Machine Automation Controller

NJ/NX-series

CPU Unit
Software

User's Manual

NX701-17□□
NX701-16□□
NJ501-15□□
NJ501-14□□
NJ501-13□□
NJ301-12□□
NJ301-11□□
NJ101-10□□
NJ101-90□□

CPU Unit

W501-E1-12

NOTE

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.

No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- Microsoft, Windows, Windows Vista, Excel, and Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the USA and other countries.
- EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- ODVA, CIP, CompoNet, DeviceNet, and EtherNet/IP are trademarks of ODVA.
- The SD and SDHC logos are trademarks of SD-3C, LLC.

Other company names and product names in this document are the trademarks or registered trademarks of their respective companies.

Copyrights

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

Introduction

Thank you for purchasing an NJ/NX-series CPU Unit.
This manual contains information that is necessary to use the NJ/NX-series CPU Unit. Please read this manual and make sure you understand the functionality and performance of the NJ/NX-series CPU Unit before you attempt to use it in a control system.
Keep this manual in a safe place where it will be available for reference during operation.

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

For programming, this manual is intended for personnel who understand the programming language specifications in international standard IEC 61131-3 or Japanese standard JIS B 3503.

Applicable Products

This manual covers the following products.

- NX-series CPU Units
  - NX701-17□□
  - NX701-16□□

- NJ-series CPU Units
  - NJ501-15□□
  - NJ501-14□□
  - NJ501-13□□
  - NJ301-12□□
  - NJ301-11□□
  - NJ101-10□□
  - NJ101-90□□

Part of the specifications and restrictions for the CPU Units are given in other manuals. Refer to *Relevant Manuals* on page 2 and *Related Manuals* on page 43.

Relevant Manuals

The following table provides the relevant manuals for the NJ/NX-series CPU Units.
Read all of the manuals that are relevant to your system configuration and application before you use the NJ/NX-series CPU Unit.

Most operations are performed from the Sysmac Studio Automation Software. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for information on the Sysmac Studio.

Purpose of use	Manual										
	Basic information										
	NX-series CPU Unit Hardware User's Manual	NJ-series CPU Unit Hardware User's Manual	NJ/NX-series CPU Unit Software User's Manual	NJ/NX-series Instructions Reference Manual	NJ/NX-series CPU Unit Motion Control User's Manual	NJ/NX-series Motion Control Instructions Reference Manual	NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual	NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual	NJ-series Database Connection CPU Units User's Manual	NJ-series SECS/GEM CPU Units User's Manual	NJ/NX-series Troubleshooting Manual
Introduction to NX-series Controllers	●										
Introduction to NJ-series Controllers		●									
Setting devices and hardware	●	●									
Using motion control					●						
Using EtherCAT							●				
Using EtherNet/IP								●			
Using the database connection service									●		
Using the GEM Services										●	
Software settings			●								
Using motion control					●						
Using EtherCAT							●				
Using EtherNet/IP								●			
Using the database connection service									●		
Using the GEM Services										●	
Writing the user program			●	●							
Using motion control					●	●					
Using EtherCAT							●				
Using EtherNet/IP								●			
Using the database connection service									●		
Using the GEM Services										●	
Programming error processing											●
Testing operation and debugging			●								
Using motion control					●						
Using EtherCAT							●				
Using EtherNet/IP								●			
Using the database connection service									●		
Using the GEM Services										●	
Learning about error management and corrections [*1]	▲	▲	▲		▲		▲	▲	▲	▲	●

Purpose of use	Manual										
	Basic information										
	NX-series CPU Unit Hardware User's Manual	**NJ-series CPU Unit Hardware User's Manual**	**NJ/NX-series CPU Unit Software User's Manual**	**NJ/NX-series Instructions Reference Manual**	**NJ/NX-series CPU Unit Motion Control User's Manual**	**NJ/NX-series Motion Control Instructions Reference Manual**	**NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual**	**NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual**	**NJ-series Database Connection CPU Units User's Manual**	**NJ-series SECS/GEM CPU Units User's Manual**	**NJ/NX-series Troubleshooting Manual**
Maintenance	●	●									
Using motion control					●						
Using EtherCAT							●				
Using EtherNet/IP								●			

*1 Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for the error management concepts and an overview of the error items. Refer to the manuals that are indicated with triangles for details on errors for the corresponding Units.

Manual Structure

Page Structure

The following page structure is used in this manual.

Level 1 heading

Level 2 heading

Level 3 heading

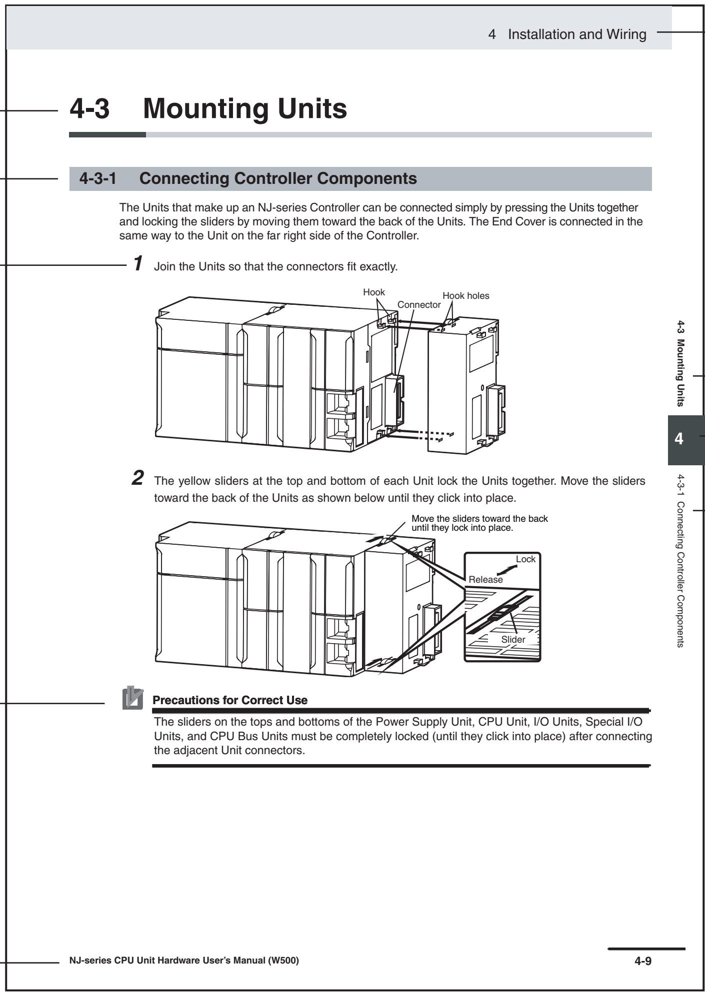
Gives the current headings.

Level 2 heading

Level 3 heading

A step in a procedure

Indicates a procedure.



Page tab

Gives the number of the main section.

Special information

Icons indicate precautions, additional information, or reference information.

Manual name

This illustration is provided only as a sample. It may not literally appear in this manual.

Special Information

Special information in this manual is classified as follows:

Precautions for Safe Use

Precautions on what to do and what not to do to ensure safe usage of the product.

Precautions for Correct Use

Precautions on what to do and what not to do to ensure proper operation and performance.

Additional Information

Additional information to read as required.
This information is provided to increase understanding or make operation easier.

Version Information

Information on differences in specifications and functionality for CPU Units with different unit versions and for different versions of the Sysmac Studio is given.

Note References are provided to more detailed or related information.

Precaution on Terminology

In this manual, “download” refers to transferring data from the Sysmac Studio to the physical Controller and “upload” refers to transferring data from the physical Controller to the Sysmac Studio.

For the Sysmac Studio, synchronization is used to both upload and download data. Here, “synchronize” means to automatically compare the data for the Sysmac Studio on the computer with the data in the physical Controller and transfer the data in the direction that is specified by the user.

Sections in this Manual

CONTENTS

Section 3 I/O Ports, Slave Configuration, and Unit Configuration

Section 4 Controller Setup

Section 5 Designing Tasks

Section 6 Programming

Section 9 Backup Functions

Appendices

Index

Terms and Conditions Agreement

Warranty, Limitations of Liability

Warranties

- **Exclusive Warranty**

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

- **Limitations**

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

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

- **Buyer Remedy**

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

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

Limitation on Liability; Etc

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

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

Application Considerations

Suitability of Use

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

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

Programmable Products

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

Disclaimers

Performance Data

Data presented in Omron Company websites, catalogs and other materials is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of Omron's test conditions, and the user must correlate it to actual application requirements. Actual performance is subject to the Omron's Warranty and Limitations of Liability.

Change in Specifications

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

Errors and Omissions

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

Safety Precautions

Definition of Precautionary Information

The following notation is used in this manual to provide precautions required to ensure safe usage of an NJ/NX-series Controller. The safety precautions that are provided are extremely important to safety. Always read and heed the information provided in all safety precautions.
The following notation is used.

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.

Precautions for Safe Use

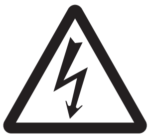
Indicates precautions on what to do and what not to do to ensure safe usage of the product.

Precautions for Correct Use

Indicates precautions on what to do and what not to do to ensure proper operation and performance.

Symbols

The circle and slash symbol indicates operations that you must not do.
The specific operation is shown in the circle and explained in text.
This example indicates prohibiting disassembly.



The triangle symbol indicates precautions (including warnings).
The specific operation is shown in the triangle and explained in text.
This example indicates a precaution for electric shock.

The triangle symbol indicates precautions (including warnings).
The specific operation is shown in the triangle and explained in text.
This example indicates a general precaution.

The filled circle symbol indicates operations that you must do.
The specific operation is shown in the circle and explained in text.
This example shows a general precaution for something that you must do.

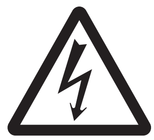
The triangle symbol indicates precautions (including warnings).
The specific operation is shown in the triangle and explained in text.
This example indicates a precaution for high temperatures.

Warnings

⚠ WARNING

During Power Supply

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



Do not attempt to take any Unit apart. In particular, high-voltage parts are present in the Power Supply Unit while power is supplied or immediately after power is turned OFF. Touching any of these parts may result in electric shock. There are sharp parts inside the Unit that may cause injury.

Fail-safe Measures

Provide safety measures in external circuits to ensure safety in the system if an abnormality occurs due to malfunction of the CPU Unit, slaves, or Units or due to other external factors affecting operation. Not doing so may result in serious accidents due to incorrect operation.

Emergency stop circuits, interlock circuits, limit circuits, and similar safety measures must be provided in external control circuits.

The Controller 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 safe operation of the system.

The CPU Unit will turn OFF all outputs from Basic Output Units and Digital Output Units in the following cases. The slaves will operate according to the settings in the slaves.

- If an error occurs in the power supply
- If the power supply connection becomes faulty
- If a CPU watchdog timer error or CPU reset occurs
- If a major fault level Controller error occurs
- While the CPU Unit is on standby until RUN mode is entered after the power is turned ON
- If a system initialization error occurs

External safety measures must be provided to ensure safe operation of the system in such cases.

If external power supplies for slaves or other devices are overloaded or short-circuited, the voltage will drop, outputs will turn OFF, and the system may be unable to read inputs. Provide external safety measures in controls with monitoring of external power supply voltage as required so that the system operates safely in such a case.

