5 ms) A266: Least-significant word A267: Most-significant word

Parallel Processing with Asynchronous Memory Access

Program Executions

Overseeing		I/O bus check and other processing 0.3 ms (0.28 ms in CJ1-H-R CPU Units)
Instruction execution time		Total execution time for all instructions
Minimum cycle time calculations		Processing time for a minimum program execution cycle time
Cyclic servicing	I/O refresh	I/O refresh time for each Unit × Number of Units
	Special I/O refresh for CPU Bus Units	Special I/O refresh time for each Unit × Number of Units
Peripheral servicing	File access	Peripheral service time set in PLC Setup (default: 4% of cycle time)

Peripheral Servicing

Overseeing		Battery check, user program memory check, etc. 0.2 ms (0.18 ms in CJ1-H-R CPU Units)
Peripheral servicing	Event servicing for Special I/O Units	Includes event servicing to access I/O memory (See note 3.) Max. of 1 s for each service.
	Event servicing for CPU Bus Units	
	Peripheral port servicing	
	RS-232C port servicing	
	Event servicing for Inner Boards (CS Series only)	
	Event servicing for communications ports (internal logic ports) that are being used (including background execution)	

Note Event servicing to access I/O memory includes 1) Servicing any received FINS commands that access I/O memory (I/O memory read/write commands with common codes beginning with 01 Hex or forced set/reset commands with common codes beginning with 23 Hex) and 2) Servicing any received C-mode commands that access I/O memory (excluding NT Links using the peripheral or RS-232C port).

Parallel Processing with Synchronous Memory Access

Program Executions

Overseeing		I/O bus check and other processing 0.3 ms (0.28 ms in CJ1-H-R CPU Units)
Instruction execution time		Total execution time for all instructions
Minimum cycle time calculations		Processing time for a minimum program execution cycle time
Cyclic servicing	I/O refresh	I/O refresh time for each Unit × Number of Units
	Special I/O refresh for CPU Bus Units	Special I/O refresh time for each Unit × Number of Units
Peripheral servicing	File access	Peripheral service time set in PLC Setup (default: 4% of cycle time)
	Event servicing requiring I/O memory access (See note.)	

Peripheral Servicing

Overseeing		Battery check, user program memory check, etc. 0.2 ms (0.18 ms in CJ1-H-R CPU Units)
Peripheral servicing	Event servicing for Special I/O Units	Except for event servicing to access I/O memory (See note 3.) Max. of 1 s for each service.
	Event servicing for CPU Bus Units	
	Peripheral port servicing	
	RS-232C port servicing	
	Event servicing for Inner Boards (CS Series only)	
	Event servicing for communications ports (internal logic ports) that are being used (including background execution)	

Note Event servicing to access I/O memory includes 1) Servicing any received FINS commands that access I/O memory (I/O memory read/write commands with common codes beginning with 01 Hex or forced set/reset commands with common codes beginning with 23 Hex) and 2) Servicing any received C-mode commands that access I/O memory (excluding NT Links using the peripheral or RS-232C port).

6-8-2 Parallel Processing Mode and Minimum Cycle Times

If a minimum cycle time is specified when a parallel processing mode is being used, a wait will be inserted after program execution until the minimum cycle time has been reached, but peripheral servicing will continue.

6-8-3 Data Concurrency in Parallel Processing with Asynchronous Memory Access

Data may not be concurrent in the following cases when using Parallel Processing with Asynchronous Memory Access.

- When more than one word is read from I/O memory using a communications command, the data contained in the words may not be concurrent.
- If an instruction reads more than one word of I/O memory and peripheral servicing is executed during execution of the instructions, the data contained in the words may not be concurrent.
- If the same word in I/O memory is read by more than instruction at different locations in the program and peripheral servicing is executed between execution of the instructions, the data contained in the word may not be concurrent.

The following steps can be used to ensure data concurrency when required.

1. Use Parallel Processing with Synchronous Memory Access
2. Use the IOSP(287) to disable peripheral servicing for where required in the program and then use IORS(288) to enable peripheral servicing again.

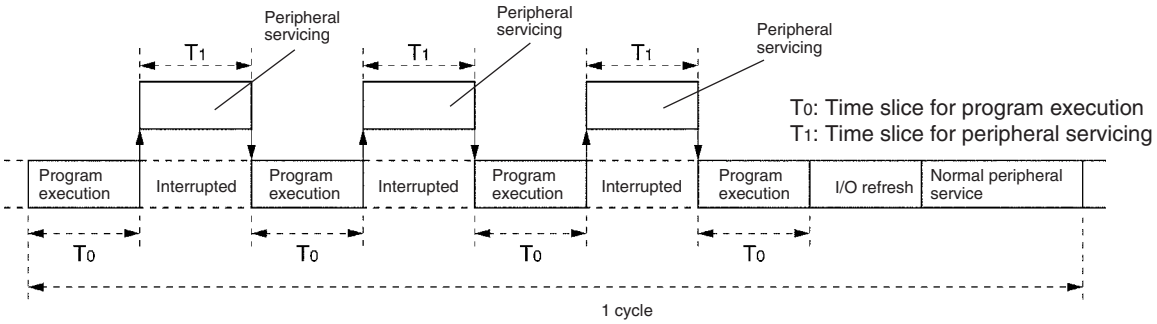
6-9 Peripheral Servicing Priority Mode

Peripheral servicing for RS-232C port, the peripheral port, the Inner Board (CS Series only), CPU Bus Units, and Special I/O Units is normally serviced only once at the end of the cycle after the I/O refresh. Either 4% of the cycle time or a user-set time is allocated to each service. A mode, however, is available that enables periodic servicing within a cycle. This mode, called the Peripheral Servicing Priority Mode, is set in the PLC Setup.

Note The Peripheral Servicing Priority Mode can be used with CJ-series CPU Unit or CS-series CPU Units, but the CS-series CS1 CPU Unit must have a lot number 001201□□□□ or later (manufacture date of December 1, 2000 or later). (Peripheral Servicing Priority Mode is not supported by CS1D for Duplex-CPU Systems.)

6-9-1 Peripheral Servicing Priority Mode

If the Peripheral Servicing Priority Mode is set, program execution will be interrupted at the specified time, the specified servicing will be performed, and program execution will be resumed. This will be repeated through program execution. Normal peripheral servicing will also be performed after the I/O refresh period.



Peripheral Servicing Priority Mode can thus be used to execute periodic servicing for specified ports or Units along with the normal peripheral servicing. This enables applications that require priority be given to peripheral servicing over program execution, such as process control applications that require rapid response for host monitoring.

- Up to five Units or ports can be specified for priority servicing. CPU Bus Units and CS/CJ Special I/O Units are specified by unit number.
- Only one Unit or port is executed during each slice time for peripheral servicing. If servicing has been completed before the specified time expires, program execution is resumed immediately and the next Unit or port is not serviced until the next slice time for peripheral servicing. It is possible, however, that the same Unit or port will be serviced more than once during the same cycle.
- Unit or ports are serviced in the order in which they are detected by the CPU Unit.

Note

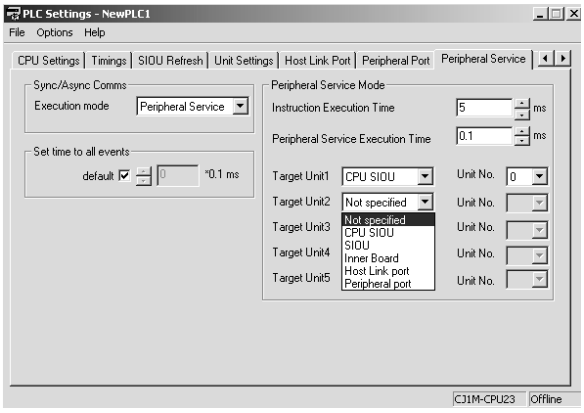
1. Even though the following instructions use the communications ports, they will be executed only once during the execution cycle even if Peripheral Servicing Priority Mode is used:
   RXD(235) (RECEIVE)
   TXD(236) (TRANSMIT)
2. If more than one word is read via a communications command, the concurrence of the read data cannot be guaranteed when Peripheral Servicing Priority Mode is used.
3. The CPU Unit might exceed the maximum cycle time when Peripheral Servicing Priority Mode is used. The maximum cycle time is set in the PLC Setup as the Watch Cycle Time setting. If the cycle time exceeds the Watch Cycle Time setting, the Cycle Time Too Long Flag (A40108) will be turned ON and PLC operation will be stopped. If the Peripheral Servicing Priority Mode is used, the current cycle time in A264 and A265 should be monitored and the Watch Cycle Time (address: +209) adjusted as required. (The setting range is 10 to 40,000 ms in 10-ms increments with a default setting of 1 s.)

PLC Setup Settings

The following settings must be made in the PLC Setup to use the Peripheral Servicing Priority Mode.

- Slice Time for Program Execution: 5 to 255 ms in 1-ms increments
- Slice Time for Peripheral Servicing: 0.1 to 25.5 ms in 0.1-ms increments
- Units and/or Ports for Priority Servicing: CPU Bus Unit (by unit No.)
  CS/CJ Special I/O Unit (by unit No.)
  Inner Board (CS Series only)
  RS-232C port
  Peripheral port

When using the CX-Programmer, make the settings on the Peripheral Service Tab Page.



The following table shows the corresponding settings when using a Programming Console.

Programming Console address		Settings	Default	Function	New setting's effectiveness
Word	Bit(s)				
+219	08 to 15	00 05 to FF (Hex)	00	00: Disable priority mode servicing 05 to FF: Time slice for instruction execution (5 to 255 ms in 1-ms increments)	Takes effect at the start of operation (Can't be changed during operation.)
	00 to 07	00 to FF (Hex)	00	00: Disable priority mode servicing 01 to FF: Time slice for peripheral servicing (0.1 to 25.5 ms in 0.1-ms increments)	
+220	08 to 15	00 10 to 1F 20 to 2F E1 FC FD (Hex)	00	00: Disable priority mode servicing 10 to 1F: CPU Bus Unit unit number (0 to 15) + 10 (Hex) 20 to 7F: Special I/O Unit unit number (0 to 95) + 20 (Hex) E1: Inner Board FC: RS-232C port FD: Peripheral port	
	00 to 07		00		
+221	08 to 15		00		
	00 to 07		00		
+222	08 to 15		00		

- Operation and errors will be as shown below depending on the settings in the PLC Setup.

- The setting cannot be made from the CX-Programmer for CS1 or CJ1 CPU Units. The setting can be made from CX-Programmer Ver. 2.1 or higher for CS1-H and CJ1-H CPU Units.

Conditions			CPU Unit operation	PLC Setup errors
Time Slice for Peripheral Servicing	**Time Slice for Instruction Execution**	**Specified Units and Ports**		
01 to FF: (0.1 to 25.5 ms)	05 to FF: (5 to 255 ms)	All correct settings	Peripheral Servicing Priority Mode	None
		00 and correct settings		
		Correct, but redundant settings		
		Some illegal settings	Peripheral Servicing Priority Mode for items with correct settings	Generated
		All 00 settings	Normal operation	Generated
		00 and illegal settings		
		All illegal settings		
00	00	---	Normal operation	None
Any other		---	Normal operation	Generated

Note If an error is detected in the PLC Setup, A40210 will turn ON and a non-fatal error will occur.

Auxiliary Area Information

If the slice times are set for program execution and peripheral servicing, the total of all the program execution and peripheral servicing slice times will be stored in A266 and A267. This information can be used as a reference in making appropriate adjustments to the slice times.

When Peripheral Servicing Priority Mode is not being used, the program execution time will be stored. This value can be used in determining appropriate settings for the slice times.

Words	Contents	Meaning	Refreshing
A266 and A267	00000000 to FFFFFFFF Hex (0 to 4294967295 decimal)	Total of all slice times for program execution and all slice times for peripheral servicing. 0.0 to 429,496,729.5 ms (0.1-ms increments) A267 (Most-significant bytes) | A266 (Least-significant bytes) — Value is stored as 32-bit binary (8-digit hexadecimal) value	The contents is refreshed each cycle and is cleared at the beginning of operation.

6-9-2 Temporarily Disabling Priority Mode Servicing

Data concurrence is not guaranteed at the following times if Peripheral Servicing Priority Mode is used.

- When more than one word is read from a peripheral device using a communications command. The data may be read during different peripheral servicing time slices, causing the data to not be concurrent.
- When instructions with long execution times are used in the program, e.g., when transferring large quantities of I/O memory data. The transfer operation may be interrupted for peripheral servicing, causing the data to not be concurrent. This can be true when words being written by the program are read from a peripheral before the write has been completed or when

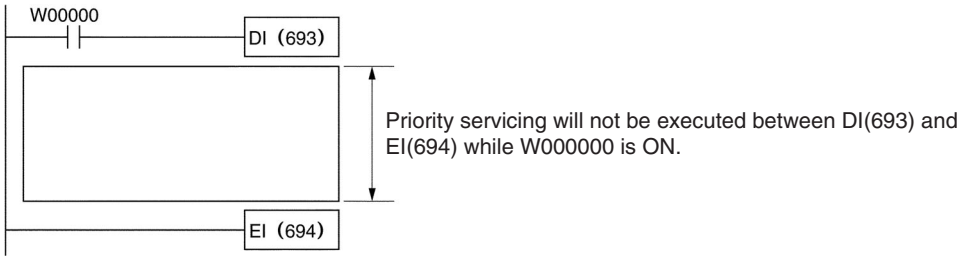
words being read by the program are written from a peripheral before the read has been completed.

- When two instructions access the same words in memory. If these words are written from a peripheral device between the times the two instructions are executed, the two instructions will read different values from memory.

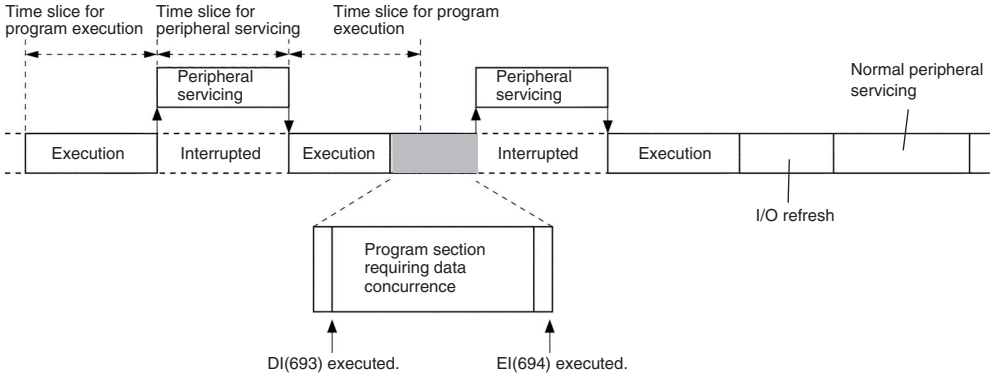
When data concurrence must be ensured, the DISABLE INTERRUPTS and ENABLE INTERRUPTS instructions (DI(693) and EI(694)) can be used for CS1 or CJ1 CPU Units to prevent priority servicing during required sections of the program, as shown in the following example.