Unintended outputs may occur when an error occurs in variable memory or in memory used for CJ-series Units. As a countermeasure for such problems, external safety measures must be provided to ensure safe operation of the system.

Provide measures in the communications system and user program to ensure safety in the overall system even if errors or malfunctions occur in data link communications or remote I/O communications.

If there is interference in remote I/O communications or if a major fault level error occurs, output status will depend on the products that are used. Confirm the operation that will occur when there is interference in communications or a major fault level error, and implement safety measures. Correctly set all of the settings in the slaves and Units.

The NJ/NX-series Controller continues normal operation for a certain period of time when a momentary power interruption occurs. This means that the NJ/NX-series Controller may receive incorrect signals from external devices that are also affected by the power interruption. Accordingly, take suitable actions, such as external fail-safe measures and interlock conditions, to monitor the power supply voltage of the external device as required.

You must take fail-safe measures to ensure safety in the event of incorrect, missing, or abnormal signals caused by broken signal lines, momentary power interruptions, or other causes. Not doing so may result in serious accidents due to incorrect operation.

Voltage and Current Inputs

Make sure that the voltages and currents that are input to the slaves and Units are within the specified ranges.
Inputting voltages or currents that are outside of the specified ranges may cause accidents or fire.

Downloading

Always confirm safety at the destination before you transfer a user program, configuration data, setup data, device variables, or values in memory used for CJ-series Units from the Sysmac Studio. The devices or machines may perform unexpected operation regardless of the operating mode of the CPU Unit.

Actual Operation

Check the user program, data, and parameter settings for proper execution before you use them for actual operation.

Cautions

⚠ Caution

Application

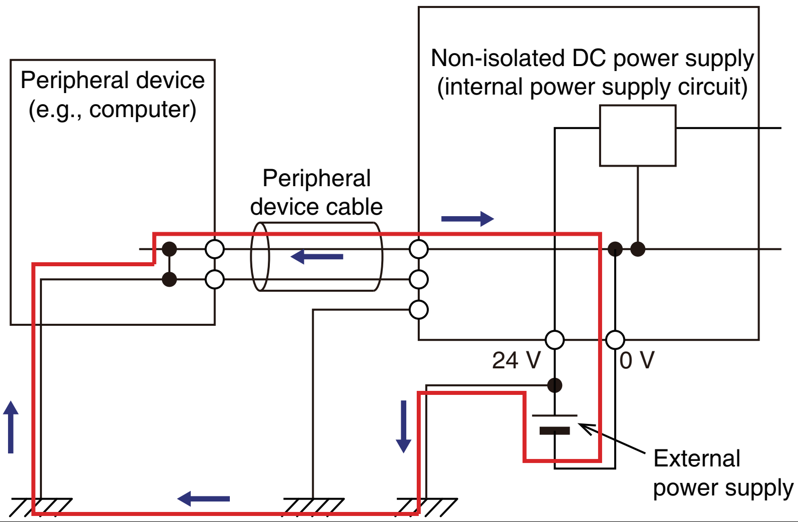
Do not touch any Unit when power is being supplied or immediately after the power supply is turned OFF. Doing so may result in burn injury.

Wiring

Be sure that all terminal screws and cable connector screws are tightened to the torque specified in the relevant manuals. The loose screws may result in fire or malfunction.

When you connect a computer or other peripheral device to a Controller that has a non-isolated DC Power Supply Unit, either ground the 0-V side of the external power supply or do not ground it at all.

If the peripheral devices are grounded incorrectly, the external power supply may be short-circuited. Never ground the 24-V side of the power supply, as shown in the following figure.



Online Editing

Execute online editing only after confirming that no adverse effects will be caused by deviations in the timing of I/O. If you perform online editing, the task execution time may exceed the task period, I/O may not be refreshed with external devices, input signals may not be read, and output timing may change.

Precaution on Error Message That Says an Instruction May Cause Unintended Operation

Instructions may results in unexpected operation and affect the system if you clear the selection of the *Detect an error when an in-out variable is passed to specific instruction argument* Check Box in the Program Check Area. Always confirm that the conditions for use that are given in the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) are met before you clear the selection of this check box.

Version Information

This error message is displayed by and the above option setting is available on Sysmac Studio version 1.02.

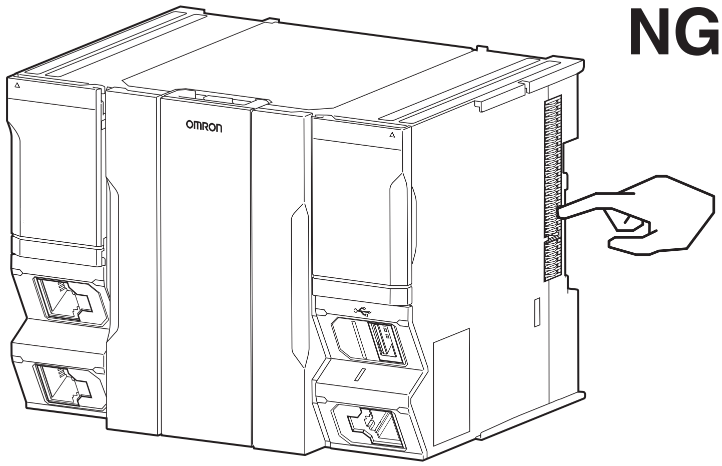
Precautions for Safe Use

Disassembly and Dropping

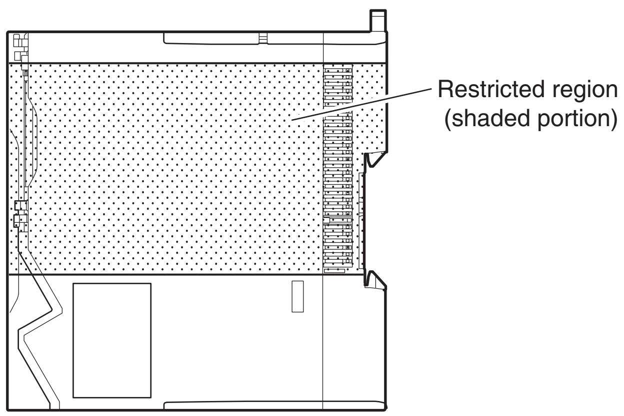
- Do not attempt to disassemble, repair, or modify any Units. Doing so may result in malfunction or fire.
- Do not drop any Unit or subject it to abnormal vibration or shock. Doing so may result in Unit malfunction or burning.

Mounting

- The sliders on the tops and bottoms of the Power Supply Unit, CPU Unit, I/O Units, and other Units must be completely locked (until they click into place) after connecting the adjacent Unit connectors.
- Do not apply labels or tape to the Unit. When the Unit is installed or removed, adhesive or scraps may adhere to the pins in the NX bus connector, which may result in malfunctions.
- Do not touch the pins in the NX bus connector on the Unit. Dirt may adhere to the pins in the NX bus connector, which may result in malfunctions.



- Do not write on the CPU Unit with ink within the restricted region that is shown in the following figure. Also do not get this area dirty. When the Unit is installed or removed, ink or dirt may adhere to the pins in the NX connector, which may result in malfunctions.



Installation

- Always connect to a ground of 100 Ω or less when installing the Units. A ground of 100 Ω or less must be installed when shorting the GR and LG terminals on the Power Supply Unit.

Wiring

- Follow the instructions in this manual to correctly perform wiring.
  Double-check all wiring and switch settings before turning ON the power supply.
- Use crimp terminals for wiring.
  Do not connect bare stranded wires directly to terminals.
- Do not pull on the cables or bend the cables beyond their natural limit.
  Do not place heavy objects on top of the cables or other wiring lines. Doing so may break the cables.
- Mount terminal blocks and connectors only after checking the mounting location carefully.
  Be sure that the terminal blocks, expansion cables, and other items with locking devices are properly locked into place.
- Always remove any dustproof labels that are on the top of the Units when they are shipped before you turn ON the power supply. If the labels are not removed, heat will accumulate and malfunctions may occur.
- Before you connect a computer to the CPU Unit, disconnect the power supply plug of the computer from the AC outlet. Also, if the computer has an FG terminal, make the connections so that the FG terminal has the same electrical potential as the GR terminal on the Power Supply Unit. A difference in electrical potential between the computer and Controller may cause failure or malfunction.
- If the external power supply to an Output Unit or slave has polarity, connect it with the correct polarity. If the polarity is reversed, current may flow in the reverse direction and damage the connected devices regardless of the operation of the Controller.

Power Supply Design

- Do not exceed the rated supply capacity of the Power Supply Units in the NJ/NX-series Controller. The rated supply capacities are given in the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) and *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500).
  If the capacity is exceeded, operation may stop, malfunctions may occur, or data may not be backed up normally for power interruptions.
  Use NJ-series Power Supply Units for both the NJ-series CPU Rack and Expansion Racks.
  Operation is not possible if a CJ-series Power Supply Unit is used with an NJ-series CPU Unit or an NJ-series Power Supply Unit is used with a CJ-series CPU Unit.
- Do not apply voltages or connect loads to the Output Units or slaves in excess of the maximum ratings.
- Surge current occurs when the power supply is turned ON. When selecting fuses or breakers for external circuits, consider the above precaution and allow sufficient margin in shut-off performance. Refer to the relevant manuals for surge current specifications. Refer to the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) and *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) for surge current specifications.
- If the full dielectric strength voltage is applied or turned OFF using the switch on the tester, the generated impulse voltage may damage the Power Supply Unit. Use the adjustment on the tester to gradually increase and decrease the voltage.
- Apply the voltage between the Power Supply Unit's L1 or L2 terminal and the GR terminal when testing insulation and dielectric strength.
- Do not supply AC power from an inverter or other device with a square-wave output. Internal temperature rise may result in smoking or burning. Always input a sinusoidal wave with the frequency that is given in the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) and *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500).
- Install external breakers and take other safety measures against short-circuiting in external wiring.

Turning ON the Power Supply

- It takes up to approximately 10 to 20 s to enter RUN mode after the power is turned ON. The outputs during this time behave according to the slave or Unit specifications. Use the RUN output on the Power Supply Unit, for example, to implement fail-safe circuits so that external devices do not operate incorrectly.
- Configure the external circuits so that the power supply to the control system turns ON only after the power supply to the Controller has turned ON. If the power supply to the Controller is turned ON after the control power supply, temporary errors may result in incorrect control system signals because the output terminals on Output Units may momentarily turn ON when power supply is turned ON to the Controller.
- If you transfer data from a backup file on an SD Memory Card to the Controller when the power supply is turned ON, properly select the data groups to transfer. If the data for an unintended data group is transferred to the Controller, it may cause the equipment to operate unpredictably.
- Never turn OFF the power supply to the Controller until RUN mode is entered after the power is turned ON. If the power supply is turned OFF, a Battery-backup Memory Check Error may occur at next time you start operation. If a Battery-backup Memory Check Error occurs, the variables retained are set to their initial values and the Holding, DM and EM Areas in memory used for CJ-series Units are cleared to all zeros. If you want to resume the operation, reload the correct data for the variables retained and CJ-series Unit memory, as required.

Turning OFF the Power Supply

- Never turn OFF the power supply to the Controller when the BUSY indicator is flashing. While the BUSY indicator is lit, the user program and settings in the CPU Unit are being backed up in the built-in non-volatile memory. This data will not be backed up correctly if the power supply is turned OFF. Also, a major fault level Controller error will occur the next time you start operation, and operation will stop.
- Do not turn OFF the power supply or remove the SD Memory Card while SD Memory Card access is in progress (i.e., while the SD BUSY indicator flashes). Data may become corrupted, and the Controller will not operate correctly if it uses corrupted data. To remove the SD Memory Card from the CPU Unit while the power supply is ON, press the SD Memory Card power supply switch and wait for the SD BUSY indicator to turn OFF before you remove the SD Memory Card.
- Do not disconnect the cable or turn OFF the power supply to the Controller when downloading data or the user program from Support Software.
- Always turn OFF the power supply to the Controller before you attempt any of the following.
  - Mounting or removing the Units
  - Assembling the Units
  - Setting DIP switches or rotary switches
  - Connecting cables or wiring the system
  - Connecting or disconnecting the connectors
  - Mounting or removing the Fan Unit

  The Power Supply Unit may continue to supply power to the rest of the Controller for a few seconds after the power supply turns OFF. The PWR indicator is lit during this time. Confirm that the PWR indicator is not lit before you perform any of the above.

Operation

- Confirm that no adverse effect will occur in the system before you attempt any of the following.
  - Changing the operating mode of the CPU Unit (including changing the setting of the Startup Mode)
  - Changing the user program or settings
  - Changing set values or present values

- Forced refreshing
- After you change any slave or Unit settings, carefully check the safety of the controlled system before you restart the Unit.
- If two different function modules are used together, such as when you use CJ-series Basic Units and EtherCAT slaves, take suitable measures in the user program and external controls to ensure that safety is maintained in the controlled system if one of the function modules stops. The relevant outputs will behave according to the slave or Unit specifications if a partial fault level error occurs in one of the function modules.
- Always confirm safety at the connected equipment before you reset Controller errors with an event level of partial fault or higher for the EtherCAT Master Function Module.
  When the error is reset, all slaves that were in any state other than Operational state due to a Controller error with an event level of partial fault or higher (in which outputs are disabled) will go to Operational state and the outputs will be enabled.
  Before you reset all errors or restart a slave, confirm that no Controller errors with an event level of partial fault have occurred for the EtherCAT Master Function Module.
- Always confirm safety at the connected equipment before you reset Controller errors for a CJ-series Special Unit. When a Controller error is reset, the Unit where the Controller error with an event level of observation or higher will be restarted.
  Before you reset all errors, confirm that no Controller errors with an event level of observation or higher have occurred for the CJ-series Special Unit. Observation level events do not appear on the Controller Error Tab Page, so it is possible that you may restart the CJ-series Special Unit without intending to do so.
  You can check the status of the *_CJB_UnitErrSta[0,0]* to *_CJB_UnitErrSta[3,9]* error status variables on a Watch Tab Page to see if an observation level Controller error has occurred.

Battery Backup

- The user program and initial values for the variables are stored in non-volatile memory in the CPU Unit. The present values of variables with the Retain attribute and the values of the Holding, DM, and EM Areas in the memory used for CJ-series Units are backed up by a Battery. If the Battery is not connected or the Battery is exhausted, the CPU Unit detects a Battery-backup Memory Check Error. If that error is detected, variables with a Retain attribute are set to their initial values and the Holding, DM, and EM Areas in memory used for CJ-series Units are cleared to all zeros. Perform thorough verifications and provide sufficient measures to ensure that the devices perform safe operation for the initial values of the variables with Retain attributes and the resulting operation.
- The absolute encoder home offsets are backed up by a Battery. If the CPU Unit detects a low battery voltage or the absence of a mounted battery when the power supply to the Controller is turned ON, the absolute encoder home offsets are cleared to zeros and an Encoder Home Offset Read Error occurs. Reset the error and perform homing to define home. If you do not define home, unintended operation of the controlled system may occur.

Debugging

- Forced refreshing ignores the results of user program execution and refreshes I/O with the specified values. If forced refreshing is used for inputs for which I/O refreshing is not supported, the inputs will first take the specified values, but they will then be overwritten by the user program. This operation differs from the force-set/reset functionality of the CJ-series PLCs.
- You cannot upload or download information for forced refreshing with the Sysmac Studio.
  After downloading data that contains forced refreshing, change to RUN mode and then use the Sysmac Studio to perform the operation for forced refreshing.
  Depending on the difference in the forced status, the control system may operate unexpectedly.
- Do not specify the same address for the AT specification for more than one variable.
  Doing so would allow the same entity to be accessed with different variable names, which would make the user program more difficult to understand and possibly cause programming mistakes.

General Communications

- When you use data link communications, check the error information that is given in *_ErrSta* (Controller Error Status) to make sure that no error has occurred in the source device. Create a user program that uses reception data only when there is no error in the source device. If there is an error in the source device, the data for the data link may contain incorrect values.
- Unexpected operation may result if inappropriate data link tables are set. Even if appropriate data link tables have been set, confirm that the controlled system will not be adversely affected before you transfer the data link tables. The data links start automatically after the data link tables are transferred.
- All CPU Bus Units are restarted when routing tables are transferred from Support Software to the CPU Unit. Confirm that the system will not be adversely affected by restarting before you transfer the routing tables.
- Tag data links will stop between related nodes while tag data link parameters are transferred during Controller operation. Confirm that the system will not be adversely affected before you transfer the tag data link parameters.

EtherNet/IP Communications

- Make sure that the communications distance, number of nodes connected, and method of connection for EtherNet/IP are within specifications. Do not connect EtherNet/IP communications to EtherCAT or other networks. An overload may cause the network to fail or malfunction.
- All related EtherNet/IP nodes are reset when you transfer settings for the built-in EtherNet/IP port (including IP addresses and tag data links settings). The settings can only be enabled after the reset. Confirm that the system will not be adversely affected by resetting nodes before you transfer the settings.
- If EtherNet/IP tag data links (cyclic communications) are used with a repeating hub, the communications load on the network will increase. This will increase collisions and may prevent stable communications. Do not use repeating hubs on networks where tag data links are used. Use an Ethernet switch instead.
- When you use two EtherNet/IP networks separately, provide separate hubs to connect nodes in each network.
- Be careful not to mix nodes that belong different EtherNet/IP networks when you connect them.

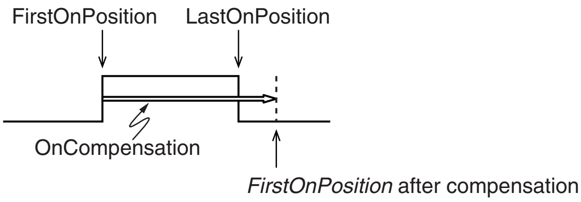
EtherCAT Communications

- Make sure that the communications distance, number of nodes connected, and method of connection for EtherCAT are within specifications.
  Do not connect EtherCAT communications to EtherNet/IP, a standard in-house LAN, or other networks. An overload may cause the network to fail or malfunction.
- Malfunctions or unexpected operation may occur for some combinations of EtherCAT revisions of the master and slaves. If you disable the revision check in the network settings, use the Sysmac Studio to check the slave revision settings in the master and the actual slave revisions, and then make sure that functionality is compatible in the slave manuals or other references. You can check the actual slave revisions from the Sysmac Studio or on slave nameplates.
- After you transfer the user program, the CPU Unit is restarted and communications with the EtherCAT slaves are cut off. During that period, the slave outputs behave according to the slave specifications. The time that communications are cut off depends on the EtherCAT network configuration.
  Before you transfer the user program, confirm that the system will not be adversely affected.
- If the Fail-soft Operation parameter is set to stop operation, process data communications will stop for all slaves when an EtherCAT communications error is detected in a slave. At that time, the Servo Drive will operate according to the Servo Drive specifications. For this reason, if Servo Drives are connected, the Servos for all axes will be turned OFF. Make sure that the Fail-soft Operation parameter setting results in safe operation when a device error occurs.

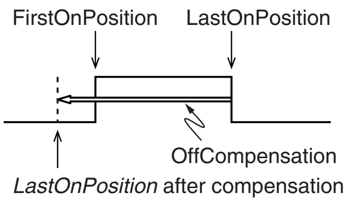
- EtherCAT communications are not always established immediately after the power supply is turned ON. Use the system-defined variables in the user program to confirm that communications are established before attempting control operations.
- If noise occurs or an EtherCAT slave is disconnected from the network, any current communications frames may be lost. If frames are lost, slave I/O data is not communicated, and unintended operation may occur. The slave outputs will behave according to the slave specifications. If a noise countermeasure or slave replacement is required, perform the following processing.
  - Program the Input Data Invalid system-defined variable as an interlock condition in the user program.
  - Set the PDO communications timeout detection count setting in the EtherCAT master to at least 2. Refer to the *NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual* (Cat. No. W505) for details.
- When an EtherCAT slave is disconnected or disabled, communications will stop and control of the outputs will be lost not only for the disconnected slave, but for all slaves connected after it. Confirm that the system will not be adversely affected before you disconnect or disable a slave.
- I/O data communications of NX bus are not always established immediately after the power supply is turned ON. Use the system-defined variables and the EtherCAT Coupler Unit device variables in the user program to confirm that I/O data communications are established before attempting control operations.
- You cannot use standard Ethernet hubs or repeater hubs with EtherCAT communications. If you use one of these, a major fault level error or other error may occur.

Motion Control

- Confirm the axis number carefully before you perform an MC Test Run.
- The motor is stopped if communications are interrupted between the Sysmac Studio and the CPU Unit during an MC Test Run. Connect the communications cable between the computer and CPU Unit securely and confirm that the system will not be adversely affected before you perform an MC Test Run.
- Always execute the Save Cam Table instruction if you change any of the cam data from the user program in the CPU Unit or from the Sysmac Studio. If the cam data is not saved, the previous condition will be restored when the power is turned ON again, possibly causing unexpected machine operation.
- The positive drive prohibit input (POT), negative drive prohibit input (NOT), and home proximity input (DEC) of the Servo Drive are used by the MC Function Module as the positive limit input, negative limit input, and home proximity input. Make sure that the signal widths for all of these input signals are longer than the control period of the MC Function Module. If the input signal widths are shorter than the control period, the MC Function Module may not be able to detect the input signals, resulting in incorrect operation.
- If you make any changes in the Detailed Settings Area of the Axis Basic Settings Display of the Sysmac Studio, make sure that the devices or machines perform the expected operation before you start actual operation.
  If the relationship between the functions of the Motion Control Function Module and the EtherCAT slave process data that is assigned to the axes is not correct, the devices or machines may perform unexpected operation.
- Always use the axis at a constant velocity for the MC_DigitalCamSwitch (Enable Digital Cam Switch) instruction.
  If you set the Count Mode to Rotary Mode, the following operation will occur if you use *OnCompensation* or *OffCompensation* and the axis velocity changes abruptly.
  - If the value of *OnCompensation* or *OffCompensation* is equivalent to the time for half a rotation or more, *InOperation* will be FALSE.
  - If the value of *OnCompensation* results in exceeding *LastOnPosition*, the output timing will be unstable.



- If the value of *OffCompensation* results in exceeding *FirstOnPosition*, the output timing will be unstable.



- Use the NX_AryDOutTimeStamp (Write Digital Output Array with Specified Time Stamp) instruction only after you confirm that *InOperation* from the MC_DigitalCamSwitch (Enable Digital Cam Switch) instruction is TRUE.