CS1-H, CJ1-H, and CJ1M CPU Units, and CS1D CPU Units for Single-CPU Systems: Use IOSP(287) and IORS(288)

CS1G/H-CPU□□(-V1) and CS1□-CPU□□ CPU Units: Use DI (693) and EI(694).



Operation



Note

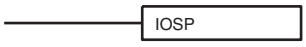
1. DI(693) and IOSP(287) will disable not only interrupts for priority servicing, but also all other interrupts, including I/O, scheduled, and external interrupts. All interrupts that have been generated will be executed after the cyclic task has been executed (after END(001) execution) unless CLI(691) is executed first to clear the interrupts.
2. Disabling interrupts with DI(693) or IOSP(287) is effective until EI(694) or IORS(288) is executed, until END(001) is executed, or until PLC operation is stopped. Program sections can thus not be created that go past the end of a task or cycle. Use DI(693) and EI(694) or IOSP(287) and IORS(288) in each cyclic task when necessary to disable interrupts in more than one cycle or task.

CS1D CPU Units for Single-CPU Systems and CS1-H, CJ1-H, and CJ1M CPU Units

IOSP(287)

When executed, IOSP(287) disables peripheral servicing. Peripheral servicing will remain disabled if IOSP(287) is executed when it is already disabled.

Symbol



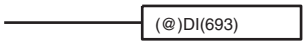
IORS(288) When executed, IORS(288) enables disables peripheral servicing that was disabled with IOSP(287). Peripheral servicing will remain enabled if IORS(288) is executed when it is already enabled.

Symbol

CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units

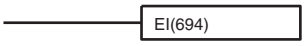
DI(693) When executed, DI(693) disables all interrupts (except for interrupts for the power interrupt task), including interrupts for priority servicing, I/O interrupts, scheduled interrupts, and external interrupts. Interrupts will remain disabled if DI(693) is executed when they are already disabled.

Symbol



EI(694) When executed, EI(694) enables all interrupts (except for interrupts for the power interrupt task), including interrupts for priority servicing, I/O interrupts, scheduled interrupts, and external interrupts. Interrupts will remain enabled if EI(694) is executed when they are already enabled.

Symbol



EI(694) requires no execution condition.

6-10 Battery-free Operation

The CS-series and CJ-series PLCs can be operated without a Battery installed (or with an exhausted Battery). The procedure used for battery-free operation depends on the following items.

- CPU Unit
- Whether or not I/O memory (e.g., CIO Area) is maintained or not
- Whether or not the DM and EM Areas are initialized at startup
- Whether or not the DM and EM Areas are initialized from the user program

The above differences are summarized in the following table.

CPU Unit	Not maintaining I/O memory			Maintaining I/O memory
	No initializing DM and EM Areas at startup	Initializing DM and EM Areas at startup		
		From user program	Not from user program	
CS1-H, CJ1-H, CJ1M, or CS1D	Use normal operation (using flash memory) or a Memory Card.		Use automatic transfer from a Memory Card at startup. (Turn ON pin 2 of DIP switch.)	Not possible with any method. A Battery must be installed.
CS1G/H-CPU□□(-V1) and CJ1□-CPU□□	Use automatic transfer from a Memory Card at startup. (Turn ON pin 2 of DIP switch.)			

Note 1. When using battery-free operation, disable detecting a low battery voltage in the PLC Setup regardless of the method used for battery-free operation.

2. If a Battery is not connected or the Battery is exhausted, the following restrictions will apply to CPU Unit operation. This is true regardless of the CPU Unit being used.
   - The status of the Output OFF Bit (A50015) will be unreliable. When the Output OFF Bit is ON, all Output Unit outputs will be turned OFF. Include the following instructions in the ladder program to prevent all Output Unit from outputs turning OFF when the power is turned ON.

   - The contents of I/O memory (including the HR, DM, and EM Areas) may not be correctly maintained. Therefore, set the PLC Setup so that the status of the I/O Memory Hold Flag (A50012) and the Forced Status Hold Flag (A50013) are not maintained when power is turned ON.
   - The clock function cannot be used. The clock data in A351 to A354 and the startup time in A510 and A511 will not be dependable. The files dates on files written to the Memory Card from the CPU Unit will also not be dependable.
   - The following data will be all-zeros at startup: Power ON Time (A523), Power Interruption Time (A512 and A513), and Number of Power Interruptions (A514).
   - The Error Log Area in A100 to A199 will not be maintained.
   - The current EM bank will always be 0 at startup.
   - There will be no files left in the EM file memory at startup and the file memory functions cannot be used. The EM file memory must be reset in the PLC Setup and the EM file memory must be reformatted to use it.

CS1-H, CJ1-H, CJ1M, or CS1D CPU Units

Battery-free operation is possible with normal operation. The user program and parameter data are automatically backed up to flash memory in the CPU Unit and are automatically restored from flash memory at startup. In this case, the I/O memory will not be maintained and the DM and EM Areas must be initialized from the user program.

Battery-free operation is also possible by automatically transferring data from a Memory Card at startup. (With a Memory Card, the DM and EM Area data can be included.)

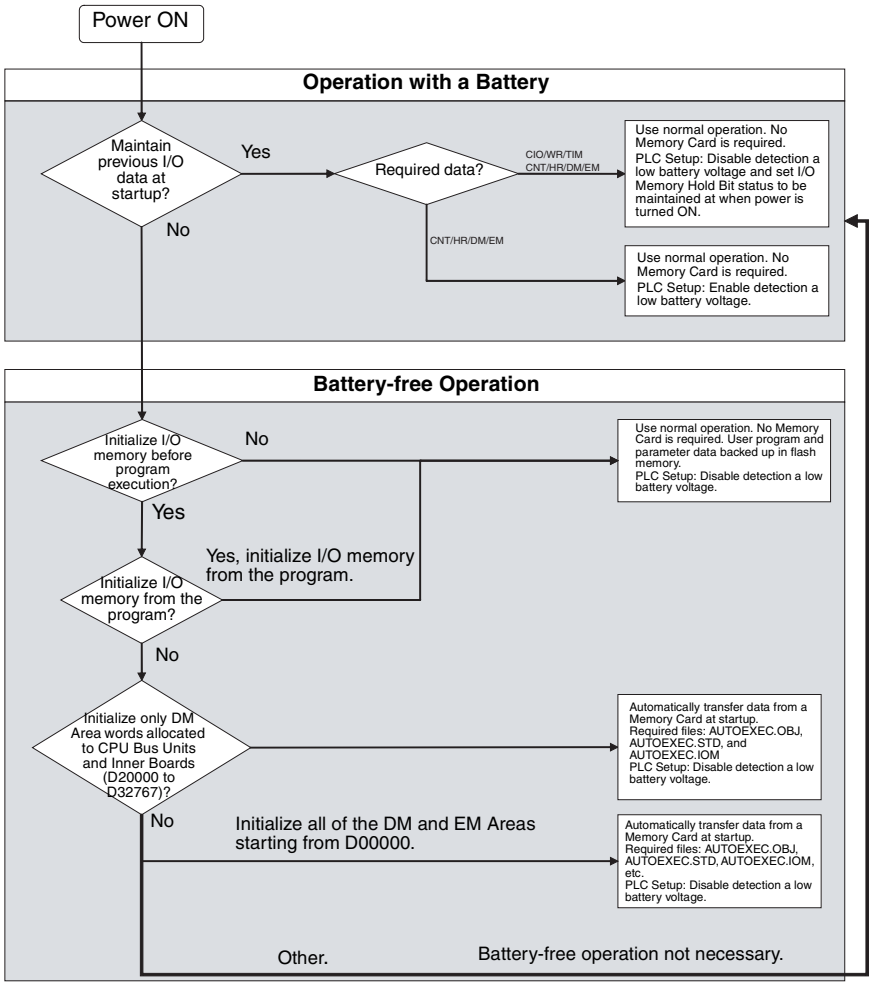
CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ Units

Battery-free operation is possible by automatically transferring data from a Memory Card at startup. In this case, the I/O memory will not be maintained. (With a Memory Card, the DM and EM Area data can be included.)

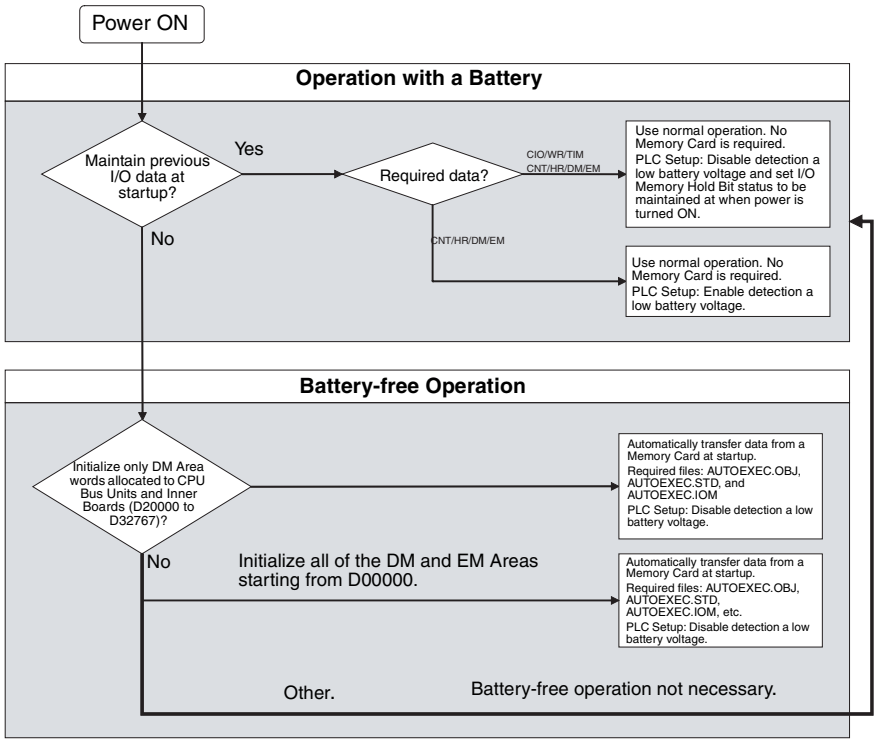
Procedure

The following flowcharts show the procedures for the two types of CPU Unit.

CS1-H, CJ1-H, CJ1M, or CS1D CPU Units



CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units

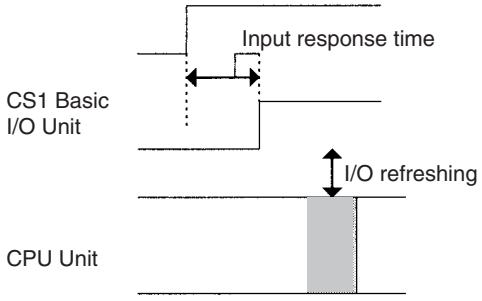


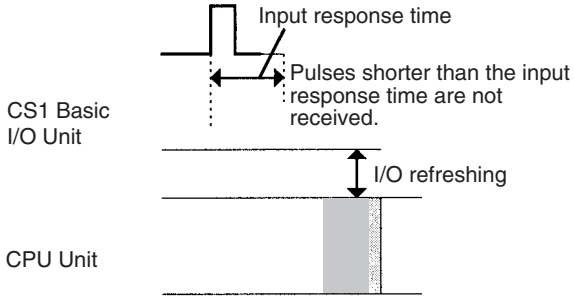
6-11 Other Functions

6-11-1 I/O Response Time Settings

The input response times for CS/CJ Basic I/O Units can be set by Rack and Slot number. Increasing the input response time reduces the effects of chattering and noise. Decreasing the input response time (but keeping the pulse width longer than the cycle time) allows reception of shorter input pulses.

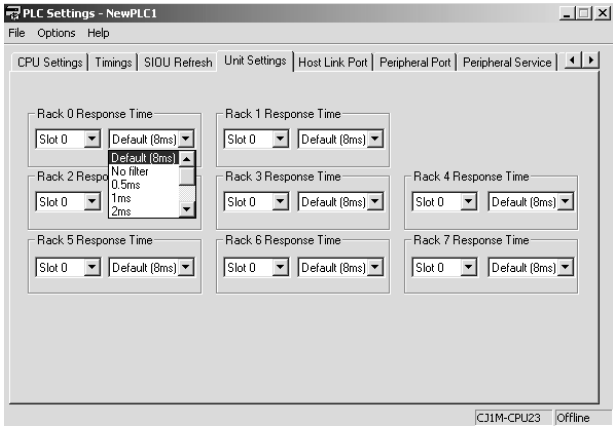
Note With CS-series CPU Units, pulses shorter than the cycle time can be input with the high-speed inputs available in some C200H High-density I/O Units or with a High-speed Input Unit. Refer to *6-1-4 High-speed Inputs* for details.





PLC Setup

When using the CX-Programmer, make the settings on the Unit Settings Tab Page.



The following table shows the corresponding settings when using a Programming Console.

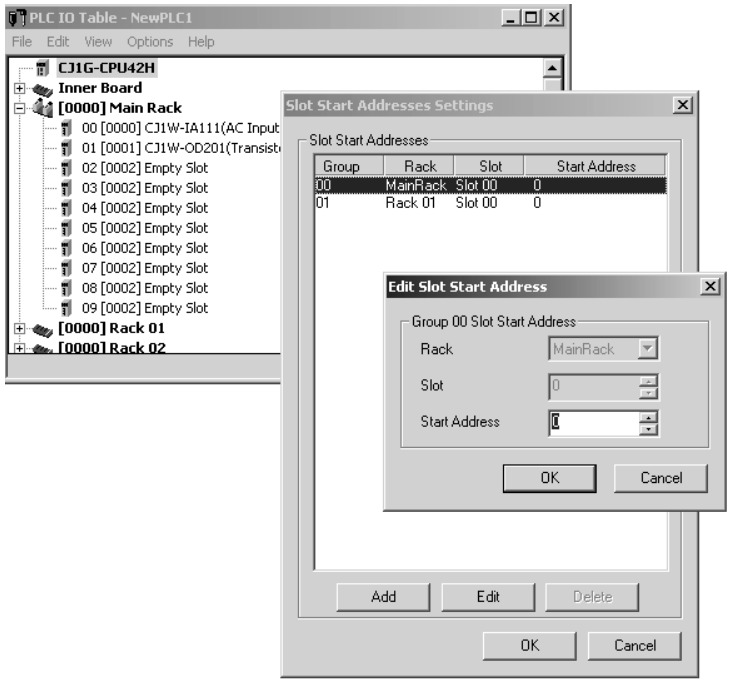
The input response times for the 80 slots in a CS/CJ PLC (Rack 0 Slot 0 through Rack 7 slot 9) can be set in the 80 bytes in addresses 10 through 49.

Programming Console address	Name	Setting (Hex)	Default (Hex)
+10, bits 0 to 7	CS/CJ Basic I/O Unit Input Response Time for Rack 0, Slot 0	00: 8 ms 10: 0 ms 11: 0.5 ms 12: 1 ms 13: 2 ms 14: 4 ms 15: 8 ms 16: 16 ms 17: 32 ms	00 (8 ms)
:	:	:	:
49, bits 8 to 15	CS/CJ Basic I/O Unit Input Response Time for Rack 7, Slot 9	Same as above.	00 (8 ms)

6-11-2 I/O Area Allocation

A Programming Device can be used to set the first word for I/O allocation in Expansion Racks (CS/CJ Expansion Racks and C200H Expansion I/O Racks). This function allows each Rack's or each slot's I/O allocation area to be fixed within the range CIO 0000 to CIO 0999. (The first words are allocated by rack number and slot number.)

Set the first words in I/O tables from the CX-Programmer.


PLC IO Table - NewPLC1
File Edit View Options Help
CJ1G-CPU42H
Inner Board
[0000] Main Rack
00 [0000] CJ1W-IA111(AC Input
01 [0001] CJ1W-OD201(Transist
02 [0002] Empty Slot
03 [0002] Empty Slot
04 [0002] Empty Slot
05 [0002] Empty Slot
06 [0002] Empty Slot
07 [0002] Empty Slot
08 [0002] Empty Slot
09 [0002] Empty Slot
[0000] Rack 01
[0000] Rack 02
Slot Start Addresses Settings
Slot Start Addresses
Group Rack Slot Start Address
00 MainRack Slot 00 0
01 Rack 01 Slot 00 0
Edit Slot Start Address
Group 00 Slot Start Address
Rack MainRack
Slot 0
Start Address 0
OK Cancel
Add Edit Delete
OK Cancel

SECTION 7
Program Transfer, Trial Operation, and Debugging

This section describes the processes used to transfer the program to the CPU Unit and the functions that can be used to test and debug the program.

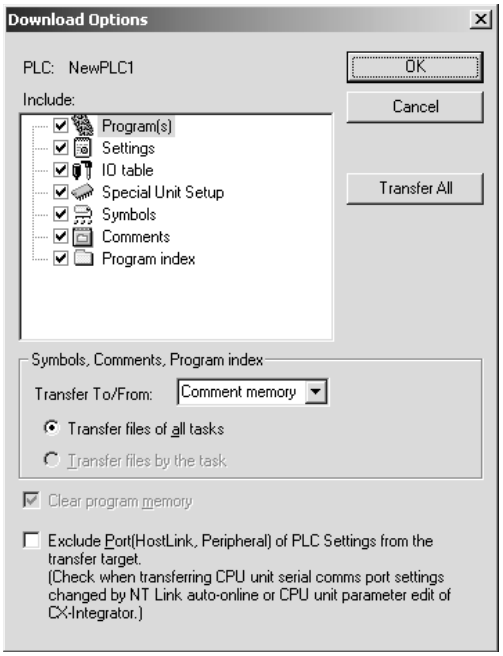
7-1 Program Transfer

A Programming Device is used to transfer the programs, PLC Setup, I/O memory data, and I/O comments to the CPU Unit with the CPU Unit in PROGRAM mode.

Program Transfer Procedure for CX-Programmer

1,2,3...

1. Select ***PLC, Transfer,*** and then ***To PLC.*** The Download Options Dialog Box will be displayed.
2. Specify the items for the transfer from among the following: Programs, Settings (PLC Setup), I/O table, Symbols, Comments, and Program index.



When a CPU Unit with unit version 3.0 or later is being used, the comment memory can be selected as the transfer destination of the comment information. When the comment memory is selected, the symbol table, comments, and program index file are also stored in the CPU Unit.

3. Click the **OK** button.

The program can be transferred using either of the following methods.

- Automatic transfer when the power is turned ON

When the power is turned ON, the AUTOEXEC.OBJ file in the Memory Card will be read to the CPU Unit (pin 2 on the DIP switch must be ON).

- Program replacement during operation

The existing program file can be replaced with the program file specified in the Auxiliary Area by turning ON the Replacement Start Bit in the Auxiliary Area (A65015) from the program while the CPU Unit is in operation. Refer to *SECTION 5 File Memory Functions* for details.

Note If CX-Programmer version 4.0 or higher is used with CS/CJ-series CPU Units Ver. 2.0 or later, task programs can be downloaded individually. For details, refer to *1-4-1 Downloading and Uploading Individual Tasks* in the *CS Series PLCs Operation Manual* or the *CJ Series PLCs Operation Manual*.

7-2 Trial Operation and Debugging

7-2-1 Forced Set/Reset

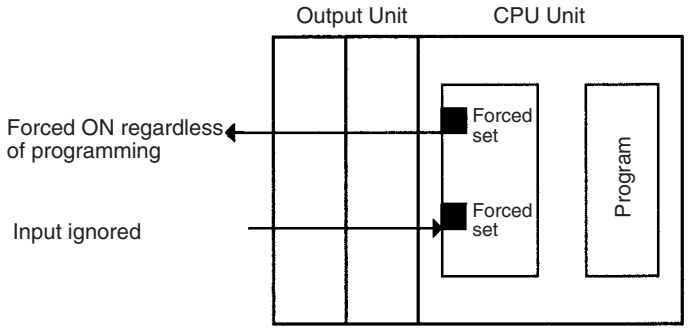
A Programming Device can force-set (ON) or reset (OFF) specified bits (CIO Area, Auxiliary Area, HR Area, and timer/counter Completion Flags). Forced status will take priority over status output from the program or I/O refreshing. This status cannot be overwritten by instructions, and will be stored regardless of the status of the program or external inputs until it is cleared from a Programming Device.

Force-set/reset operations are used to force input and output during a trial operation or to force certain conditions during debugging.

Force-set/reset operations can be executed in either MONITOR or PROGRAM modes, but not in RUN mode.

Note Turn ON the Forced Status Hold Bit (A50013) and the IOM Hold Bit (A50012) at the same time to retain the status of bits that have been force-set or reset when switching the operating mode.

Turn ON the Forced Status Hold Bit (A50013) and the IOM Hold Bit (A50012), and set the Forced Status Hold Bit at Startup setting PLC Setup to retain the status of the Forced Status Hold Bit hold to retain the status of bits that have been force-set or reset when turning OFF the power.



The following areas can be force-set and reset.

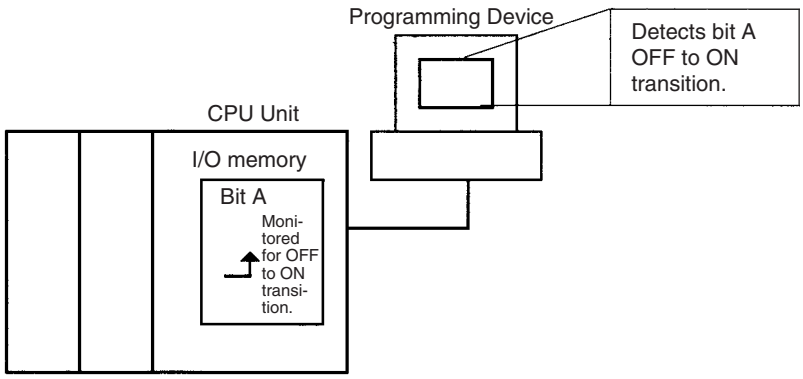
CIO (I/O bits, data link bits, CPU Bus Unit bits, Special I/O Unit bits, Inner Board bits, SYSMAC BUS bits, Optical I/O Unit bits, work bits), WR Area, Timer Completion Flags, HR Area, Counter Completion Flags. (The Inner Board, SYSMAC BUS, and I/O Terminal Areas are supported by the CS-series CPU Units only.)

Programming Device Operation

- Select bits for forced setting/resetting.
- Select forced set or forced reset.
- Clear forced status (including clearing all forced status at the same time).

7-2-2 Differential Monitoring

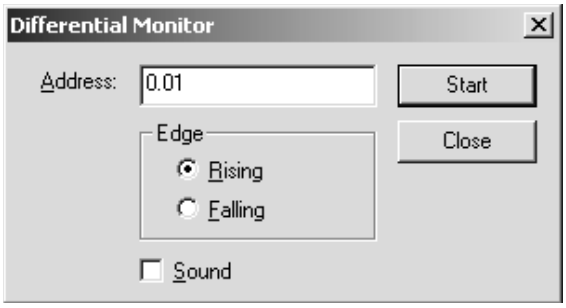
When the CPU Unit detects that a bit set by a Programming Device has changed from OFF to ON or from ON to OFF, the results are indicated in the a Differentiate Monitor Completed Flag (A50809). The Flag will turn ON when conditions set for the differential monitor have been met. A Programming Device can monitor and display these results on screen.



Programming Device Operation for CX-Programmer

1,2,3... 1. Right-click the bit for differential monitoring.

2. Click ***Differential Monitor*** from the PLC Menu. The Differential Monitor Dialog Box will be displayed.



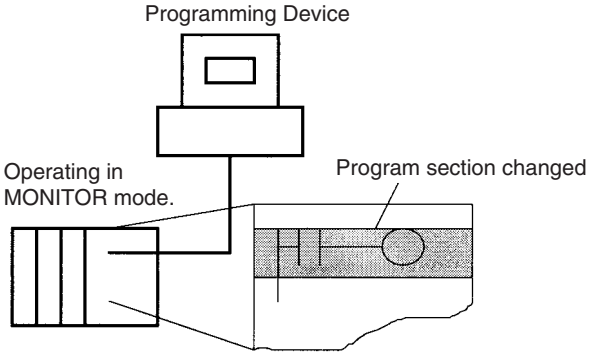
3. Click ***Rising*** or ***Falling.***
4. Click the **Start** button. The buzzer will sound when the specified change is detected and the count will be incremented.
5. Click the **Stop** button. Differential monitoring will stop.

7-2-3 Online Editing

The Online Editing function is used to add to or change part of a program in a CPU Unit directly from the Programming Devices when the CPU Unit is in MONITOR or PROGRAM mode. Additions or changes are made one instruction at a time for the Programming Console and one or more program sections at a time from the CX-Programmer. The function is thus designed for minor program changes without stopping the CPU Unit.

Online editing is possible simultaneously from more than one computer running the CX-Programmer as well as from a Programming Console as long as different tasks are edited.

Online Editing



The cycle time will be increased by from one to several cycle times if the program in the CPU Unit is edited online in MONITOR mode.

The cycle time will also be increased to back up data in the flash memory after online editing. The BKUP indicator will be lit during this period. The progress of the backup is displayed on the CX-Programmer. The increases per cycle are listed in the following table.

Note The CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units do not support the flash memory functions.

CPU Unit	Increase in cycle time (guidelines)	
	Online editing	Backup to flash memory
CS1 CPU Units pre-EV1	90 ms max.	Not supported.
CS1 CPU Units EV1 or higher	12 ms max.	
CS1-H CPU Units		4% or cycle time
CS1D CPU Units		
CS1 CPU Units		Not supported.
CJ1-H CPU Units		4% or cycle time
CJ1M CPU Units		

There is a limit to the number of edits that can be made consecutively. The actual number depends on the type of editing that is performed, but the following can be used as guidelines.

CJ1M-CPU□□:	40 edits
CS1G-CPU□□H/CJ1G-CPU□□H:	160 edits
CS1H-CPU□□H/CJ1H-CPU□□H/CS1D-CPU□□H/ CS1D-CPU□□S:	400 edits

A message will be displayed on the CX-Programmer or Programming Console if the limit is exceeded, and further editing will not be possible until the CPU Unit has completed backing up the data.

Note This limit does not apply to the CS1G/H-CPU□□ (-V1) and CJ1□-CPU□□ CPU Units.

Task Size and Cycle Time Extension

The relation to the size of the task being edited to cycle time extension is as follows:

When using a version 1 or higher CS1 CPU Unit, CS1-H CPU Unit, CS1D CPU Unit, CJ1 CPU Unit, or CJ1M CPU Unit, the length of time that the cycle time is extended due to online editing is almost unaffected by the size of the task (program) being edited.

When using a pre-EV1 CS1 CPU Unit, the size of the task that is being edited will determine the length of time that a program will be stopped for online editing. By splitting the program into smaller tasks, the amount of time that the cycle is extended will be shorter using the Online Editing function than with previous PLC models.

Precautions

The cycle time will be longer than normal when a program is overwritten using Online Editing in MONITOR mode, so make sure that the amount of time that it is extended will not exceed the cycle monitoring time set in the PLC Setup. If it does exceed the monitoring time, then a Cycle Time Over error will occur, and the CPU Unit will stop. Restart the CPU Unit by selecting PROGRAM mode first before changing to RUN or MONITOR mode.

Note If the task being edited online contains a block program, then standby status created by WAIT(805), TIMW(813), TIMWX(816), CNTW(814), CNTWX(817), TMHW(815), or TMHWX(818) will be cleared by BPPS, and the next execution will be from the beginning.

Online Editing from CX-Programmer

1,2,3...

1. Display the program section that will be edited.
2. Select the instructions to be edited.
3. Select ***Program, Online Edit,*** and then ***Begin.***
4. Edit the instructions.
5. Select ***Program, Online Edit,*** and then ***Send Changes*** The instructions will be check and, if there are no errors, they will be transferred to the CPU Unit. The instructions in the CPU Unit will be overwritten and cycle time will be increased at this time.

⚠ **Caution** Proceed with Online Editing only after verifying that the extended cycle time will not affect operation. Input signals may not be input if the cycle time is too long.

Temporarily Disabling Online Editing

It is possible to disable online editing for a cycle to ensure response characteristics for machine control in that cycle. Online editing from the Programming Device will be disabled for one cycle and any requests for online editing received during that cycle will be held until the next cycle.

Online editing is disabled by turning ON the Online Editing Disable Bit (A52709) and setting the Online Editing Disable Bit Validator (A52700 to A52707) to 5A. When these settings have been made and a request for online editing is received, online editing will be put on standby and the Online Editing Wait Flag (A20110) will be turned ON.

When the Online Editing Disable Bit (A52709) is turned OFF, online editing will be performed, the Online Editing Processing Flag (A20111) will turn ON, and the Online Editing Wait Flag (A20110) will turn OFF. When online editing has been completed, the Online Editing Processing Flag (A20111) will turn OFF.

Online editing can also be temporarily disabled by turning ON the Online Editing Disable Bit (A52709) while online editing is being performed. Here too, the Online Editing Wait Flag (A20110) will turn ON.

If a second request for online editing is received while the first request is on standby, the second request will not be recorded and an error will occur.

Online editing can also be disabled to prevent accidental online editing. As described above, disable online editing by turning ON the Online Editing Dis-

able Bit (A52709) and setting the Online Editing Disable Bit Validator (A52700 to A52707) to 5A.

Enabling Online Editing from a Programming Device

When online editing cannot be enabled from the program, it can be enabled from the CX-Programmer.

1,2,3... 1. Performing Online Editing with the CX-Programmer

If operations continue with online editing in standby status, CX-Programmer may go offline. If this occurs, reconnect the computer to the PLC and turn OFF the Online Edit Disable Bit (A52709).