Restoring Data

- You cannot back up, restore, or compare some or all of the settings for certain slaves and Units. Also, you cannot back up, restore, or compare data for disabled slaves or Units. After you restore data, sufficiently confirm that operation is correct before you start actual operation.
- The absolute encoder home offsets are backed up with a Battery in the CPU Unit as absolute encoder information. If any of the following conditions is met, clear the absolute encoder home offsets from the list of data items to restore, and then restore the data. Then, define the absolute encoder home again. If you do not define home, unintended operation of the controlled system may occur.
  - The Servomotor or Servo Drive was changed since the data was backed up.
  - The absolute encoder was set up after the data was backed up.
  - The absolute data for the absolute encoder was lost.

Battery Replacement

- The Battery may leak, rupture, heat, or ignite. Never short-circuit, charge, disassemble, heat, or incinerate the Battery or subject it to strong shock.
- Dispose of any Battery that has been dropped on the floor or otherwise subjected to excessive shock. Batteries that have been subjected to shock may leak if they are used.
- UL standards require that only an experienced engineer replace the Battery. Make sure that an experienced engineer is in charge of Battery replacement.
- Apply power for at least five minutes before changing the Battery. Install a new Battery within five minutes (at 25°C) of turning OFF the power supply. If power is not supplied for at least 5 minutes, the saved data may be lost.
- We recommend replacing the Battery with the power turned OFF to prevent the CPU Unit's sensitive internal components from being damaged by static electricity and to prevent malfunctions. The Battery can be replaced without turning OFF the power supply. To do so, always touch a grounded piece of metal to discharge static electricity from your body before you start the procedure.
  After you replace the Battery, connect the Sysmac Studio and clear the Low Battery Voltage error.

Unit Replacement

- Make sure that the required data, including the user program, configurations, settings, variables, and memory used for CJ-series Units, is transferred to a CPU Unit that was replaced and to externally connected devices before restarting operation.
  Be sure to include the tag data link settings, routing tables, and other CPU Bus Unit data, which are stored in the CPU Unit.
- The absolute encoder home offsets are backed up with a Battery in the CPU Unit as absolute encoder information. When you change the combination of the CPU Unit and Servomotor, e.g., when you add or replace a Servomotor, define the absolute encoder home again.

Disposal

- Dispose of the product and Batteries according to local ordinances as they apply.

  廢電池請回收
- The following information must be displayed for all products that contain primary lithium batteries with a perchlorate content of 6 ppb or higher when shipped to or transported through the State of California, USA.

  Perchlorate Material - special handling may apply.
  See www.dtsc.ca.gov/hazardouswaste/perchlorate.
- The CPU Unit contains a primary lithium battery with a perchlorate content of 6 ppb or higher. Place the above information on the individual boxes and shipping boxes when shipping finished products that contain a CPU Unit to the State of California, USA.

Precautions for Correct Use

Storage and Installation

- Follow the instructions in this manual to correctly perform installation.
- Do not operate or store the Controller in the following locations. Operation may stop or malfunctions may occur.
  - 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
- Take appropriate and sufficient countermeasures when installing the Controller in the following locations.
  - Locations subject to strong, high-frequency noise
  - 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 lines
- Before touching a Unit, be sure to first touch a grounded metallic object in order to discharge any static build-up.
- Install the Controller away from sources of heat and ensure proper ventilation. Not doing so may result in malfunction, in operation stopping, or in burning.

Mounting

- An I/O bus check error will occur and the Controller will stop if an I/O Connecting Cable's connector is disconnected from the Rack. Be sure that the connectors are secure.
- When you install the Unit, be careful not to touch or bump the pins in the NX bus connector.
- When you handle the Unit, be careful not to apply stress to the pins in the NX bus connector.
  If the Unit is installed and the power supply is turned ON when the pins in the NX bus connector are deformed, contact failure may cause malfunctions.
- Always mount an End Cover to the end of the CPU Rack to protect the last Unit on the CPU Rack. Not mounting the End Cover may result in malfunction or failure of the CPU Unit.
- After you mount the Unit, always secure it with End Plates at both sides. If you do not secure it, the Unit may be damaged or malfunction.
- If you use DIN Track Insulation Spacers to install a CPU Rack, the height will be increased by approximately 10 mm. Make sure that the CPU Rack and connecting cables do not come into contact with other devices.

Wiring

- Do not allow foreign matter to enter the openings in the Unit. Doing so may result in Unit burning, electric shock, or failure.
- Do not allow wire clippings, shavings, or other foreign material to enter any Unit. Otherwise, Unit burning, failure, or malfunction may occur. Cover the Units or take other suitable countermeasures, especially during wiring work.
- For EtherCAT and EtherNet/IP, use the connection methods and cables that are specified in the *NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual* (Cat. No. W505) and the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506). Otherwise, communications may be faulty.
- Use the rated power supply voltage for the Power Supply Units. Take appropriate measures to ensure that the specified power with the rated voltage and frequency is supplied in places where the power supply is unstable.
- Make sure that the current capacity of the wire is sufficient. Otherwise, excessive heat may be generated. When cross-wiring terminals, the total current for all the terminals will flow in the wire. When wiring cross-overs, make sure that the current capacity of each of the wires is not exceeded.
- Do not touch the terminals on the Power Supply Unit immediately after turning OFF the power supply. Residual voltage may cause electrical shock.
- If you use reed switches for the input contacts for AC Input Units, use switches with a current capacity of 1 A or greater.
  If the capacity of the reed switches is too low, surge current may fuse the contacts.

Error Processing

- In applications that use the results of instructions that read the error status, consider the affect on the system when errors are detected and program error processing accordingly. For example, even the detection of a minor error, such as Battery replacement during operation, can affect the system depending on how the user program is written.
- If you change the event level of a Controller error, the output status when the error occurs may also change. Confirm safety before you change an event level.

Restoring and Automatically Transferring Data

- When you edit the restore command file or the automatic transfer command file, do not change anything in the file except for the "yes" and "no" specifications for the selectable data groups. If you change anything else in the file, the Controller may perform unexpected operation when you restore or automatically transfer the data.

Replacing Slaves and Units

- If you replace a slave or Unit, refer to the operation manual for the slave or Unit for information on the data required for individual slaves or Units and redo the necessary settings.

Task Settings

- If a Task Period Exceeded error occurs, shorten the programs to fit in the task period or increase the setting of the task period.

Motion Control

- Use the system-defined variable in the user program to confirm that EtherCAT communications are established before you attempt to execute motion control instructions. Motion control instructions are not executed normally if EtherCAT communications are not established.
- Use the system-defined variables to monitor for errors in communications with the slaves that are controlled by the motion control function module. Motion control instructions are not executed normally if an error occur in slave communications.
- Before you start an MC Test Run, make sure that the operation parameters are set correctly.
- Do not download motion control settings during an MC Test Run.

EtherCAT Communications

- If you need to disconnect the cable from an EtherCAT slave during operation, first disconnect the software connection to the EtherCAT slave or disable the EtherCAT slave and all of the EtherCAT slaves that are connected after it.
- Set the Servo Drives to stop operation if an error occurs in EtherCAT communications between the Controller and a Servo Drive.
- Make sure that all of the slaves to be restored are participating in the network before you reset a Network Configuration Verification Error, Process Data Communications Error, or Link OFF Error in the EtherCAT Master Function Module. If any slave is not participating when any of these errors is reset, the EtherCAT Master Function Module may access slave with a different node address than the specified node address or the error may not be reset correctly.
- Always use the specified EtherCAT slave cables. If you use any other cable, the EtherCAT master or the EtherCAT slaves may detect an error and one of the following may occur.
  - Continuous refreshing of process data communications will not be possible.
  - Continuous refreshing of process data communications will not end during the set cycle.

Battery Replacement

- Be sure to install a replacement Battery within two years of the production date shown on the Battery label.
- Turn ON the power after replacing the Battery for a CPU Unit that has been unused for a long time. Leaving the CPU Unit unused again without turning ON the power even once after the Battery is replaced may result in a shorter Battery life.
- When you replace the Battery, use the CJ1W-BAT01 Battery Set.

SD Memory Cards

- Insert the SD Memory Card all the way.
- Do not turn OFF the power supply to the Controller during SD Memory Card access. The files may be corrupted.
  If there is a corrupted file in the SD Memory Card, the file is automatically deleted by the restoration function when the power supply is turned ON.
- If you use an OMRON SD Memory Card, the end of the life of the SD Memory Card can be detected in the following ways.
  - *_Card1Deteriorated* (SD Memory Card Life Warning Flag) system-defined variable
  - SD Memory Card Life Exceeded event in the event log

  When the end of the life is detected in any of the above ways, replace the SD Memory Card.

Online Editing

When performing online editing in combination of a CPU Unit with a unit version of 1.04 or later and Sysmac Studio version 1.05 or higher, the CPU Unit saves a program updated by the online editing to built-in non-volatile memory. Sysmac Studio shows a message that it is in a backup operation. Do not turn OFF the power supply to the Controller while this message is displayed. If the power supply to the Controller is turned OFF, a Controller error will occur when the power supply is turned ON next time.

Regulations and Standards

Conformance to EC Directives

Applicable Directives

- EMC Directives
- Low Voltage Directive

Concepts

EMC Directive

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.*
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.

* Applicable EMC (Electromagnetic Compatibility) standards are as follows:
EMS (Electromagnetic Susceptibility): EN 61131-2 and EN 61000-6-2
EMI (Electromagnetic Interference): EN 61131-2 and EN 61000-6-4 (Radiated emission: 10-m regulations)

Low Voltage Directive

Always ensure that devices operating at voltages of 50 to 1,000 VAC and 75 to 1,500 VDC meet the required safety standards. The applicable directive is EN 61131-2.

Conformance to EC Directives

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

- The NJ/NX-series Controller must be installed within a control panel.
- You must use reinforced insulation or double insulation for the DC power supplies connected to DC Power Supply Units and I/O Units.
- NJ/NX-series Controllers that comply with EC Directives also conform to the Common Emission Standard (EN 61000-6-4). 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.

Conformance to KC Standards

Observe the following precaution if you use NX-series Units in Korea.

A 급 기기 (업무용 방송통신기자재)
이 기기는 업무용(A 급) 전자파적합기기로서 판매자
또는 사용자는 이 점을 주의하시기 바라며, 가정외의
지역에서 사용하는 것을 목적으로 합니다.

Class A Device (Broadcasting Communications Device for Office Use)

This device obtained EMC registration for office use (Class A), and it is intended to be used in places other than homes.

Sellers and/or users need to take note of this.

Conformance to Shipbuilding Standards

The NJ/NX-series Controllers comply with the following shipbuilding standards. Applicability to the shipbuilding standards is based on certain usage conditions. It may not be possible to use the product in some locations. Contact your OMRON representative before attempting to use a Controller on a ship.

Usage Conditions for NK and LR Shipbuilding Standards

- The NJ/NX-series Controller must be installed within a control panel.
- Gaps in the door to the control panel must be completely filled or covered with gaskets or other material.
- The following noise filter must be connected to the power supply line.

Noise Filter

Manufacturer	Model
Cosel Co., Ltd.	TAH-06-683

Software Licenses and Copyrights

This product incorporates certain third party software. The license and copyright information associated with this software is available at http://www.fa.omron.co.jp/nj_info_e/.