2. Performing Online Editing with a Programming Console

If online editing is executed from a Programming Console and the online editing standby status cannot be cleared, the Programming Console will be locked out and Programming Console operations will not be possible.

In this case, connect the CX-Programmer to another serial port and turn OFF the Online Edit Disable Bit (A52709). The online editing will be processed and Programming Console operations will be possible again.

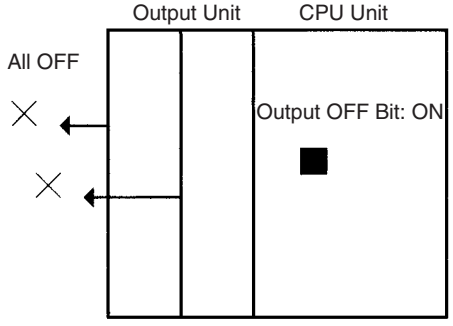
Related Auxiliary Bits/Words

Name	Address	Description
Online Edit Disable Bit Validator	A52700 to A52707	Validates the Online Edit Disable Bit (A52709). Not 5A: Online Edit Disable Bit invalid 5A: Online Edit Disable Bit valid
Online Edit Disable Bit	A52709	To disable online editing, turn this bit ON and set the Online Edit Disable Bit Validator (A52700 to A52707) to 5A.
Online Editing Wait Flag	A20110	ON when an online editing process is on standby because online editing is disabled.
Online Editing Processing Flag	A20111	ON when an online editing process is being executed.

Turning OFF Outputs

If the Output OFF Bit (A50015) is turned ON through the OUT instruction or from a Programming Device, all outputs from all Output Units will be turned OFF (this applies to the built-in general-purpose or pulse outputs on CJ1M CPU Units as well), and the INH indicator on the front of the CPU Unit will turn ON.

The status of the Output OFF Bit is maintained even if power is turned OFF and ON.



7-2-4 Tracing Data

The Data Trace function samples specified I/O memory data using any one of the following timing methods, and it stores the sampled data in Trace Memory, where they can be read and checked later from a Programming Device.

- Specified sampling time (10 to 2,550 ms in 10-ms units)
- One sample per cycle
- When the TRACE MEMORY SAMPLING instruction (TRSM) is executed

Up to 31 bits and 6 words in I/O memory can be specified for sampling. Trace Memory capacity is 4,000 words.

Basic Procedure

1,2,3...

1. Sampling will start when the parameters have been set from the CX-Programmer and the command to start tracing has been executed.
2. Sampled data (after step 1 above) will be traced when the trace trigger condition is met, and the data just after the delay (see note 1) will be stored in Trace Memory.
3. Trace Memory data will be sampled, and the trace ended.

Note Delay value: Specifies how many sampling periods to offset the sampling in Trace Memory from when the Trace Start Bit (A50814) turns ON. The setting ranges are shown in the following table.

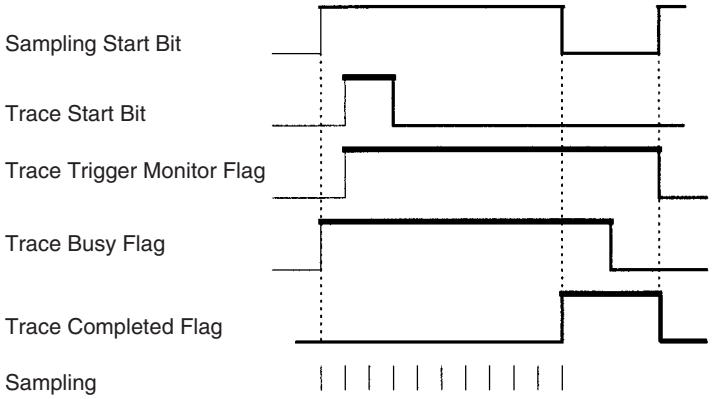
No. of words sampled	Setting range
0	–1999 to 2000
1	–1332 to 1333
2	–999 to 1000
3	–799 to 800
4	–665 to 666
5	–570 to 571
6	–499 to 500

Positive delay: Store data delayed by the set delay.

Negative delay: Store previous data according go to the set delay.

Example: Sampling at 10 ms with a –30 ms delay time yields –30 × 10 = 300 ms, so data 300 ms before the trigger will be stored.

Note Use a Programming Device to turn ON the Sampling Start Bit (A50815). Never turn ON this bit from the user program.



The following traces can be executed.

Scheduled Data Trace

A scheduled data trace will sample data at fixed intervals. Specified sampling times are 10 to 2,550 ms in 10-ms units. Do not use the TRSM instruction in the user program and be sure to set the sampling period higher than 0.

One-cycle Data Trace

A one-cycle data trace will sample I/O refresh data after the end of the tasks in the full cycle. Do not use the TRSM instruction in the user program and be sure to set the sampling period higher than 0.

Data Trace via TRSM

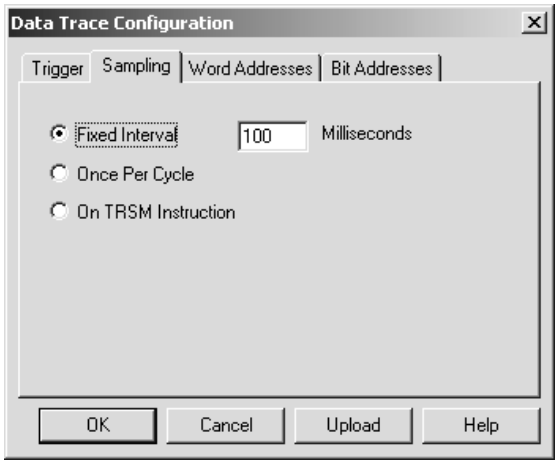
A sample will be taken once when the TRACE MEMORY SAMPLING instruction (TRSM) instruction is executed. When more than one TRSM instruction is used in the program, a sample will be taken each time the TRSM instruction is executed after the trace trigger condition has been met.

Data Trace Procedure

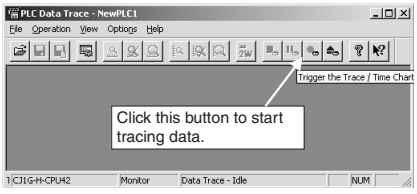
Use the following procedure to execute a trace.

1,2,3...

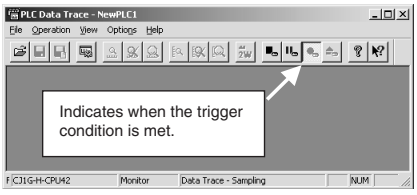
1. Use the CX-Programmer to set trace parameters. (Select *PLC - Data Trace* or ***PLC - Time Chart Monitor*** to open the Data Tracing Configuration Window, and then select ***Execute - Set***). Set the address of the sampled data, sampling period, delay time, and trigger conditions.



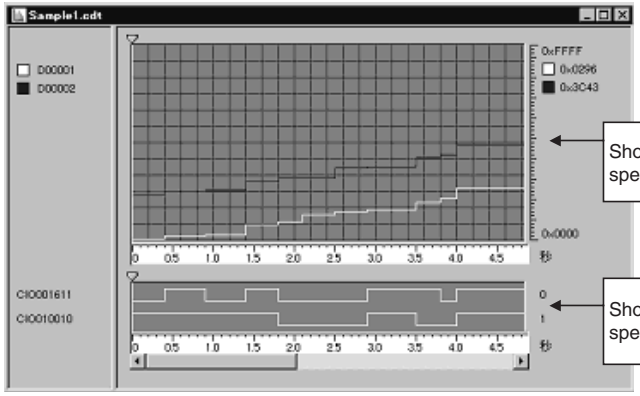
2. Use CX-Programmer to start sampling or turn ON the Sampling Start Bit (A50815).



3. Put the trace trigger condition into effect.



4. End tracing.



Appendix A

PLC Comparison Charts: CJ-series, CS-series, C200HG/HE/HX, CQM1H, CVM1, and CV-series PLCs

Functional Comparison

Item			CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Basic features	Capacity	No. of I/O points	2,560 points			5,120 points	1,184 points	6,144 points	512 points
		Program capacity	250 Ksteps One step is basically equivalent to one word. Refer to the end of *10-5 Instruction Execution Times and Number of Steps* in the *Operation Manual* for details.				2 Kwords (63.2 Kwords for -Z)	62 Kwords	15.2 Kwords
		Max. data memory	32 Kwords			32 Kwords	6 Kwords	24 Kwords	6 Kwords
		I/O bits	160 words (2,560 bits)			320 words (5,120 bits)	40 words (640 bits)	128 words (2,048 bits)	32 words (512 bits)
		Work bits	2,644 words (42,304 bits) + WR: 512 words (8,192 bits) = 3,156 words (50,496 bits)				408 words (6,528 bits)	168 words (2,688 bits) +400 words (6,400 bits)	158 words (2,528 bits)
		Holding bits	512 words (8,192 bits)				100 words (1,600 bits)	300 words (4,800 bits) Max.: 1, 400 words (2, 400 bits)	100 words (1,600 bits)
		Max. extended data memory	32 Kwords × 13 banks				6 Kwords × 3 banks (6 Kwords × 16 banks for -Z)	32 Kwords × 8 banks (Optional)	6 Kwords
		Max. No. timers/ counters	4,096 each				Timers/ counters combined: 512	1,024	Timers/ counters combined: 512
	Processing speed	Basic instructions (LD)	0.016 μs min.	0.02 μs min.	0.1 μs min.	0.02 μs min.	0.104 μs min.	0.125 μs min.	0.375 μs min.
		Special instructions (MOV)	0.144 μs min.	0.18 μs min.	0.3 μs min.	0.18 μs min.	0.417 μs min.	4.3 μs min.	17.7 μs
		System overhead time	0.13 ms min. in normal mode, 0.28 ms in parallel processing mode	0.3 ms min. in normal mode, 0.3 ms in parallel processing mode	0.5 ms	0.3 ms min. in normal mode, 0.3 ms in parallel processing mode	0.7 ms	0.5 ms	0.7 ms
		Delay during Online Edit (write)	Approx. 8 ms	Approx. 11 ms for CPU4□ and 8 ms for CPU6□	Approx. 14 ms	Approx. 11 ms for CPU4□ and 8 ms for CPU6□	80 ms (160 ms for -Z)	500 ms	250 ms
Structure	Screw mounting		No			Yes	Yes	Yes	No
	DIN Track mounting		Yes			Yes	Yes	No	Yes
	Backplanes		No			Yes	Yes	Yes	No
	Size (H × D, mm)		90 × 65			130 × 123	130 × 118	250 × 100	110 × 107

Item		CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
		CJ1-H-R	CJ1-H	CJ1-M				
Number of Units/Racks	I/O Units	40 Units			89 Units (Including Slave Racks)	10 or 16 Units	64 Units (8 Racks × 8 Units)	16 Units
	CPU Bus Units	16 Units			16 Units	None	16 Units	None
	Expansion I/O Racks	3 Racks			7 Racks	3 Racks	7 Racks	1 Rack
Task function		Yes				No	No	No
CPU processing mode (program execution and peripheral servicing)	Normal Mode	Yes				---	---	---
	Peripheral Servicing Priority Mode	Yes				---	---	---
	Parallel Processing with Synchronous Memory Access	Yes	Yes	No	Yes	No	No	No
	Parallel Processing with Asynchronous Memory Access	Yes	Yes	No	Yes	No	No	No
I/O refresh format	Cyclic refreshing	Yes				Yes	Yes	Yes
	Scheduled refreshing	No				No	Yes	No
	Zero-cross refreshing	No				No	Yes	No
	Immediate refreshing	Yes				No	Yes	No
	Immediate refreshing using IORF instruction	Yes				Yes	Yes	Yes
	Immediate refreshing using FIORF instruction	Yes	No			No	No	No
Clock function		Yes				Yes	Yes	Yes (Memory Cassette required)
RUN output		Yes (Depending on Power Supply Unit)				Yes (Depending on Power Supply Unit)	Yes	No
Startup Mode (for default PLC Setup setting when no Programming Console is connected)		RUN mode				RUN mode	RUN mode	PROGRAM mode
Disabling Power Interrupt Processing		No				No	Yes	No
Battery-free operation		Enabled using Memory Card or flash memory.				Memory Card	Memory Card	Memory Cassette
Automatic backup to flash memory (user program and parameter area)		Yes				No	No	No
Restart continuation		No				No	Yes	No

Item		CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
		CJ1-H-R	CJ1-H	CJ1-M				
External memory	Medium	Memory Card (Flash ROM)				Memory Cassette (EEPROM, EPROM)	Memory Card (RAM, EEPROM, EPROM)	Memory Cassette (ROM, EEPROM, EPROM)
	Capacity	Memory Card				4 to 32 Kwords (4 to 64 Kwords for -Z)	32 to 512 Kwords (RAM: 64 to 512 Kbytes, EEPROM: 64 to 128 Kbytes, EPROM: 0.5 to 1 Mbytes	4 to 16 Kwords
	Contents	Programs, I/O memory, parameters				Programs, I/O memory, parameters	Programs, I/O memory, parameters	Programs, read-only DM, parameters
	Read/write method	Programming Device, user program (file memory instructions), or Host Link				Turning ON SR bit	Programming Device, user program (file memory instructions), Host Link, or Memory Card Writer	Turning ON AR bit
	File format	Binary				Binary	Binary	Binary
	Extended Data Memory handled as files	Yes	Yes	No	Yes	No	No	No
	Programs automatically transferred at startup	Yes				Yes	Yes	Yes
Inner Board		No			Serial Communications Board	Communications Board	No	Communications Board
Built-in serial ports		Yes (RS-232C × 1)				Yes (RS-232C × 1)	Yes (RS-232C or RS-422 × 1)	Yes (RS-232C × 1)

Item			CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Serial communications	Peripheral port	Peripheral bus	Yes				Yes	Yes	Yes
		Host Link (SYSMAC WAY)	Yes				Yes	No (Possible with connection to peripheral interface)	Yes
		Serial Gateway (conversion to CompoWay/F)	Yes (CPU Units with unit version 3.0 or later)				No	No	No
		No protocol	No				Yes	No	Yes
		NT Link	Yes				No	No	No
	CPU Unit built-in RS-232C port	Peripheral bus	Yes				No	No	No
		Host Link (SYSMAC WAY)	Yes				Yes	Yes	Yes
		Serial Gateway (conversion to CompoWay/F)	Yes (CPU Units with unit version 3.0 or later)				No	No	No
		No protocol	Yes				Yes	No	Yes
		NT Link	Yes (1:N)				Yes	No	Yes (1:1)
		Serial PLC Links	No	No	Yes	No	No	No	No
	RS-232C or RS-422/RS-485 on Communications Board	Peripheral bus	No				No	No	No
		Host Link (SYSMAC WAY)	No			Yes The WG, MP, and CR commands are not supported.	Yes The CR command is not supported.	Yes The WG and MP commands are not supported.	Yes The CR command is not supported.
		Serial Gateway (conversion to CompoWay/F, Modbus-RTU, Modbus-ASCII, or Host Link FINS)	No			Yes (Serial Communications Board/ Unit with unit version 1.2 or later)	No	No	No
		No protocol	No			Yes (Serial Communications Board/ Unit with unit version 1.2 or later)	Yes	No	Yes
		NT Link	No			Yes	Yes	No	Yes (1:1 and 1:N)
		Protocol macro	No			Yes	Yes	No	Yes
		CompoWay/F Master	No			Yes (using protocol macro or Serial Gateway)	Yes (using protocol macro)	No	Yes (using protocol macro)