Versions

Unit versions are used to manage the hardware and software in NJ/NX-series Units and EtherCAT slaves. The unit version is updated each time there is a change in hardware or software specifications. Even when two Units or EtherCAT slaves have the same model number, they will have functional or performance differences if they have different unit versions.

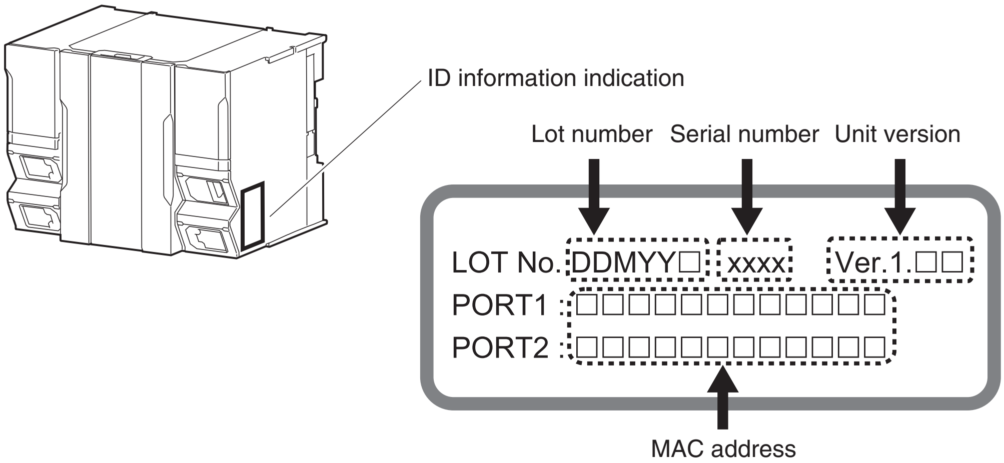
Checking Versions

You can check versions on the ID information indications or with the Sysmac Studio.

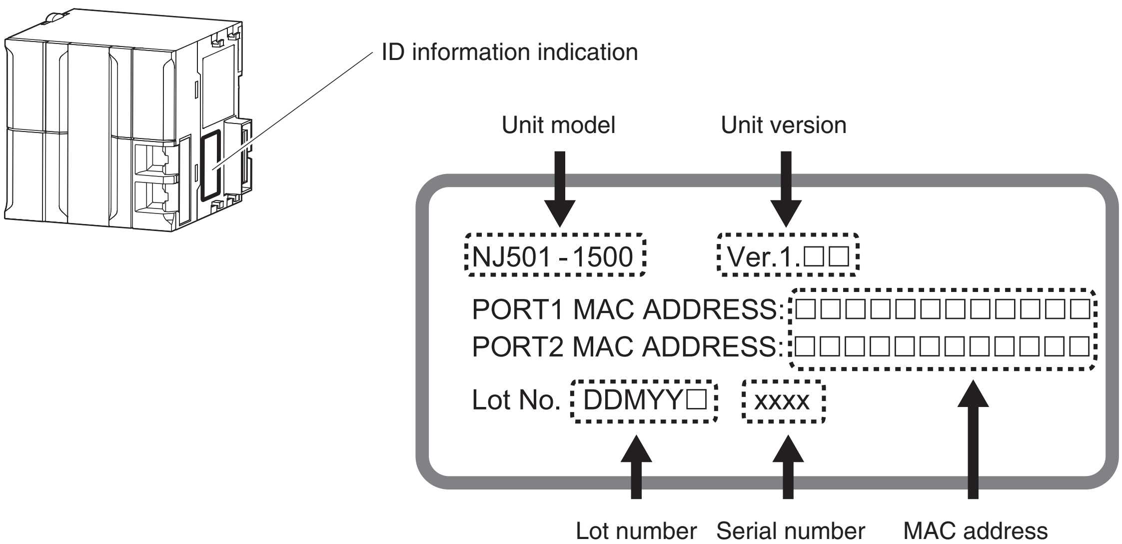
Checking Unit Versions on ID Information Indications

The unit version is given on the ID information indication on the side of the product.

The ID information on an NX-series NX701-□□□□ CPU Unit is shown below.



The ID information on an NJ-series NJ501-1500 CPU Unit is shown below.



Checking Unit Versions with the Sysmac Studio

You can use the Sysmac Studio to check unit versions. The procedure is different for Units and for EtherCAT slaves.

● Checking the Unit Version of an NX-series CPU Unit

You can use the Production Information while the Sysmac Studio is online to check the unit version of a Unit. You can only do this for the CPU Unit.

1 Right-click **CPU Rack** under **Configurations and Setup** - **CPU/Expansion Racks** in the Multiview Explorer and select ***Production Information***.

The Production Information Dialog Box is displayed.

● Checking the Unit Version of an NJ-series CPU Unit

You can use the Production Information while the Sysmac Studio is online to check the unit version of a Unit. You can do this for the CPU Unit, CJ-series Special I/O Units, and CJ-series CPU Bus Units. You cannot check the unit versions of CJ-series Basic I/O Units with the Sysmac Studio.

Use the following procedure to check the unit version.

1 Double-click **CPU/Expansion Racks** under **Configurations and Setup** in the Multiview Explorer. Or, right-click **CPU/Expansion Racks** under **Configurations and Setup** and select ***Edit*** from the menu.

The Unit Editor is displayed.

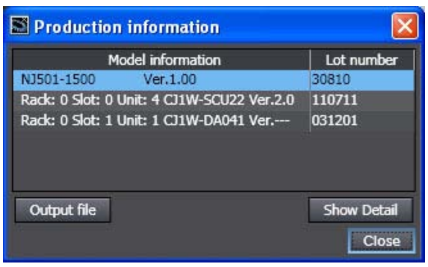
2 Right-click any open space in the Unit Editor and select ***Production Information***.

The Production Information Dialog Box is displayed.

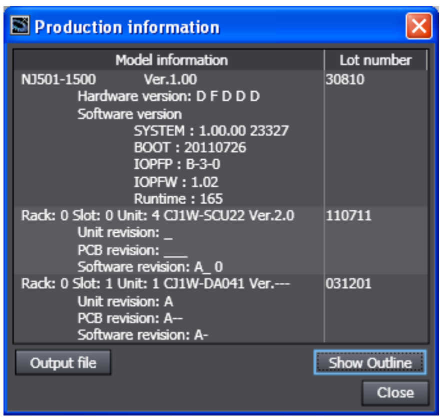
● Changing Information Displayed in Production Information Dialog Box

1 Click the **Show Detail** or **Show Outline** Button at the lower right of the Production Information Dialog Box.

The view will change between the production information details and outline.



Outline View



Detail View

The information that is displayed is different for the Outline View and Detail View. The Detail View displays the unit version, hardware version, and software versions. The Outline View displays only the unit version.

● Checking the Unit Version of an EtherCAT Slave

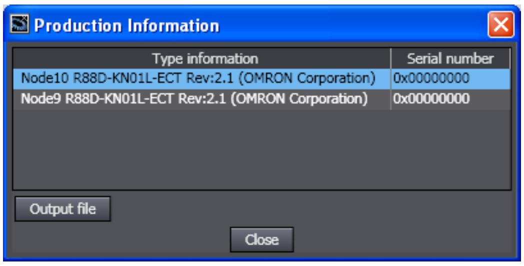
You can use the Production Information while the Sysmac Studio is online to check the unit version of an EtherCAT slave. Use the following procedure to check the unit version.

1 Double-click **EtherCAT** under **Configurations and Setup** in the Multiview Explorer. Or, right-click **EtherCAT** under **Configurations and Setup** and select ***Edit*** from the menu.

The EtherCAT Tab Page is displayed.

2 Right-click the master on the EtherCAT Tab Page and select ***Display Production Information***.

The Production Information Dialog Box is displayed.
The unit version is displayed after “Rev.”



Unit Versions of CPU Units and Sysmac Studio Versions

The functions that are supported depend on the unit version of the NJ/NX-series CPU Unit. The version of Sysmac Studio that supports the functions that were added for an upgrade is also required to use those functions.

Refer to *A-13 Version Information for NX-series Controllers* and *A-14 Version Information for NJ-series Controllers* for the relationship between the unit versions of the CPU Units and the Sysmac Studio versions, and for the functions that are supported by each unit version.

Related Manuals

The followings are the manuals related to this manual. Use these manuals for reference.

Manual name	Cat. No.	Model numbers	Application	Description
NX-series CPU Unit Hardware User's Manual	W535	NX701-□□□□	Learning the basic specifications of the NX-series CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX-series system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection Use this manual together with the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ-series CPU Unit Hardware User's Manual	W500	NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning the basic specifications of the NJ-series CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NJ-series system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General Specifications • Installation and wiring • Maintenance and inspection Use this manual together with the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series CPU Unit Software User's Manual	W501	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning how to program and set up an NJ/NX-series CPU Unit. Mainly software information is provided.	The following information is provided on a Controller built with an NJ/NX-series CPU Unit. • CPU Unit operation • CPU Unit features • Initial settings • Programming based on IEC 61131-3 language specifications Use this manual together with the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) or *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500).
NJ/NX-series Instructions Reference Manual	W502	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning detailed specifications on the basic instructions of an NJ/NX-series CPU Unit.	The instructions in the instruction set (IEC 61131-3 specifications) are described. When programming, use this manual together with the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) or *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and with the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series CPU Unit Motion Control User's Manual	W507	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about motion control settings and programming concepts.	The settings and operation of the CPU Unit and programming concepts for motion control are described. When programming, use this manual together with the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) or *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and with the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).

Manual name	Cat. No.	Model numbers	Application	Description
NJ/NX-series Motion Control Instructions Reference Manual	W508	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the specifications of the motion control instructions.	The motion control instructions are described. When programming, use this manual together with the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) or *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500), with the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501), and with the *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507).
NJ/NX-series CPU Unit Built-in EtherCAT® Port User's Manual	W505	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Using the built-in EtherCAT port on an NJ/NX-series CPU Unit.	Information on the built-in EtherCAT port is provided. This manual provides an introduction and provides information on the configuration, features, and setup. Use this manual together with the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) or *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and with the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual	W506	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Using the built-in EtherNet/IP port on an NJ/NX-series CPU Unit.	Information on the built-in EtherNet/IP port is provided. Information is provided on the basic setup, tag data links, and other features. Use this manual together with the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) or *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and with the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ-series Database Connection CPU Units User's Manual	W527	NJ501-1□20	Using the database connection service with NJ-series Controllers	Describes the database connection service.
NJ-series SECS/GEM CPU Units User's Manual	W528	NJ501-1340	Using the GEM Services with NJ-series Controllers.	Provides information on the GEM Services.
NJ/NX-series Troubleshooting Manual	W503	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the errors that may be detected in an NJ/NX-series Controller.	Concepts on managing errors that may be detected in an NJ/NX-series Controller and information on individual errors are described. Use this manual together with the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) or *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and with the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
Sysmac Studio Version 1 Operation Manual	W504	SYSMAC -SE2□□□	Learning about the operating procedures and functions of the Sysmac Studio.	Describes the operating procedures of the Sysmac Studio.
NX-series EtherCAT® Coupler Unit User's Manual	W519	NX-ECC□□□	Learning how to use an NX-series EtherCAT Coupler Unit and EtherCAT Slave Terminals.	The following items are described: the overall system and configuration methods of an EtherCAT Slave Terminal (which consists of an NX-series EtherCAT Coupler Unit and NX Units), and information on hardware, setup, and functions to set up, control, and monitor NX Units through EtherCAT.
NX-series Safety Control Unit User's Manual	Z930	NX-SL□□□□ NX-SI□□□□ NX-SO□□□□	Learning how to use NX-series Safety Control Units	Describes the hardware, setup methods, and functions of the NX-series Safety Control Units.
Vision System FH/FZ5 Series Vision System User's Manual	Z340	FH-1□□□ FH-3□□□ FZ5-L35□ FZ5-6□□ FZ5-11□□	Learning how to use the FH/FZ5-series Vision Systems.	Describes the software functions, setup and operating methods required for using the FH/FZ5-series system.

Manual name	Cat. No.	Model numbers	Application	Description
Vision Sensor FQ-M series Specialized Vision Sensor for Positioning User's Manual	Z314	FQ-MS12□	Learning how to use the Specialized Vision Sensors for Positioning.	Describes the hardware, setup methods and functions of the Specialized Vision Sensors for Positioning.
Vision Sensor FZ3 Series User's Manual	Z290	FZ3-□□□□	Learning how to use the FZ3-series Vision Sensors.	Describes the software functions, setup and operating methods of the FZ3-series Vision Sensors.
Displacement Sensor ZW series Confocal Fiber Type Displacement Sensor User's Manual	Z332	ZW-CE1□	Learning how to use the ZW-series Displacement Sensors.	Describes the hardware, setup methods and functions of the ZW-series Displacement Sensors.
CJ-series Special Unit Manuals for NJ-series CPU Unit	W490	CJ1W-AD□□□ CJ1W-DA□□□ CJ1W-MAD42	Learning how to use CJ-series Units with an NJ-series CPU Unit.	The methods and precautions for using CJ-series Units with an NJ501 CPU Unit are described, including access methods and programming interfaces. Manuals are available for the following Units. Analog I/O Units, Insulated-type Analog I/O Units, Temperature Control Units, ID Sensor Units, High-speed Counter Units, Serial Communications Units, DeviceNet Units, EtherNet/IP Units and CompoNet Master Units. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
	W491	CJ1W-TC□□□		
	W492	CJ1W-CT021		
	W498	CJ1W-PDC15 CJ1W-PH41U CJ1W-AD04U		
	W493	CJ1W-CRM21		
	W494	CJ1W-SCU□□		
	W495	CJ1W-EIP21		
	W497	CJ1W-DRM21		
	Z317	CJ1W-V680□□□		
NA-series Programmable Terminal Software User's Manual	V118	NA5-□W□□□□	Learning about NA-series PT pages and object functions.	Describes the pages and object functions of the NA-series Programmable Terminals.
NS-series Programmable Terminals Programming Manual	V073	NS15-□□□□□ NS12-□□□□□ NS10-□□□□□ NS8-□□□□□ NS5-□□□□□	Learning how to use the NS-series Programmable Terminals.	Describes the setup methods, functions, etc. of the NS-series Programmable Terminals.
CX-Designer User's Manual	V099	---	Learning to create screen data for NS-series Programmable Terminals.	Describes operating procedures for the CX-Designer.

Terminology

Term	Description
absolute encoder home offset	This data is used to restore in the CPU Unit the actual position of a Servo Drive with an absolute encoder. The offset is the difference between the command position after homing and the absolute data that is read from the absolute encoder.
array specification	One of the variable specifications. An array variable contains multiple elements of the same data type. The elements in the array are specified by serial numbers called subscripts that start from the beginning of the array.
AT	One of the attributes of a variable. This attribute allows the user to specify what is assigned to a variable. An I/O port or an address in memory used for CJ-series Units can be specified.
axes group	A functional unit that groups together axes within the Motion Control Function Module.
Axes Group Variable	A system-defined variable that is defined as a structure and provides status information and some of the axes parameters for an individual axes group. An Axes Group Variable is used to specify an axes group for motion control instructions and to monitor the command interpolation velocity, error information, and other information for the axes group.
axis	A functional unit within the Motion Control Function Module. An axis is assigned to the drive mechanism in an external Servo Drive or the sensing mechanism in an external Encoder Input Slave Unit.
Axis Variable	A system-defined variable that is defined as a structure and provides status information and some of the axis parameters for an individual axis. An Axis Variable is used to specify an axis for motion control instructions and to monitor the command position, error information, and other information for the axis.
basic data type	Any of the data types that are defined by IEC 61131-3. They include Boolean, bit string, integer, real, duration, date, time of day, date and time, and text string data types. "Basic data type" is used as opposed to derivative data types, which are defined by the user.
cam data variable	A variable that represents the cam data as a structure array. A cam data variable is an array structure that consists of phases and displacements.
CJ-series Unit	Any of the CJ-series Units that can be used with an NJ-series Controller.
Communications Coupler Unit	The generic name of an interface unit for remote I/O communications on a network between NX Units and a host network master. For example, an EtherCAT Coupler Unit is a Communications Coupler Unit for an EtherCAT network.
Constant	One of the attributes of a variable. If you specify the Constant attribute for a variable, the value of the variable cannot be written by any instructions, ST operators, or CIP message communications.
Controller	The range of devices that are directly controlled by the CPU Unit. In the NX-series System, the Controller includes the CPU Rack and EtherCAT slaves (including general-purpose slaves and Servo Drives). In the NJ-series System, the Controller includes the CPU Rack, Expansion Racks, and EtherCAT slaves (including general-purpose slaves and Servo Drives).
Controller error	Errors that are defined by the NJ/NX-series System. “Controller error” is a collective term for major fault level, partial fault level, minor fault level, and observation Controller events.
Controller event	One of the events in the NJ/NX-series System. Controller events are errors and information that are defined by the system for user notification. A Controller event occurs when the system detects a factor that is defined as a Controller event.
Controller information	Information that is defined by the NJ/NX-series System that is not an error. It represents an information Controller event.
CPU Unit	The Unit that serves as the center of control for a Machine Automation Controller. The CPU Unit executes tasks, refreshes I/O for other Units and slaves, etc. The NJ/NX-series CPU Units include the NX701-□□□□, NJ501-□□□□ and NJ301-□□□□.

Term	Description
derivative data type	A data type that is defined by the user. Structures, unions, and enumerations are derivative data types.
device variable	A variable that is used to access a specific device through an I/O port.
download	To transfer data from the Sysmac Studio to the Controller with the synchronization operation of the Sysmac Studio.
edge	One of the attributes of a variable. This attribute makes a BOOL variable pass TRUE to a function block when the variable changes from FALSE to TRUE or when it changes from TRUE to FALSE.
enumeration	One of the derivative data types. This data type takes one item from a prepared name list of enumerators as its value.
enumerator	One of the values that an enumeration can take expressed as a character string. The value of an enumeration is one of the enumerators.
EtherCAT Master Function Module	One of the function modules. This function module controls the EtherCAT slaves as the EtherCAT master.
EtherNet/IP Function Module	One of the function modules. This function module controls the built-in EtherNet/IP port.
event log	A function that recognizes and records errors and other events.
Event Setup	Settings that define user-defined errors and user-defined information.
event task	A task that executes a user program only once when the task execution conditions are met.
FB	An acronym for "function block."
forced refreshing	Forcing the refreshing of an input from an external device or an output to an external device, e.g., when the user debugs a program. Addresses that are subject to forced refreshing can still be overwritten from the user program.
FUN	An abbreviation for "function."
function	A POU that is used to create an object that determines a unique output for the same input, such as for data processing.
function block	A POU that is used to create an object that can have a different output for the same input, such as for a timer or counter.
function module	One of the functional units of the software configuration of the CPU Unit.
general-purpose slave	Any of the EtherCAT slaves that cannot be assigned to an axis.
global variable	A variable that can be read or written from all POUs (programs, functions, and function blocks).
I/O map settings	Settings that assign variables to I/O ports. Assignment information between I/O ports and variables.
I/O port	A logical interface that is used by the CPU Unit to exchange data with an external device (slave or Unit).
I/O refreshing	Cyclic data exchange with external devices that is performed with predetermined memory addresses.
information	One of the event levels for Controller events or user-defined events. These are not errors, but appear in the event log to notify the user of specific information.
Initial Value	One of the attributes of a variable. The variable is set to the initial value in the following situations. • When power is turned ON • When the CPU Unit changes to RUN mode • When you specify to initialize the values when the user program is transferred • When a major fault level Controller error occurs
inline ST	ST programming that is included within a ladder diagram program.
instruction	The smallest unit of the processing elements that are provided by OMRON for use in POU algorithms. There are ladder diagram instructions (program inputs and outputs), function instructions, function block instructions, and ST statements.
literal	A constant expression that is used in a user program.
local variable	A variable that can be accessed only from inside the POU in which it is defined. “Local variable” is used as opposed to “global variable.” Local variables include internal variables, input variables, output variables, in-out variables, and external variables.

Term	Description
main memory	The memory inside the CPU Unit that is used by the CPU Unit to execute the OS and user program.
major fault level Controller error	An error for which all NJ/NX-series Controller control operations stop. The CPU Unit immediately stops user program execution and turns OFF the loads for all slaves and Units (including remote I/O).
MC Test Run	A function to check motor operation and wiring from the Sysmac Studio.
memory used for CJ-series Units	One type of I/O memory in an NJ-series CPU Unit. It contains addresses that can be directly specified by the user. It can be accessed only with variables with an AT attribute. This memory is used to access CJ-series Units and CJ-series networks.
minor fault level Controller error	An error for which part of the control operations for one of the function modules in the NJ/NX-series Controller stop. An NJ/NX-series CPU Unit continues operation even after a minor fault level Controller error occurs.
Motion Control Function Module	One of the function modules. The MC Function Module performs motion control based on commands from the motion control instructions that are executed in the user program.
motion control instruction	A function block instruction that executes motion control. The Motion Control Function Module supports instructions that are based on function blocks for PLCopen® motion control as well as instructions developed specifically for the Motion Control Function Module.
namespace	A system that is used to group and nest the names of functions, function block definitions, and data types.
Network Publish	One of the attributes of a variable. This attribute allows you to use CIP message communications or tag data links to read/write variables from another Controller or from a host computer.
NX Units	Any of the NX-series Units that perform I/O processing with connected external devices. The Communications Coupler Units are not included with the NX Units.
observation	One of the event levels for Controller events or user-defined events. These are minor errors that do not affect control operations, but appear in the event log to notify the user of specific information.
partial fault level Controller error	An error for which all of the control operations for one of the function modules in the NJ/NX-series Controller stop. An NJ/NX-series CPU Unit continues operation even after a partial fault level Controller error.
PDO communications	An abbreviation for process data communications. Data is exchanged between the master and slaves on a process data communications cycle. (The process data communications cycle is the same as the task period of the primary periodic task.)
periodic task	A task for which user program execution and I/O refreshing are performed each period.
PLC Function Module	One of the function modules. This function module executes the user program, sends commands to the Motion Control Function Module, and provides an interface to the USB and SD Memory Card.
POU	An acronym for "program organization unit." A POU is a unit in a program execution model that is defined in IEC 61131-3. A POU contains an algorithm and a local variable table and forms the basic unit used to build a user program. There are three types of POUs: programs, functions, and function blocks.
primary periodic task	The task with the highest priority.
process data communications	One type of EtherCAT communications in which process data objects (PDOs) are used to exchange information cyclically and in realtime. Process data communications are also called PDO communications.
program	Along with functions and function blocks, one of the three types of POUs. Programs are assigned to tasks to execute them.
Range Specification	One of the variable specifications. You can specify a range for a variable in advance. The variable can take only values that are in the specified range.