Item		CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
		CJ1-H-R	CJ1-H	CJ1-M				
Interrupts	I/O interrupts	Yes (Max 2 Interrupt Input Units: 32 points, plus 4 points for built-in I/O on CJIM CPU Units) (CJ1 CPU Units do not support I/O interrupts.)			Yes (Max. 4 or 2 Interrupt Input Units: 32 points)	Yes (Max. 2 Interrupt Input Units: 16 points)	Yes (Max. 4 Interrupt Input Units: 32 points)	Yes (4 built into CPU Bus Unit)
	Scheduled interrupts	Yes				Yes	Yes	Yes
	One-shot timer interrupts	No				No	No	Yes
	Input interrupts in counter mode	No	No	Yes	No	No	No	Yes
	High-speed counter interrupts	No	No	Yes	No	No	No	Yes
	External interrupts	Yes (See note 1.)			Yes	No	No	No
	From Communications Board	No			Yes	Yes	No	No
	Power-ON interrupt	No				No	Yes	No
	Power-OFF interrupt	Yes				No	Yes	No
	Interrupt response time	0.17 ms Built-in I/O on CJ1 CPU Units: 0.12 ms			C200H Special I/O Unit: 1 ms CJ-series I/O: 0.1 ms	1 ms	---	Approx. 0.1 ms
PLC Setup Area		No user addresses (Setting is possible from CX-Programmer or Programming Console.)				Fixed DM Area allocation: DM 6600 to DM 6655, DM 6550 to DM 6559. Setting possible from Programming Console.	No user addresses (setting possible only from Programming Device, including partially from Programming Console)	Fixed DM Area allocation: DM 6600 to DM 6655. Setting possible from Programming Console.

Item			CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Initial settings	I/O	Input response time for Basic I/O Unit	Set in PLC Setup				No	No	Set in PLC Setup
		Rack first addresses	Set in I/O table from Programming Device (but order of rack numbers is fixed).				No	Set in PLC Setup (Rack No. order can be set.)	No
		First address of SYSMAC BUS Optical I/O Units by Master	No				No	Set in PLC Setup	No
		Operation for I/O verification error	No				No	Set in PLC Setup	No
	Memory	User memory protection	Set on DIP switch				Set on DIP switch	Determined by key switch setting	Set on DIP switch
		Holding areas	No				No	Set in PLC Setup	No
		Holding I/O words for fatal errors (except power failure)	No				No	Set in PLC Setup	No
		Memory saved using IOM Hold Bit when power to PLC is turned ON	Set in PLC Setup				Set in PLC Setup	Set in PLC Setup	Set in PLC Setup
		Memory saved using Forced Status Hold Bit when power to PLC is turned ON	Set in PLC Setup				Set in PLC Setup	Set in PLC Setup	Set in PLC Setup
		DIP switch status monitoring	Yes				Yes	No	Yes
	Instructions	Setting indirect DM data to BCD or binary	Direct input possible				No	Set in PLC Setup	No
		Multiple use of JMP(0) instruction	Multiple use already possible				No	Set in PLC Setup	No
		Operation for instruction errors (Continue or stop)	Set in PLC Setup				No	No	No
		Background execution	Yes				No	No	No
	File memory	Automatic transfer at startup	Determined by DIP switch setting (Automatically read from Memory Card)				Determined by DIP switch setting (Automatically read from memory cassette)	Set in PLC Setup or DIP switch setting (Automatically read from Memory Card)	Determined by DIP switch setting (Automatically read from Memory Card)
		EM file memory	Set in PLC Setup				No	No	No
	Interrupts	Interrupt response	No				Set in PLC Setup (C200H/High-speed response)	No	No
		Error detection	Set in PLC Setup				Set in PLC Setup	No	No
		Holding I/O interrupts during I/O interrupt program execution	No				No	Set in PLC Setup	No
		Power OFF interrupt enabled/disabled	Set in PLC Setup				No	Set in PLC Setup	No
		Scheduled interrupt interval setting	Set in PLC Setup (10 ms, 1.0 ms, 0.1 ms)	Set in PLC Setup (10 ms, 1.0 ms)	Set in PLC Setup (10 ms, 1.0 ms, 0.1 ms)	Set in PLC Setup (10 ms, 1.0 ms)	Set in PLC Setup (10 ms, 1 ms, 0.1 ms)	Set in PLC Setup (10 ms, 1 ms, 0.5 ms)	No

Item			CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Initial settings (contd.)	Power supply	Restart Continuation Bit Hold	No				No	Set in PLC Setup	No
		Startup mode	Set in PLC Setup				Set in PLC Setup	Set in PLC Setup	Set in PLC Setup
		Startup Condition Settings	Yes				No	No	No
		Startup trace	No				No	Set in PLC Setup	No
		Detect low battery voltage	Set in PLC Setup				Set in PLC Setup	Set in PLC Setup	Set in PLC Setup
		Momentary power interruption time	No				No	Set in PLC Setup	No
		Power OFF detection delay time	Set in PLC Setup				Set in PLC Setup (Time that operation will continue after power OFF has been detected)	No	No
		Momentary power interruption as fatal/non-fatal error	No				No	Set in PLC Setup	No
	Cycles	I/O refresh	No				Set in PLC Setup (Special I/O Units only)	Set in PLC Setup	No
		Constant cycle time	Set in PLC Setup (1 to 32,000 ms)				Set in PLC Setup(1 to 9,999 ms)	Set in PLC Setup (1 to 32,000 ms)	Set in PLC Setup(1 to 9,999 ms)
		Monitor cycle time	Set in PLC Setup (10 to 40,000 ms) (Initial setting: 1,000 ms fixed)				Set in PLC Setup (0 to 99) Unit: 1 s, 10 ms, 100 ms (Initial setting: 120 ms fixed)	Set in PLC Setup (10 to 40,000 ms) (Initial setting: 1,000 ms fixed)	Set in PLC Setup (0 to 99) Unit: 1 s, 10 ms, 100 ms (Initial setting: 120 ms fixed)
		Detect cycle time over disable	No				Set in PLC Setup	No	Set in PLC Setup
		Asynchronous instruction execution and peripheral servicing	No				No	Set in PLC Setup	No
	CPU processing mode	Parallel processing modes	Yes	Yes	No	Yes	No	No	No
		Peripheral Servicing Priority Mode	Yes				No	No	No
	Serial communications	RS-232C port communications settings	DIP switch setting for auto-detect or PLC Setup				DIP switch setting selects defaults or PLC Setup		
		Peripheral port communications settings	Set in PLC Setup				PLC Setup	Set on DIP switch.	Set in PLC Setup
		Communications Board communications settings	No				PLC Setup	No	PLC Setup
	Servicing other peripherals	Service time	Set in PLC Setup (Fixed Peripheral Servicing Time)				Set in PLC Setup (Built-in RS-232C port, Communications Board, peripheral port)	No	Set in PLC Setup (Built-in RS-232C port, Communications Board, peripheral port)
		Measure CPU Bus Unit service interval	No				No	Set in PLC Setup	No
		Stop Special I/O Unit Cyclic Refreshing	Set in PLC Setup				Set in PLC Setup	No	No
		CPU Bus link application	No				No	Set in PLC Setup	No

Item			CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Initial settings (contd.)	Programming Console	Programming Console language	Selected using Programming Console keys.				Set on DIP switch	No	Set on DIP switch
	Errors	Error Log Area	No (Fixed)				No (Fixed: DM 6001 to DM6030)	Set in PLC Setup	No (Fixed: DM 6569 to DM 6599)
		Not registering user-defined FAL errors in error log	Yes				No	No	No
	Operation	CPU Standby	No				No	Set in PLC Setup	No
Auxiliary Area	Condition Flags	ER, CY, <, >, =, Always ON/OFF Flag, etc.	Input using symbols, e.g., ER.				Yes	Yes	Yes
		Clock pulses	Input using symbols, e.g., 0.1 s.				Yes	Yes	Yes
	Servicing	CPU Service Disable Bit	No				No	Yes	No
		Codes for connected devices	No				No	Yes	No
		Peripherals processing cycle times	No				No	Yes	No
		CPU Bus Unit service interval	No				No	Yes	No
		Peripherals connected to CPU enabled/disabled	No				No	Yes	No
		Host Link/NT Link Service Disable Bit	No				No	Yes	No
		Peripheral Service Disable Bit	No				No	Yes	No
		Scheduled Refresh Disable Bit	No				No	Yes	No
		Inner Board General Purpose Monitoring Area	No			Yes	Yes	No	Yes
		Cycle time over	Yes				Yes	Yes	Yes
	Tasks	First Task Flag	Yes				No (Only First Cycle Flag)		
	Debugging	Online Editing Disabled Flag	Yes				Yes (AR)	No	No
		Online Edit Standby Flag	Yes				Yes (AR)	No	No
		Output OFF Bit	Yes				Yes	Yes	Yes
		Forced Status Hold Bit	Yes				Yes	Yes	Yes
	File memory	File Memory Instruction Flag	Yes				No	Yes	No
		EM File Memory Format Error Flag	Yes	Yes	No	Yes	No	No	No
		EM File Format Starting Bank	Yes	Yes	No	Yes	No	No	No
	Memory	DIP Switch Status Flags	Yes (pin 6)				Yes (AR, pin 6 only)	No	Yes (AR, pin 6)
		IOM Hold Bit	Yes				Yes	Yes	Yes
	Interrupts	Max. subroutine/action processing time	Yes				Yes	No	No
		Interrupt Task Error Flag	Yes				Yes	No	No

Item			CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Auxiliary Area, contd	Errors	Error log storage area/pointer	Yes				No	Yes	No
		Error codes	Yes				Yes	Yes	Yes
	Initial settings	Initializing PLC Setup	No				Yes	No	Yes
	Communications	PLC Link Operating Level Flags	Yes (PLC Link Auxiliary Area bit)				Yes (AR)	No	No
	Power supply	Power Interruption Flag	No				No	Yes	No
		Power Interruption Time	No				No	Yes	No
		Power ON Time	Yes				No	Yes	No
		Time at Power Interruption (including power OFF)	Yes				No	Yes	Yes
		Number of Momentary Power Interruptions	Yes (Number of power interruptions)				Yes (Number of power interruptions)	Yes	Yes (Number of power interruptions)
		Total Power ON Time	Yes				No	No	No

Item			CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Allocation methods	Format		Allocation is based on number of words required by Units in order of connection.			Allocation is based on number of words required by Units and vacant slots are skipped.	Fixed word allocation: Each Unit is automatically allocated one word	Allocation is based on number of words required by Units and vacant slots are skipped.	Allocation is based on number of words required by Units in order of connection.
	Group 2 High-density I/O Unit allocation		None			Same as for Basic I/O	Group-2 allocation area in IR Area (position determined by front panel switch)	None	None
	Word reservation method		Change I/O table from CX-Programmer.				Create I/O table with empty slot or change I/O table made from CX-Programmer.	Dummy I/O Unit or change I/O table from CX-Programmer.	Automatic allocation at startup.
	Special I/O Unit allocation	CIO Area	Allocation in Special I/O Unit Area according to Unit No. 10 words per Unit for total of 96 Units.				Allocation in Special I/O Unit Area (in IR Area) according to Unit No. 10 words per Unit for total of 16 Units.	Same as for Basic I/O Units; 2 or 4 words allocated in I/O Area (differs for each Unit)	Same as for Basic I/O Units; 1, 2, or 4 words allocated in I/O Area (differs for each Unit)
		DM Area	Allocation in D20000 to D29599 according to unit number, 100 words per Unit for total of 96 Units.				Allocation in DM 1000 to DM 1999, and DM 2000 to DM 2599 100 words per Unit for total of 16 Units.	None	None
	CPU Bus Unit/ CPU Bus Unit allocation	CIO Area	Allocation in CPU Bus Unit Area according to Unit No. 25 words per Unit for total of 16 Units.				None	Allocation in CPU Bus Unit Area according to Unit No. 25 words per Unit for total of 16 Units.	None
		DM Area	Allocation in D30000 to D31599 according to Unit No. 100 words per Unit for total of 16 Units.				None	Allocation in D02000 to D03599 according to Unit No. 100 words per Unit for total of 16 Units.	None

Item			CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
I/O Memory	CIO Area		Yes				Yes	Yes	Yes
	WR Area		Yes				No	No	No
	Temporary Relay Area		Yes				Yes	Yes	Yes
	Auxiliary Area		Yes				Yes	Yes	Yes
	SR Area		No				Yes	No	Yes
	Link Area		Yes (Data Link Area)				Yes	No	Yes
	C200H Special I/O Unit Area		Yes				Yes (CIO Area)	No	No
	Built-in I/O Area		No	No	Yes (In some models)	No	No	No	No
	Serial PLC Link Area		No	No	Yes	No	No	No	No
	DM Area		Yes				Yes	Yes	Yes
	Extended Data Memory (EM) Area		Yes (Addresses including bank No. can be designated) (Not supported by CJ1M CPU Unit)				Yes (Addresses can be designated for -Z, but banks cannot)	Yes (Address including bank cannot be designated; bank must be changed. EM Unit required.)	Yes (no banks)
	Timer/Counter Area		Yes				Yes	Yes	Yes
	Index Registers		Yes				No	Yes	No
	Data Registers		Yes				No	Yes	No
	Force-set/reset areas	CIO Area	Yes				Yes	Yes	Yes
		WR Area	Yes				No	No	No
		Holding Area	Yes				Yes	No	Yes
		Auxiliary Area	No				Yes	No	Yes
		SR Area	No				No	No	No
		Link Area	No				Yes	No	Yes
		Timer/ Counters	Yes (Flag)				Yes (Flag)	Yes (Flag)	Yes (Flag)
		DM Area	No				No	No	No
		EM Area	No				No	No	No
Instruction variations/ indirect addresses	Upward differentiation (executed once)		Yes (Specified by @)				Yes (Specified by @)	Yes (Specified by ↑)	Yes (Specified by @)
	Downward differentiation (executed once)		Yes (Specified by %)				No (DIFD instruction used instead)	Yes (Specified by ↓)	No (achieved by using DIFD)
	Immediate refresh		Yes (Specified by !)				No (IORF instruction used instead)	Yes (Specified by !)	No (achieved by using IORF)
	Indirect addressing for DM/ EM	BCD mode	Yes (0000 to 9999) Asterisk is used.				Yes (0 to 9999)		
		Binary mode	Yes (00000 to 32767) @ is used. 0000 to 7FFF Hex: 0000 to 31767 8000 to FFFF Hex: 00000 to 32767 in next bank				No	Yes, but only for indirect addressing using PLC memory addresses.	No
Allocation methods	Setting first word on a Rack		Yes (for all CPU Units)				No	No	No
	Setting first word for a slot		Pre-Ver.2.0 CPU Units: See note. CPU Units Ver. 2.0 or later: Yes				No	No	No

Item		CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
		CJ1-H-R	CJ1-H	CJ1-M				
Online connections via networks without creating I/O tables		With Automatic I/O Allocation at Power ON: Yes (for all CPU Units) For Manual Operation Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes			Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes	No	No	Yes, but for Controller Link only
Sending FINS messages across network levels		FINS messages can be used for remote programming/monitoring over network layers. Messages can be sent through different kinds of networks. Pre-Ver.2.0 CPU Units: Up to 3 network levels CPU Units Ver. 2.0 or later: Controller Link or Ethernet (See note 2.): Up to 8 network levels Other networks: Up to 3 network levels				No	Yes, for 3 levels	No
Online connections to PLCs via NS-series PTs		Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes				No	No	No
Protecting CPU Units from being written by FINS commands sent via networks		Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes				No	No	No
Downloading individual tasks		Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes				No	No	No
Read-protection using passwords	Entire user program	Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes				No	No	No
	Specified tasks	Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes				No	No	No
	Enabling/disabling creating file memory program files	Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes				No	No	No
	Program write-protection	Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes				No	No	No
Automatic transfer at power ON without parameter file (.STD)		Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes				No	No	No

Note 1. For CPU Units manufactured June 1, 2002 or later (lot numbers 020601□□□□ or later), up to eight slot addresses can be set.

2. When a network with 8 levels is constructed, Routing tables must be set using CX-Integrator or CX-Net in CX-Programmer version 4.0 or higher.

Instruction Comparison

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Sequence Input Instructions	LOAD/AND/OR	LD/AND/OR	Yes				Yes	Yes	Yes
	AND LOAD/OR LOAD	AND LD/OR LD	Yes				Yes	Yes	Yes
	NOT	NOT	Yes				No	Yes	No
	CONDITION ON	UP	Yes				No	Yes (*1)	No
	CONDITION OFF	DOWN	Yes				No	Yes (*1)	No
	BIT TEST	TST/TSTN	Yes (Bit position specified in binary: 0000 to 000F Hex.)				Yes (Bit position specified in BCD.) (*2)	Yes (Bit position specified in BCD.) (*1)	No
Sequence Output Instructions	OUTPUT	OUT	Yes				Yes	Yes	Yes
	TR	TR	Yes				Yes	Yes	Yes
	KEEP	KEEP	Yes				Yes	Yes	Yes
	DIFFERENTIATE UP/DOWN	DIFU/DIFD	Yes (LD↑, AND↑, OR↑) (LD↓, AND↓, OR↓)				Yes (DIFU/DIFD)	Yes (LD↑, AND↑, OR↑)/(LD↓, AND↓, OR↓)	Yes (DIFU/DIFD)
	SET and RESET	SET/RSET	Yes				Yes	Yes	Yes
	MULTIPLE BIT SET/RESET	SETA/RSTA	Yes (Beginning bit and number of bits specified in binary.)				No	(*1) (Beginning bit and number of bits specified in BCD.)	No
	SINGLE BIT SET/RESET	SET/RSTB	Yes				No	No	No
	SINGLE BIT OUTPUT	OUTB	Yes				No	No	No
Sequence Control Instructions	END/NO OPERATION	END/NOP	Yes				Yes	Yes	Yes
	INTERLOCK/INTERLOCK CLEAR	IL/ILC	Yes				Yes	Yes	Yes
	Multiple Interlock Instructions	MILH/MILR/MILC	Pre-Ver.2.0 CPU Units: No CPU Units Ver. 2.0 or later: Yes				No	No	No
	JUMP/JUMP END	JMP/JME	Yes (Jump number: 0 to 1023)				Yes (Jump number specified in BCD: 0 to 99.)	Yes (Jump number specified in BCD: 0 to 999.)	Yes (Jump number specified in BCD: 0 to 99.)
	CONDITIONAL JUMP	CJP/CJPN	Yes (Jump number: 0 to 1023)				No	Yes (Jump number specified in BCD: 0 to 999.) (*1)	No
	MULTIPLE JUMP/JUMP END	JMP0/JME0	Yes				No	No (but PLC Setup can be set to enable multiple jumps with jump number 0)	No
	FOR/NEXT LOOPS	FOR/NEXT	Yes				No	No	No
	BREAK LOOP	BREAK	Yes				No	No	No

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Timer and Counter Instructions	HUNDRED-MS TIMER	TIM (BCD)	Yes				Yes	Yes	Yes
		TIMX (binary)	Yes (*4)				No	No	No
	TEN-MS TIMER	TIMH (BCD)	Yes				Yes	Yes	Yes
		TIMHX (binary)	Yes (*4)				No	No	No
	ONE-MS TIMER	TMHH (BCD)	Yes				No	No	No
		TMHHX (binary)	Yes (*4)				No	No	No
	TENTH-MS TIMER	TIMU (BCD)	Yes	No			No	No	No
		TIMUX (BIIN)	Yes	No			No	No	No
	HUNDREDTH-MS TIMER	TMUH (BCD)	Yes	No			No	No	No
		TMUHX (BIN)	Yes	No			No	No	No
	ACCUMULATIVE TIMER	TTIM (BCD)	Yes				Yes	Yes	Yes
		TTIMX (binary)	Yes (*4)				No	No	No
	LONG TIMER	TIML (BCD)	Yes				No	Yes	No
		TIMLX (binary)	Yes (*4)				No	No	No
	MULTI-OUTPUT TIMER	MTIM (BCD)	Yes				No	Yes	No
		MTIMX (binary)	Yes (*4)				No	No	No
	COUNTER	CNT (BCD)	Yes				Yes	Yes	Yes
		CNTX (binary)	Yes (*4)				No	No	No
	REVERSIBLE COUNTER	CNTR (BCD)	Yes				Yes	Yes	Yes
		CNTRX (binary)	Yes (*4)				No	No	No
	RESET TIMER/ COUNTER	CNR (BCD)	Yes (Only resets timer or counter.)				No	Yes (Also clears specified range in CIO area to zero.)	No
		CNRX (binary)	Yes (*4)				No	No	No

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Comparison Instructions	Symbol comparison	=, <, etc.	Yes (All are supported for LD, OR, and AND)				Yes (*2) (Supported for AND only)	Yes (*1) (Supported for AND only)	No
	Data Comparison Instructions	=DT, <DT, etc.	Yes (*6)				No	No	No
	COMPARE/ DOUBLE COMPARE	CMP/ CMPL	Yes				Yes	Yes (*3)	Yes
	SIGNED BINARY COMPARE/ DOUBLE SIGNED BINARY COMPARE	CPS/ CPSL	Yes				Yes	Yes (*1)	Yes
	BLOCK COMPARE	BCMP	Yes				Yes	Yes	Yes
	EXTENDED BLOCK COMPARE	BCMP2	Yes (*6)				No	No	No
	TABLE COMPARE	TCMP	Yes				Yes	Yes	Yes
	MULTIPLE COMPARE	MCMP	Yes				Yes	Yes	Yes
	EQUALS	EQU	No				No	Yes	No
	AREA RANGE COMPARE	ZCP/ ZCPL	Yes				Yes	No	No (achieved using comparison instructions)

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Data Movement Instruction	MOVE	MOV	Yes				Yes	Yes	Yes
	DOUBLE MOVE	MOVL	Yes				No	Yes	No
	MOVE NOT	MVN	Yes				Yes	Yes	Yes
	DOUBLE MOVE	MVNL	Yes				No	Yes	No
	DATA EXCHANGE	XCHG	Yes				Yes	Yes	Yes
	DOUBLE DATA EXCHANGE	XCGL	Yes				No	Yes	No
	MOVE QUICK	MOVQ	No (MOV(021) can be used instead.)				No	Yes	No
	BLOCK TRANSFER	XFER	Yes (Number of words to be transferred specified in binary: 0 to 65535.)				Yes (Number of words to be transferred specified in BCD: 0 to 6144.)	Yes (Number of words to be transferred specified in BCD: 0 to 9999.)	Yes (Number of words to be transferred specified in BCD: 0 to 9999.)
		XFERC	Yes (*7) (Number of words to be transferred specified in BCD: 0 to 9999.)						
	BLOCK SET	BSET	Yes				Yes	Yes	Yes
	MOVE BIT	MOVB	Yes (Source bit position and destination bit position specified in binary.)				Yes (Source bit position and destination bit position specified in BCD.)		
		MOVBC	Yes (*7) (Source bit position and destination bit position specified in BCD.)						
	MULTIPLE BIT TRANSFER	XFRB	Yes				Yes	Yes (*1)	Yes
	MOVE DIGIT	MOVD	Yes				Yes	Yes	Yes
	SINGLE WORD DISTRIBUTE	DIST	Yes (Offset value specified in binary: 0 to 65535.)				Yes (Offset value specified in BCD: 0 to 8999.)	Yes (Offset value specified in BCD: 0 to 9999.)	Yes (Offset value specified in BCD: 0 to 8999.)
		DISTC	Yes (*7) (Offset value specified in BCD: 0 to 8999.)						
	DATA COLLECT	COLL	Yes (Offset value specified in binary: 0 to 65535.)				Yes (Offset value specified in BCD: 0 to 7999.)	Yes (Offset value specified in BCD: 0 to 9999.)	Yes (Offset value specified in BCD: 0 to 9999.)
		COLLC	Yes (*7) (Offset value specified in BCD: 0 to 9999.)						
	EM BLOCK TRANSFER BETWEEN BANKS	BXFR	No (XFER can be used instead.)				No	Yes (*1)	No
	EM BLOCK TRANSFER	XFR2	No				Yes	No	No
	EM BANK TRANSFER	BXF2	No				Yes	No	No
	MOVE TO REGISTER	MOVR	Yes (No address is specified for indirect DM/EM.)				No	Yes (Address is specified for indirect EM/DM.)	No
	MOVE TIMER/ COUNTER PV TO REGISTER	MOVRW	Yes				No	No (Possible for Completion Flags only using MOVR)	No

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Data Shift Instructions	SHIFT REGISTER	SFT	Yes				Yes	Yes	Yes
	REVERSIBLE SHIFT REGISTER	SFTR	Yes				Yes	Yes	Yes
	ASYNCHRONOUS SHIFT REGISTER	ASFT	Yes				Yes	Yes	Yes
	WORD SHIFT	WSFT	Yes (Same as CV: 3 operands)				Yes	Yes	Yes
	ARITHMETIC SHIFT LEFT/ ARITHMETIC SHIFT RIGHT	ASL/ ASR	Yes				Yes	Yes	Yes
	ROTATE LEFT/ ROTATE RIGHT	ROL/ ROR	Yes				Yes	Yes	Yes
	ONE DIGIT SHIFT LEFT/ ONE DIGIT SHIFT RIGHT	SLD/ SRD	Yes				Yes	Yes	Yes
	SHIFT N-BIT DATA LEFT/ SHIFT N-BIT DATA RIGHT	NSFR/ NSFL	Yes (Shift data and beginning bit specified in binary.)				No	Yes (Shift data and beginning bit specified in BCD.) (*1)	No
	SHIFT N-BITS LEFT/SHIFT N-BITS RIGHT/ DOUBLE SHIFT N-BITS LEFT/ DOUBLE SHIFT NITS RIGHT	NASL/ NASR, NSLL/ NSRL	Yes (Number of bits to be shifted specified in binary.)				No	Yes (Number of bits to be shifted specified in BCD.) (*1)	No
	DOUBLE SHIFT LEFT/DOUBLE SHIFT RIGHT	ASLL/ ASRL	Yes				No	Yes	No
	DOUBLE ROTATE LEFT/ DOUBLE ROTATE RIGHT	ROLL/ RORL	Yes				No	Yes	No
	ROTATE LEFT WITHOUT CARRY/ROTATE RIGHT WITHOUT CARRY/ DOUBLE ROTATE LEFT WITHOUT CARRY/DOUBLE ROTATE RIGHT WITHOUT CARRY	RLNC/ RRNC, RLNL/ RRNL	Yes				No	Yes (*1)	No
Increment and Decrement Instructions	INCREMENT BCD/DECREMENT BCD	++B/– –B	Yes				Yes (INC/DEC)		
	DOUBLE INCREMENT BCD/ DOUBLE DECREMENT BCD	++BL/– –BL	Yes				No	Yes (INCL/ DECL)	No
	INCREMENT BINARY/ DECREMENT BINARY	++/– –	Yes (CY turns ON for carry or borrow)				No	Yes (INCB/ DECB)	No
	DOUBLE INCREMENT BINARY/ DOUBLE DECREMENT BINARY	++L/– –L	Yes (CY turns ON for carry or borrow)				No	Yes (INBL/ DCBL)	No
Math Instructions			Yes				Yes	Yes	Yes

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Conversion Instructions	BCD-TO-BINARY/DOUBLE BCD-TO-DOUBLE BINARY	BIN/ BINL	Yes				Yes	Yes	Yes
	BINARY-TO-BCD/DOUBLE BINARY-TO-DOUBLE BCD	BCD/ BCDL	Yes				Yes	Yes	Yes
	2'S COMPLEMENT/ DOUBLE 2'S COMPLEMENT	NEG/ NEGL	Yes (Same as CV but UP does not turn ON for 8000 Hex at source)				Yes	Yes	Yes
	16-BIT TO 32-BIT SIGNED BINARY	SIGN	Yes				No	Yes	No
	DATA DECODER	MLPX	Yes				Yes	Yes	Yes
	DATA ENCODER	DMPX	Yes (Same as CVM1-V2: Can specify rightmost bit for ON.)				Yes (Leftmost bit only for ON.)	Yes (CVM1-V2: Can specify rightmost bit for ON.)	Yes (Leftmost bit only for ON.)
	ASCII CONVERT	ASC	Yes				Yes	Yes	Yes
	ASCII TO HEX	HEX	Yes				Yes	Yes (*1)	Yes
	COLUMN TO LINE/LINE TO COLUMN	LINE/ COLM	Yes (Bit position specified in binary.)				Yes (Bit position specified in BCD)		
	SIGNED BCD-TO-BINARY/ DOUBLE SIGNED BCD-TO-BINARY	BINS/ BISL	Yes				No	Yes (*1)	No
	SIGNED BINARY-TO-BCD/DOUBLE SIGNED BINARY-TO-BCD	BCDS/ BDSL	Yes				No	Yes (*1)	No
	GRAY CODE CONVERSION	GRY	Yes				No	No	No
	Numeral to ASCII Conversion Instructions	STR4/ STR8/ STR16	Yes (*8)				Yes	Yes	Yes
	ASCII to Numeral Conversion Instructions	NUM4/ NUM8/ NUM16	Yes (*8)				Yes	Yes	Yes
Logic Instructions	LOGICAL AND/ LOGICAL OR/ EXCLUSIVE OR/ EXCLUSIVE NOR	ANDW, ORW, XORW, XNRW	Yes				Yes	Yes	Yes
	DOUBLE LOGICAL AND/DOUBLE LOGICAL OR/DOUBLE EXCLUSIVE OR/ DOUBLE EXCLUSIVE NOR	ANDL, ORWL, XORL, XNRL	Yes				No	Yes	No
	COMPLEMENT/ DOUBLE COMPLEMENT	COM/ COML	Yes				Yes (COM only)	Yes	Yes (COM only)