Term	Description
Retain	One of the attributes of a variable. The values of variables with a Retain attribute are held at the following times. (Variables without a Retain attribute are set to their initial values.) • When power is turned ON after a power interruption • When the CPU Unit changes to RUN mode • When you specify to not initialize the values when the user program is transferred
SDO communications	One type of EtherCAT communications in which service data objects (SDOs) are used to transmit information whenever required.
Servo Drive/encoder input slave	Any of the EtherCAT slaves that is assigned to an axis. In the NJ/NX-series System, it would be a Servo Drive or Encoder Input Slave Unit.
slave	A device that performs remote I/O for a master.
slave and Unit configurations	A generic term for the EtherCAT configuration and Unit configuration.
Slave Terminal	A building-block remote I/O terminal to which a Communications Coupler Unit and NX Units are mounted. A Slave Terminal is one type of slave.
Special Unit Setup	A generic term for the settings for a Special Unit, including the settings in allocated DM Area words.
structure	One of the derivative data types. It consists of multiple data types placed together into a layered structure.
synchronization	A function that automatically compares the information in the NJ/NX-series Controller with the information in the Sysmac Studio, displays any differences and locations in a hierarchical form, and can be used to synchronize the information.
Sysmac Studio	A computer software application for setting, programming, debugging, and troubleshooting NJ/NX-series Controllers. It also provides operations for motion control and a Simulator.
system common processing	System processing that is performed by the CPU Unit to perform I/O refreshing and the user program execution within a task. Exclusive control of variables between tasks, data trace processing, and other processing is performed.
system service	Processing that is performed by the CPU Unit in unused time between task processing. The system service includes communications processing, SD Memory Card access processing, self-diagnosis processing, and other processing.
system-defined variable	A variable for which all attributes are defined by the system and cannot be changed by the user.
task	An attribute that defines when a program is executed.
task period	The interval at which the primary periodic task or a periodic task is executed.
union	One of the derivative data types. It allows you to handle the same data as different data types.
Unit	A device that mounts to the CPU Rack or an Expansion Rack. In NJ/NX-series manuals, it refers to CJ-series Units.
Unit configuration	The configuration information for the Units that are set on the Sysmac Studio. This information tells what Unit models are connected to the CPU Unit and where they are connected.
upload	To transfer data from the Controller to the Sysmac Studio with the synchronization operation of the Sysmac Studio.
user program	All of the programs in one project.
user-defined event	One of the events in the NJ/NX-series System. These events are defined by the user. “User-defined events” is a generic term for user-defined errors and user-defined information.
user-defined variable	A variable for which all of the attributes are defined by the user and can be changed by the user.
variable	A representation of data, such as a numeric value or character string, that is used in a user program. You can change the value of a variable by assigned the required value. “Variable” is used as opposed to “constant,” for which the value does not change.
variable memory	A memory area that contains the present values of variables that do not have AT specifications. It can be accessed only with variables without an AT attribute.

Revision History

A manual revision code appears as a suffix to the catalog number on the front and back covers of the manual.

Cat. No. W501-E1-12

Revision code

Revision code	Date	Revised content
01	July 2011	Original production
02	March 2012	• Added information on the NJ301-□□□□. • Added information on the functions supported by unit version 1.01 of the CPU Units. • Corrected mistakes.
03	May 2012	• Made changes accompanying release of unit version 1.02 of the CPU Unit. • Corrected mistakes.
04	August 2012	• Added information on the functions supported by unit version 1.03 of the CPU Units. • Corrected mistakes.
05	February 2013	• Made changes accompanying release of unit version 1.04 of the CPU Unit. • Corrected mistakes.
06	April 2013	• Added information on the functions supported by unit version 1.05 of the CPU Units. • Added information on the NX Series. • Corrected mistakes.
07	June 2013	• Added information on the functions supported by unit version 1.06 of the CPU Units. • Corrected mistakes.
08	September 2013	• Added information on the functions supported by unit version 1.07 of the CPU Units. • Corrected mistakes.
09	December 2013	• Added information on the functions supported by unit version 1.08 of the CPU Units. • Corrected mistakes.
10	July 2014	• Corrected mistakes.
11	January 2015	• Added information on the functions supported by unit version 1.10 of the CPU Units. • Corrected mistakes.
12	April 2015	• Added information on the NX701-□□□□ and NJ101-□□□□. • Corrected mistakes.

1

Introduction to NJ/NX-series Controllers

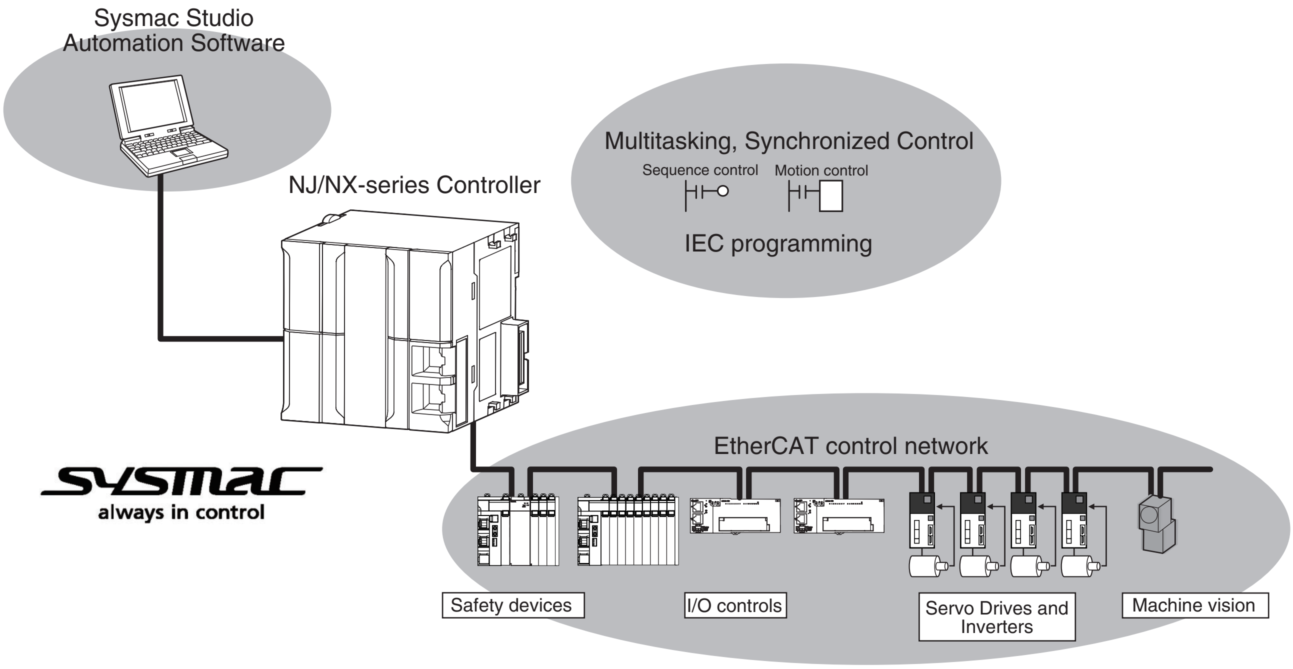
This section describes the features, basic system configuration, specifications, and overall operating procedure of an NJ/NX-series Controller.

1-1 The NJ/NX-series Controllers

The SYSMAC NJ/NX-series Controllers are next-generation machine automation controllers that provide the functionality and high-speed performance that are required for machine control.
They provide the safety, reliability, and maintainability that are required of industrial controllers.

The NJ/NX-series Controllers provide the functionality of previous OMRON PLCs, and they also provide the functionality that is required for motion control. Synchronized control of I/O devices on high-speed EtherCAT can be applied to safety devices, vision systems, motion equipment, discrete I/O, and more.

OMRON offers the new Sysmac Series of control devices designed with unified communications specifications and user interface specifications. The NJ/NX-series Machine Automation Controllers are part of the Sysmac Series. You can use them together with EtherCAT slaves, other Sysmac products, and the Sysmac Studio Automation Software to achieve optimum functionality and ease of operation. With a system that is created from Sysmac products, you can connect components and operate the system through unified concepts and usability.



1-1-1 Features

Hardware Features

- **Standard-feature EtherCAT Control Network Support**

All CPU Units provide an EtherCAT master port for EtherCAT communications. EtherCAT is an advanced industrial network system that achieves faster, more-efficient communications. It is based on Ethernet. Each node achieves a short fixed communications cycle time by transmitting Ethernet frames at high speed. The standard-feature EtherCAT control network allows you to connect all of the devices required for machine control (e.g., I/O systems, Servo Drives, Inverters, and machine vision) to the same network.

- **Support for EtherCAT Slave Terminals**

You can use EtherCAT Slave Terminals to save space. You can also flexibly build systems with the wide variety of NX Units.

- **Achieving a Safety Subsystem on EtherCAT**

You can use NX-series Safety Control Units to integrate safety controls in a sequence and motion control system as a subsystem on EtherCAT.

Version Information

A CPU Unit with unit version 1.06 or later and Sysmac Studio version 1.07 or higher are required to use the NX-series Safety Control Units.

- **CJ-series Units**

NJ-series CPU Units allow you to mount CJ-series Basic I/O Units and Special Units on the I/O bus, in addition to EtherCAT network slaves.

- **Standard-feature EtherNet/IP Communications Port**

All CPU Units provide an EtherNet/IP port for EtherNet/IP communications. EtherNet/IP is a multi-vendor industrial network that uses Ethernet. You can use it for networks between Controllers or as a field network. The use of standard Ethernet technology allows you to connect to many different types of general-purpose Ethernet devices.

- **Standard-feature USB Port**

You can connect a computer that runs the Support Software directly to the CPU Unit with a USB connection.

- **Standard-feature SD Memory Card Slot**

You can access an SD Memory Card that is mounted in the CPU Unit from the user program.

- **Highly Reliable Hardware**

The NJ-series Controllers provide the hardware reliability and RAS functions that you expect of a PLC.

- **Parallel Execution of Tasks with a Multi-core Processor**

The NX701-□□□□ CPU Unit has a multi-core processor that can execute more than one task in parallel. This enables high-speed control of even large-scale devices.

Software Features

- **Integrated Sequence Control and Motion Control**

An NJ-series CPU Unit can perform both sequence control and motion control. You can simultaneously achieve both sequence control and multi-axes synchronized control. Sequence control, motion control, and I/O refreshing are all executed in the same control period. The same control period is also used for the process data communications cycle for EtherCAT. This enables precise sequence and motion control in a fixed period with very little deviation.

- **Multitasking**

You assign I/O refreshing and programs to tasks and then specify execution conditions and execution order for them to flexibly combine controls that suit the application.