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Special Math Instructions	BCD SQUARE ROOT	ROOT	Yes				Yes	Yes	Yes
	BINARY ROOT	ROTB	Yes				No	Yes (*1)	No
	ARITHMETIC PROCESS	APR	Yes				Yes	Yes	Yes
	FLOATING POINT DIVIDE	FDIV	Yes				Yes	Yes	No
	BIT COUNTER	BCNT	Yes (Number of words to count and count results in binary: 0 to FFFF Hex)				Yes (Number of words to count and count results in BCD: 1 to 6656)	Yes (Number of words to count and count results in BCD: 0 to 9999, but error for 0)	Yes (Number of words to count and count results in BCD: 1 to 6656)
		BCNTC	Yes (*7) (Number of words to count and count results in BCD: 0 to 9999)						
Floating-point Math Instructions	FLOATING TO 16-BIT/32-BIT BIN, 16-BIT/32-BIT BIN TO FLOATING	FIX/ FIXL, FLT/ FLTL	Yes				No	Yes (*1)	Yes
	FLOATING-POINT ADD/ FLOATING-POINT SUBTRACT/FLOATING-POINT MULTIPLY/ FLOATING-POINT DIVIDE	+F, –F, *F, /F	Yes				No	Yes (*1)	Yes
	DEGREES TO RADIANS/RADIANS TO DEGREES	RAD, DEG	Yes				No	Yes (*1)	Yes
	HIGH-SPEED SINE/COSINE/ TANGENT	SINQ, COSQ, TANQ	Yes	No			No	No	No
	SINE/COSINE/ TANGENT/ARC SINE/ARC TANGENT	SIN, COS, TAN, ASIN, ACOS, ATAN	Yes				No	Yes (*1)	Yes
	SQUARE ROOT	SQRT	Yes				No	Yes (*1)	Yes
	EXPONENT	EXP	Yes				No	Yes (*1)	Yes
	LOGARITHM	LOG	Yes				No	Yes (*1)	Yes
	EXPONENTIAL POWER	PWR	Yes				No	No	No
	Floating-point Decimal Comparison	Examples: =F, <>F	Yes				No	No	No
	Floating-point Decimal to Text String	FSTR, FVAL	Yes				No	No	No
	MOVE FLOATING-POINT	MOVF	Yes	No			No		
Double-precision Floating-point Conversion and Calculation Instructions	Same as Single-precision Floating-point Conversion and Calculation Instructions, above	Example: FIXD	Yes				No	No	No

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Table Data Processing Instructions	SET STACK	SSET	Yes (Four words of stack control information. Number of words specified in binary: 5 to 65535)				No	Yes (Four words of stack control information. Number of words specified in BCD: 3 to 9999)	No
	PUSH ONTO STACK:	PUSH	Yes				No	Yes	No
	FIRST IN FIRST OUT	FIFO	Yes				No	Yes	No
	LAST IN FIRST OUT	LIFO	Yes				No	Yes	No
	FIND MAXIMUM/ FIND MINIMUM	MAX, MIN	Yes (Two words in control data field. Table length specified in binary: 1 to FFFF)				Yes (One word in control data field. Table length specified in BCD: 1 to 999)		
	DATA SEARCH	SRCH	Yes (Table length specified in binary: 1 to FFFF. PLC memory address output to IR0. Number of matches can be output to DR0.)				Yes (Table length specified in BCD: 1 to 6556. PLC memory address output to C+1. Number of matches cannot be output to DR0.)	Yes (Table length specified in BCD: 1 to 9999. PLC memory address output to IR0. Number of matches cannot be output to DR0.)	Yes (Table length specified in BCD: 1 to 6556. PLC memory address output to C+1. Number of matches cannot be output to DR0.)
	FRAME CHECKSUM	FCS	Yes				Yes	No	Yes
	SUM	SUM	Yes (Same as C200HX/HG/HE: Sum possible for bytes as well as words.)				Yes (Sum possible for bytes as well as words.)	Yes (Sum possible for words only.)	Yes (Sum possible for bytes as well as words.)
	SWAP BYTES	SWAP	Yes (Can be used for data communications and other applications.)				No	No	No
	DIMENSION RECORD TABLE:	DIM	Yes				No	No	No
	SET RECORD LOCATION	SETR	Yes				No	No	No
	GET RECORD LOCATION	GETR	Yes				No	No	No
Data Control Instructions	SCALING	SCL	Yes				Yes	No	Yes
	SCALING 2	SCL2	Yes				No	No	Yes
	SCALING 3	SCL3	Yes				No	No	Yes
	PID CONTROL	PID	Yes (Output can be switched between 0% and 50% when PV = SV. PID and sampling period specified in binary.)				Yes (PID and sampling period specified in BCD)	Yes (PID and sampling period specified in BCD) (*1)	Yes (PID and sampling period specified in BCD)
	PID CONTROL WITH AUTOTUNIG	PIDAT	Yes				No	No	No
	LIMIT CONTROL	LMT	Yes				No	Yes (*1)	No
	DEAD BAND CONTROL	BAND	Yes				No	Yes (*1)	No
	DEAD ZONE CONTROL	ZONE	Yes				No	Yes (*1)	No
	TIME-PROPORTIONAL OUTPUT	TPO	Yes (*6)				No	No	No
	AVERAGE	AVG	Yes (Number of scans specified in binary)				Yes (Number of scans specified in BCD)	No	Yes (Number of scans specified in BCD)

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Subroutines Instructions	SUBROUTINE CALL/SUBROUTINE ENTRY/ SUBROUTINE RETURN	SBS, SBN, RET	Yes (Subroutine number: 0 to 1023)				Yes (Subroutine number specified in BCD: 0 to 255)	Yes (Subroutine number specified in BCD: 0 to 999)	Yes (Subroutine number specified in BCD: 0 to 255)
	MACRO	MCRO	Yes (Subroutine number: 0 to 1023)				Yes (Subroutine number specified in BCD: 0 to 255)	Yes (Subroutine number specified in BCD: 0 to 999) (*1)	Yes (Subroutine number specified in BCD: 0 to 255)
	Global Subroutine Instructions	GSBS, GSBN, RET	Yes				No	No	No
Interrupt Control Instructions	SET INTERRUPT MASK	MSKS	Yes				No (All interrupt processing performed with INT)	Yes	No (All interrupt processing performed with INT)
	CLEAR INTERRUPT	CLI	Yes				No	Yes	No
	READ INTERRUPT MASK:	MSKR	Yes				No	Yes	No
	DISABLE INTERRUPTS	DI	Yes				No	No	No
	ENABLE INTERRUPTS	EI	Yes				No	No	No
	ENABLE TIMER	STIM	No				No	No	Yes
High-speed Counter/ Pulse Output Instructions	MODE CONTROL	INI	Yes (*5)				No	No	Yes
	PRESENT VALUE READ	PRV	Yes (*5)				No	No	Yes
	COUNTER FREQUENCY CONVERT	PRV2	Yes (*9)				No	No	No
	SET COMPARISON TABLE	CTBL	Yes (*5)				No	No	Yes
	SET PULSES	PULS	Yes (*5)				No	No	Yes
	SET FREQUENCY	SPED	Yes (*5)				No	No	Yes
	ACCELERATION CONTROL	ACC	Yes (*5)				No	No	Yes
	POSITION CONTROL	PLS2	Yes (*5)				No	No	Yes
	ORIGIN SEARCH	ORG	Yes (*5)				No	No	Yes
	PWM OUTPUT	PWM	Yes (*5)				No	No	Yes
Step Instructions	STEP DEFINE and STEP START	STEP/ SNXT	Yes				Yes	Yes	Yes

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Basic I/O Unit Instructions	I/O REFRESH	IORF	Yes (Used for C200H Group-2 High-density I/O Units and Special I/O Units as well. Includes functionality of GROUP-2 HIGH-DENSITY I/O REFRESH (MPRF))				Yes (Used for C200H Group-2 High-density I/O Units and Special I/O Units as well.)	Yes	Yes
	7-SEGMENT DECODER	SDEC	Yes				Yes	Yes	Yes
	DIGITAL SWITCH INPUT	DSW	Yes (*6)				Yes	No	Yes
	TEN KEY INPUT	TKY	Yes (*6)				Yes	No	Yes
	HEXADECIMAL KEY INPUT	HKY	Yes (*6)				Yes	No	Yes
	MATRIX INPUT	MTR	Yes (*6)				Yes	No	No
	7-SEGMENT DISPLAY OUTPUT	7SEG	Yes (*6)				Yes	No	Yes
	GROUP-2 HIGH-DENSITY I/O REFRESH	MPRF	No				Yes	No	No
	TEN KEY INPUT	TKY	No				Yes	No	Yes
	HEXADECIMAL KEY INPUT	HKY	No				Yes	No	Yes
	DIGITAL SWITCH INPUT	DSW	No				Yes	No	Yes
	MATRIX INPUT	MTR	No				Yes	No	No
	7-SEGMENT DISPLAY OUTPUT	7SEG	No				Yes	No	Yes
Special I/O Unit Instructions	SPECIAL I/O UNIT I/O REFRESH	FIORF	Yes	No			No	No	No
	SPECIAL I/O UNIT READ and SPECIAL I/0 UNIT WRITE (I/O READ and I/O WRITE)	IORD/IOWR (READ/WRIT)	IORD/IOWR (Up to 96 Units. Will not be used to send FINS commands any more.				IORD/IOWR	READ/WRIT	No
	I/O READ 2 and I/O WRITE 2	RD2/WR2	No				No	Yes (*1)	No
Text String Processing Instructions	MOV STRING	MOV$	Yes				No	No	No
	CONCATENATE STRING	+$	Yes				No	No	No
	GET STRING LEFT	LEFT$	Yes				No	No	No
	GET STRING RIGHT	RGHT$	Yes				No	No	No
	GET STRING MIDDLE	MID$	Yes				No	No	No
	FIND IN STRING	FIND$	Yes				No	No	No
	STRING LENGTH	LEN$	Yes				No	No	No
	REPLACE IN STRING	RPLC$	Yes				No	No	No
	DELETE STRING	DEL$	Yes				No	No	No
	EXCHANGE STRING	XCHG$	Yes				No	No	No
	CLEAR STRING:	CLR$	Yes				No	No	No
	INSERT INTO STRING	INS$	Yes				No	No	No

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Serial Communications Instructions	RECEIVE	RXD	Yes (Number of stored bytes specified in binary) (Used only for CPU Unit's RS-232C port and Serial Communications Boards with unit version 1.2 or later. Cannot be used for CPU Unit's peripheral port.)				Yes (Number of stored bytes specified in BCD) (Used for peripheral port, RS-232C port or Communications Board in CPU Unit.)	No	Yes (Number of stored bytes specified in BCD) (Used for peripheral port, RS-232C port or Communications Board in CPU Unit.)
	RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU	Yes (Number of stored bytes specified in binary) (Used for Serial Communications Unit with unit version 1.2 or later.)				No	No	No
	TRANSMIT	TXD	Yes (Number of stored bytes specified in binary) (Used only for CPU Unit's RS-232C port and Serial Communications Boards with unit version 1.2 or later. Cannot be used for CPU Unit's peripheral port) (Unsolicited communications are not possible using the Host Link EX command.)				Yes (Number of stored bytes specified in BCD) (Used for peripheral port, RS-232C port or Communications Board in CPU Unit.) (Unsolicited communications are possible using Host Link EX command.)	No	Yes (Number of stored bytes specified in BCD) (Used for peripheral port, RS-232C port or Communications Board in CPU Unit.) (Unsolicited communications are possible using Host Link EX command.)
	TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU	Yes (Number of stored bytes specified in binary) (Used for Serial Communications Unit with unit version 1.2 or later.)				No	No	No
	CHANGE SERIAL PORT SETUP	STUP	Yes (10 words set) Can be used for Serial Communications Unit.				Yes (5 words set)	No	Yes (5 words set)
	PROTOCOL MACRO	PMCR	Yes (Sequence number specified in binary. Four operands. Can specify destination unit address and Serial Port number.)				Yes (Sequence number specified in BCD. Three operands.)	No	Yes (Sequence number specified in BCD. Three operands.)
	PCMCIA CARD MACRO	CMCR	No				Yes	No	No
Network Instructions	NETWORK SEND/NETWORK RECEIVE	SEND/RECV	Yes (Can be used for host computer via Host Link connections. Cannot be used for Serial Communications Units, CPU Unit's RS-232C port, or Inner Board.)				Yes (Cannot be used for host computer via Host Link connections.)	Yes (Can be used for host computer via Host Link connections.)	Yes (Cannot be used for host computer via Host Link connections.)
	DELIVER COMMAND	CMND	Yes (Used for host computer via Host Link connections. Cannot be used for Serial Communications Units, CPU Unit's RS-232C port, or Inner Board.)				No	Yes (Can be used for host computer via Host Link connections.)	Yes (Cannot be used for host computer via Host Link connections.)
	EXPLICIT MESSAGE SEND	EXPLT	Yes (*6)				No	No	No
	EXPLICIT GET ATTRIBUTE	EGATR	Yes (*6)				No	No	No
	EXPLICIT SET ATTRIBUTE	ESATR	Yes (*6)				No	No	No
	EXPLICIT WORD READ	ECHRD	Yes (*6)				No	No	No
	EXPLICIT WORD WRITE	ECHWR	Yes (*6)				No	No	No

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
File Memory Instructions	READ DATA FILE/WRITE DATA FILE	FREAD/FWRIT	Yes				No	Yes (FILR/FILW)	No
	WRITE TEXT FILE	TWRIT	Yes (*6)				No	No	No
	READ PROGRAM FILE	FILP	No				No	Yes	No
	CHANGE STEP PROGRAM	FLSP	No				No	Yes	No
Display Instructions	DISPLAY MESSAGE	MSG	Yes (Messages ended by NUL)				Yes (Messages ended by CR)		
	DISPLAY LONG MESSAGE	LMSG	No				Yes (Messages ended by CR)	No	No
	I/O DISPLAY	IODP	No				No	Yes	No
	TERMINAL MODE	TERM	No				Yes	No	No
Clock Instructions	CALENDAR ADD	CADD	Yes				No	Yes	No
	CALENDAR SUBTRACT	CSUB	Yes				No	Yes	No
	HOURS TO SECONDS	SEC	Yes				Yes	Yes	Yes
	SECONDS TO HOURS	HMS	Yes				Yes	Yes	Yes
	CLOCK ADJUSTMENT	DATE	Yes				No	Yes (*1)	No
Debugging Instructions	TRACE MEMORY SAMPLING	TRSM	Yes				Yes	Yes	Yes
	MARK TRACE	MARK	No				No	Yes (Mark number specified in BCD)	No
Failure Diagnosis Instructions	FAILURE ALARM/SEVERE FAILURE ALARM	FAL/FALS	Yes (Messages ended by NUL, text strings stored in order of leftmost to rightmost byte and then rightmost to leftmost word. FAL number specified in binary.)				Yes (Messages ended by CR, text strings stored in order of leftmost to rightmost byte and then rightmost to leftmost word. FAL number specified in BCD.)	Yes (Messages ended by CR, text strings stored in order of leftmost to rightmost byte and then rightmost to leftmost word. FAL number specified in BCD.)	Yes (Messages ended by CR, text strings stored in order of leftmost to rightmost byte and then rightmost to leftmost word. FAL number specified in BCD.)
	FAILURE POINT DETECTION	FPD	Yes (Messages ended by NUL, text strings stored in order of leftmost to rightmost byte and then rightmost to leftmost word. FAL number specified in binary.)				Yes (Messages ended by CR, text strings stored in order of leftmost to rightmost byte and then rightmost to leftmost word. FAL number specified in BCD.)	Yes (Messages ended by CR, text strings stored in order of leftmost to rightmost byte and then rightmost to leftmost word. FAL number specified in BCD.) (*1)	Yes (Messages ended by CR, text strings stored in order of leftmost to rightmost byte and then rightmost to leftmost word. FAL number specified in BCD.)

Item		Mnemonic	CJ1-H/CJ1M Series			CS1-H Series	C200HX/HG/HE	CVM1/CV Series	CQM1H
			CJ1-H-R	CJ1-H	CJ1-M				
Other Instructions	SET CARRY/ CLEAR CARRY	STC/ CLC	Yes				Yes	Yes	Yes
	LOAD FLAGS/ SAVE FLAGS	CCL, CCS	Yes				No	Yes	No
	EXTEND MAXIMUM CYCLE TIME	WDT	Yes				Yes	Yes (*1)	Yes
	CYCLE TIME	SCAN	No				Yes	No	No
	LOAD REGISTER/SAVE REGISTER	REGL, REGS	No				No	Yes	No
	SELECT EM BANK:	EMBC	Yes				Yes	Yes	No
	EXPANSION DM READ	XDMR	No				Yes	No	No
	INDIRECT EM ADDRESSING	IEMS	No				Yes	No	No
	ENABLE ACCESS/DISABLE ACCESS	IOSP, IORS	No				No	Yes	No
	CV-CS Address Conversion Instructions	FRMCV TOCV	Yes				No	No	No
Block Programming Instructions		BPRG/ BEND, IF/ ELSE/ IEND, WAIT, EXIT, LOOP/ LEND, BPPS/ BPRS, TIMW, CNTW, TMHW	Yes				No	Yes (*1)	No
Task Control Instructions	TASK ON/TASK OFF	TKON/ TKOF	Yes				No	No	No

Note *1: Supported only by CVM1 (V2).
*2: Supported only by CPU□□-Z models.
*3: Continuation on same program run supported by CV1M version 2.
*4: Except for CS1G/H-CPU□□(-V1) and CJ1□-CPU□□ CPU Units.
*5: CJ1M CPU Units with built-in I/O only. Some operands differ from those used by the CQM1H.
*6: Supported by CPU Units with unit version 2.0 or later.
*7: Supported by CPU Units with unit version 3.0 or later.
*8: Supported by CPU Units with unit version 4.0 or later.
*9: Supported by CJ1M CPU Units with unit version 2.0 or later.

Appendix B
Changes from Previous Host Link Systems

There are differences between Host Link Systems created using the CS/CJ-series Serial Communications Boards (CS Series only) and Unit in comparison to Host Link Systems created with Host Link Units and CPU Units in other PLC product series. These differences are described in this sections.

RS-232C Ports

Take the following differences into consideration when changing from an existing Host Link System to one using an RS-232C port on a CS/CJ-series CPU Unit, Serial Communications Boards (CS Series only), or Serial Communications Unit (CS1H/G-CPU□□ RS-232C port, CS1W-SCU21 ports, CS1W-SCB21 ports, CS1W-SCB41 port 1, or CJ1W-SCU41 port 2).

Previous products	Model number	Changes required for CS/CJ-series product	
		Wiring	Other
C-series Host Link Units	3G2A5-LK201-E C500-LK203 3G2A6-LK201-E	The connector has been changed from a 25-pin to a 9-pin connector. The CS/CJ-series products do not support the ST1, ST2, and RT signals and wiring them is not required.	**The following changes are necessary for systems that sync with ST1, ST2, and RT.** Synchronized transfers will no longer be possible. Full-duplex transmissions will be possible with the CS/CJ-series product, but the host computer's communications program, hardware, or both will need to be altered. **The following changes are necessary for systems that did not sync with ST1, ST2, and RT.** It may be possible to use the host computer programs without alteration as long as the same communications settings (e.g., baud rate) are used. It may be necessary, however, to change programs to allow for different text lengths in frames or different CS/CJ command specifications. (See note.)
	C200H-LK201	The connector has been changed from a 25-pin to a 9-pin connector.	It may be possible to use the host computer programs without alteration as long as the same communications settings (e.g., baud rate) are used. It may be necessary, however, to change programs to allow for different text lengths in frames or different CS/CJ command specifications. (See note.)
C-series CPU Units	SRM1 CPM1 CPM1A CQM1-CPU□□-E C200HS-CPU□□-E C200HX/HG/HE-CPU□□-E C200HW-COM□□-E	No changes have been made in wiring.	It may be possible to use the host computer programs without alteration as long as the same communications settings (e.g., baud rate) are used. It may be necessary, however, to change programs to allow for different CS/CJ command specifications.

Previous products	Model number	Changes required for CS/CJ-series product	
		Wiring	Other
CVM1 or CV-series CPU Units	CVM1/CV-CPU□□-E	No changes have been made in wiring.	It may be possible to use the host computer programs without alteration as long as the same communications settings (e.g., baud rate) are used. It may be necessary, however, to change programs to allow for different CS/CJ command specifications.
CVM1 or CV-series Host Link Unit	CV500-LK201	Port 1: The connector has been changed from a 25-pin to a 9-pin connector. Port 2 set for RS-232C: The SG signal has been changed from pin 7 to pin 9.	**The following changes are necessary for half-duplex transmissions that use CD.** Check the system for timing problems when using SEND, RECV, or CMND to initiate communications from the PLC or timing problems in sending commands from the host computer. If necessary, switch to full-duplex transmissions. **The following changes are necessary for full-duplex transmissions that do not use CD.** Half-duplex It may be possible to use the host computer programs without alteration as long as the same communications settings (e.g., baud rate) are used. It may be necessary, however, to change programs to allow for different CS/CJ command specifications.

Note The number of words that can be read and written per frame (i.e., the text lengths) when using C-mode commands is different for C-series Host Link Units and CS/CJ-series Serial Communications Boards/ Units. A host computer program previously used for C-series Host Link Units may not function correctly if used for CS/CJ-series PLCs. Check the host computer program before using it and make any corrections required to handle different frame text lengths. Refer to the *CS/CJ-series Communications Commands Reference Manual* (W342) for details.

RS-422A/485 Ports

Take the following differences into consideration when changing from an existing Host Link System to one using an RS-422A/485 port on a CS-series Serial Communications Board (CS1W-SCB41 port 2) or a CJ-series Serial Communications Unit (CJ1W-SCU41 port 1).

Previous products	Model number	Changes required for CS/CJ-series product	
		Wiring	Other
C-series Host Link Units	3G2A5-LK201-E C200H-LK202 3G2A6-LK202-E	Wiring pins have been changed as shown below. SDA: Pin 9 to pin 1 SDB: Pin 5 to pin 2 RDA: Pin 6 to pin 6 RDB: Pin 1 to pin 8 SG: Pin 3 to Not connected FG: Pin 7 to pin Connector hood	It may be possible to use the host computer programs without alteration as long as the same communications settings (e.g., baud rate) are used. It may be necessary, however, to change programs to allow for different text lengths in frames or different CS/CJ command specifications. (See note.)
C200HX/HG/HE Communications Board	C200HW-COM□□-E	No changes have been made in wiring.	It may be possible to use the host computer programs without alteration as long as the same communications settings (e.g., baud rate) are used. It may be necessary, however, to change programs to allow for different CS/CJ command specifications.

Previous products	Model number	Changes required for CS/CJ-series product	
		Wiring	Other
CVM1 or CV-series CPU Units	CVM1/CV-CPU□□-E	No changes have been made in wiring.	It may be possible to use the host computer programs without alteration as long as the same communications settings (e.g., baud rate) are used. It may be necessary, however, to change programs to allow for different CS/CJ command specifications.
CVM1 or CV-series Host Link Unit	CV500-LK201		

Note The number of words that can be read and written per frame (i.e., the text lengths) when using C-mode commands is different for C-series Host Link Units and CS/CJ-series Serial Communications Boards/Units. A host computer program previously used for C-series Host Link Units may not function correctly if used for CS/CJ-series PLCs. Check the host computer program before using it and make any corrections required to handle different frame text lengths. Refer to the *CS/CJ-series Communications Commands Reference Manual* (W342) for details.

Index

J-L

M

N

O

P

R

S

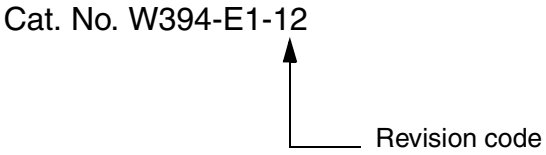
T

U

V–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	April 2001	Original production
02	October 2001	Added information on high-speed CS-series and high-speed CJ-series CPU Units (CS1G/H-CPU□□H and CJ1G/H-CPU□□H)) throughout the manual.
03	July 2002	Information on CJ1M CPU Units added throughout. PC changed to PLC for "Programmable Controller." Other changes are as follows: **Pages xvi and xviii:** Caution added. **Page xix:** Item 2 at bottom of page changed. **Page 28:** Description for text string changed. **Page 167:** Programming example changed. **Pages 168, 169, 265, and 266:** Information added on DC power supplies. **Page 179:** Precautions added on Memory Cards. **Page 229:** Illustration changed. **Page 262:** Information added on timer/counter refresh method. **Page 273:** Precaution added on DeviceNet. **Page 301:** Units corrected in processing speeds. **Page 304:** Interrupt response time corrected. **Page 320:** CJ1 support for IOSP/IORS changed.
04	September 2002	Information on CJ1D CPU Units added throughout. Other changes are as follows: **Page xv:** Caution added on backup function. **Page xvi:** First caution rewritten and CPU Unit types in startup operating mode caution corrected. **Page xviii:** Caution added on RS-232C port toward middle of page. **Page 6:** Information added on CX-Programmer versions. **Page 184:** Memory Card information in table corrected. **Page 274:** Range for setting maximum unit number changed toward top of page. **Page 294:** Note changed to Caution and rewritten. **Page 303:** Note 3 rewritten. **Page 304:** Addition made to middle table.
05	April 2003	**Page 44:** First basic ladder program concept rewritten. **Page 45:** Second restriction information changed. **Page 46:** Fourth restriction information changed. Sixth restriction information removed. **Page 54:** Information on rungs requiring caution or rewriting changed. **Pages 157 and 158:** Changes made to table of differences between extra and normal cyclic tasks. **Page 220:** Note 5 added. **Page 226:** Information changed regarding supported units. **Page 233:** Information changed regarding power interruptions while accessing file memory. **Page 264:** Note added. **Page 273:** Note added. **Page 276:** Information changed in tables. Note added under first table and note changed under second table. **Page 310:** Information under first table regarding CX-Programmer changed. **Page 314:** Information added to second note. **Page 326:** Information added to step 1 in data trace procedure.

Revision code	Date	Revised content
06	December 2003	Information added on new functions supported by new unit versions of the CPU Units (too numerous to list). **Pages xi to xx:** PLP information updated. **Page 72:** Notes added at top of table and AND NOT and OR NOT instructions added. **Pages 160, 201, 202, 228, 293, and 320:** Notes added. **Page 189:** Information on CX-Programmer version 1.0 or higher removed. **Page 191:** Information added above and in table and new table added. **Pages 197, 199, 216, to 218, 230, and 231:** Information added for replacement without parameter area file. **Page 226:** Table expanded. **Page 294:** New section 6-6-8 added. **Pages 337 to 339, 342, 344, 345, and 347:** Rows added to table. **Page 339:** Information added for BCMP2.
07	July 2004	Changes were made throughout the manual related to information on new functions supported by the upgrade from unit version 2.0 to unit version 3.0 of the CS/CJ-series CPU Units, including the following changes: **Page 8:** Changed graphic. **Page 32:** Changed values in decimal and hexadecimal columns for signed binary. **Page 137:** Added information to explanation of TXD(236) and RXD(235) instructions. Information on new instructions added in new sections *3-23* and *3-24*. **Page 190:** Changed and added information in overview and graphic. Information in table changed and new information added. **Pages 192 and 193:** Reorganized and changed information in *5-1-2*. **Page 194:** Expanded table. **Page 198:** Expanded table, removed and added new notes, and added information on CX-Programmer system files. **Page 202:** Changed “numeric or characters” to “characters” for cell format. **Page 204:** Changed table and added note. **Page 207:** Added information on symbol table files and comment files. **Page 228:** Added new information including tables on I/O allocation status and verification. Also added information on the simple backup function. **Page 229:** Added information on data comparison **Pages 232 and 233:** Corrected I/O memory to CPU Unit in table headings. **Page 233:** Added new table. **Page 240:** Added information on I/O allocation status and verification. **Page 277:** Added information on Serial Gateway to table. **Page 285:** Added new section, *6-3-4,* on Serial Gateway. **Pages 345 and 364:** Added information to table.
08	June 2005	The following minor changes were made. **Page v:** Added information on general precaution notices. **Page xxi:** Added information on warranty and liability.
09	October 2006	Information was added on CS/CJ-series CPU Units with unit version 4.0. Descriptions and contents were improved. The 15-Mbyte flash memory was deleted.
10	April 2007	Information was added on the CJ1H-CPU□□H-R CPU Units.
11	October 2007	**Page xvi:** Added note. **Page 83:** Corrected the number of days for binary for LONG TIMER. **Page 160:** Corrected “binary result” to “BCD result” for BIT COUNTER. **Page 305:** Added sentence at end of overview. **Page 315:** Corrected last operand in each timer instruction.
12	January 2008	Added information on unit version 4.1 of the CJ1H-CPU□□H-R CPU Units (CJ1-H-R). **Page xvi:** Added information to note.