● Programming Languages Based on the IEC 61131-3 International Standard

The NJ-series Controllers support language specifications that are based on IEC 61131-3. To these, OMRON has added our own improvements. Motion control instructions that are based on PLCopen® standards and an instruction set (POUs) that follows IEC rules are provided.

● Programming with Variables to Eliminate Worrying about the Memory Map

You access all data through variables in the same way as for the advanced programming languages that are used on computers. Memory in the CPU Unit is automatically assigned to the variables that you create so that you do not have to remember the physical addresses.

● A Wealth of Security Features

The many security features of the NJ-series Controllers include operation authority settings and restriction of program execution with IDs.

● Complete Controller Monitoring

The CPU Unit monitors events in all parts of the Controller, including mounted Units and EtherCAT slaves. Troubleshooting information for errors is displayed on the Sysmac Studio or on an NS-series PT. Events are also recorded in logs.

● Sysmac Studio Automation Software

The Sysmac Studio provides an integrated development environment that covers not only the Controller, but also covers peripheral devices and devices on EtherCAT. You can use consistent procedures for all devices regardless of the differences in the devices. The Sysmac Studio supports all phases of Controller application, from designing through debugging, simulations, commissioning, and changes during operation.

● A Wealth of Simulation Features

The many simulation features include execution, debugging, and task execution time estimates on a virtual controller.

1-1-2 Introduction to the System Configurations

This section describes the system configurations of the NX-series and NJ-series Controllers.

Introduction to the System Configurations of the NX-series Controllers

The NX Series supports the following system configurations.

● Basic System Configurations

The NX-series basic configurations include the EtherCAT network configuration and the Support Software.

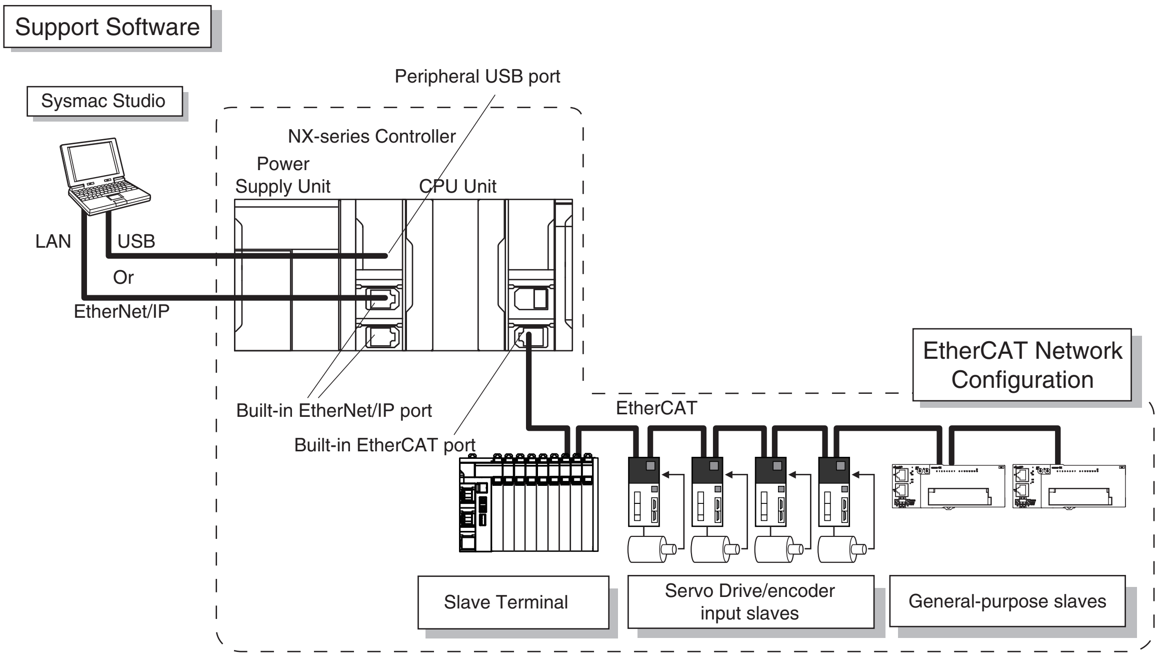
- EtherCAT Network Configuration

  You can use the built-in EtherCAT port to connect to EtherCAT Slave Terminals, to general-purpose slaves for analog and digital I/O, and to Servo Drives and encoder input slaves. An EtherCAT network configuration enables precise sequence and motion control in a fixed cycle with very little deviation.

- Support Software

  The Support Software is connected to the peripheral USB port on the CPU Unit with a commercially available USB cable. You can also connect it through an Ethernet cable that is connected to the built-in EtherNet/IP port.

  Refer to *10-2 Connection with Sysmac Studio* for details on the connection configuration of the Support Software.



Precautions for Correct Use

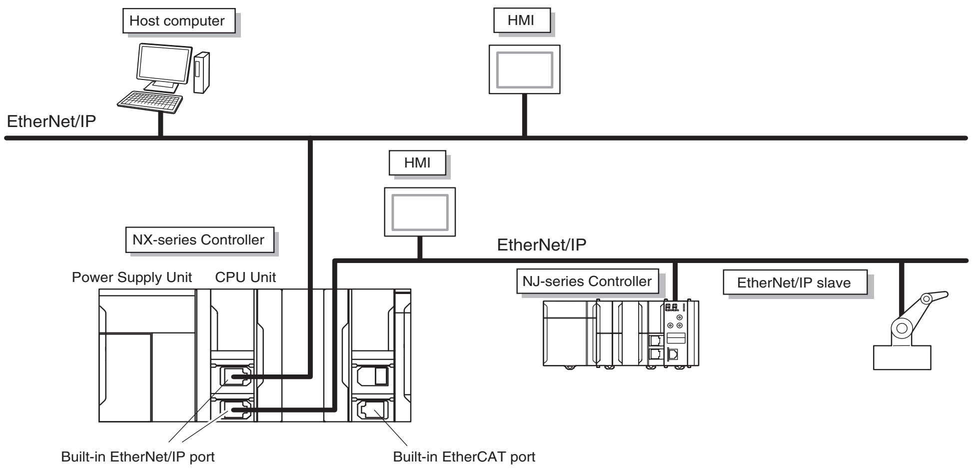
NX Units should be connected to Slave Terminals. The NX bus connector of the CPU Unit is provided for future expansion so that it cannot be used to connect any NX Unit.

Additional Information

You can connect the Sysmac Studio directly to the Communications Coupler Unit to set up the Slave Terminal. Refer to the *NX-series EtherCAT Coupler Units User's Manual* (Cat. No. W519) for details.

● Network Configurations

- Host computers, HMIs, and other NJ/NX-series Controllers are connected to the built-in EtherNet/IP port.
- An NX-series CPU Unit has two EtherNet/IP ports.



Refer to *Section 10 Communications Setup* for details on the network configuration.

● Support Software

You can use the following Support Software to set up, monitor, and debug an NX-series Controller.

- Sysmac Studio

  The Sysmac Studio is the main Support Software that you use for an NX-series Controller. On it, you can set up the Controller configurations, parameters, and programs, and you can debug and simulate operation.

- Other Support Software

  The following Support Software is also included in the Sysmac Studio Software Package Standard Edition.

Configuration software	Application
Sysmac Studio	The Sysmac Studio is used for sequence control, motion control, and all other operations except those described below.
Network Configurator	The Network Configurator is used for tag data links on EtherNet/IP ports or Units.*1
CX-Integrator	The CX-Integrator is used for remote I/O communications with a DeviceNet Unit or CompoNet Master Unit.
CX-Protocol	The CX-Protocol is used for protocol macros with Serial Communications Units.
CX-Designer	The CX-Designer is used to create screens for NS-series PTs.

*1 If the NJ/NX-series Controller is a target device, you may also use Sysmac Studio version 1.10 or higher. Use the Network Configurator if a CS/CJ-series PLC operates as the originator device.

Introduction to the System Configurations of the NJ-series Controllers

The NJ Series supports the following system configurations.

● Basic System Configurations

The NJ-series basic configurations include the EtherCAT network configuration, CJ-series Unit configuration, and the Support Software.

EtherCAT Network Configuration

You can use the built-in EtherCAT port to connect to EtherCAT Slave Terminals, to general-purpose slaves for analog and digital I/O, and to Servo Drives and encoder input slaves. An EtherCAT network configuration enables precise sequence and motion control in a fixed cycle with very little deviation.

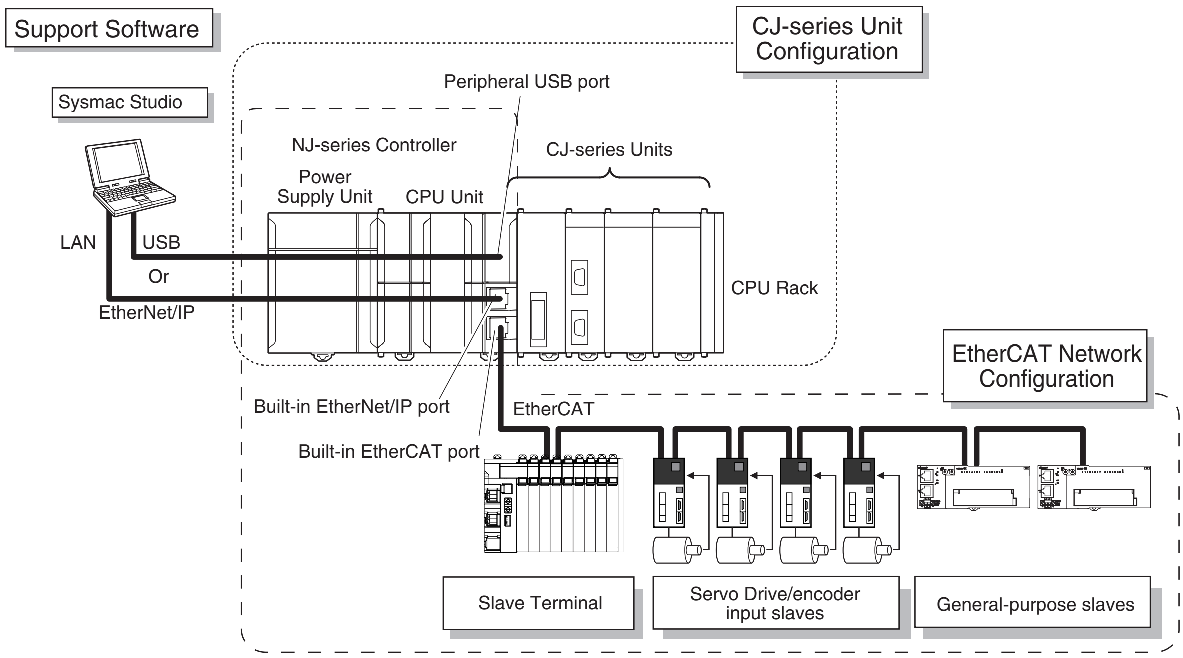
CJ-series Unit Configuration

In addition to the EtherCAT network, you can mount CJ-series Basic I/O Units and Special Units. CJ-series Units can be mounted both to the CPU Rack where the CPU Unit is mounted and to Expansion Racks.

Support Software

The Support Software is connected to the peripheral USB port on the CPU Unit with a commercially available USB cable. You can also connect it through an Ethernet cable that is connected to the built-in EtherNet/IP port.

Refer to *10-2 Connection with Sysmac Studio* for details on the connection configuration of the Support Software.

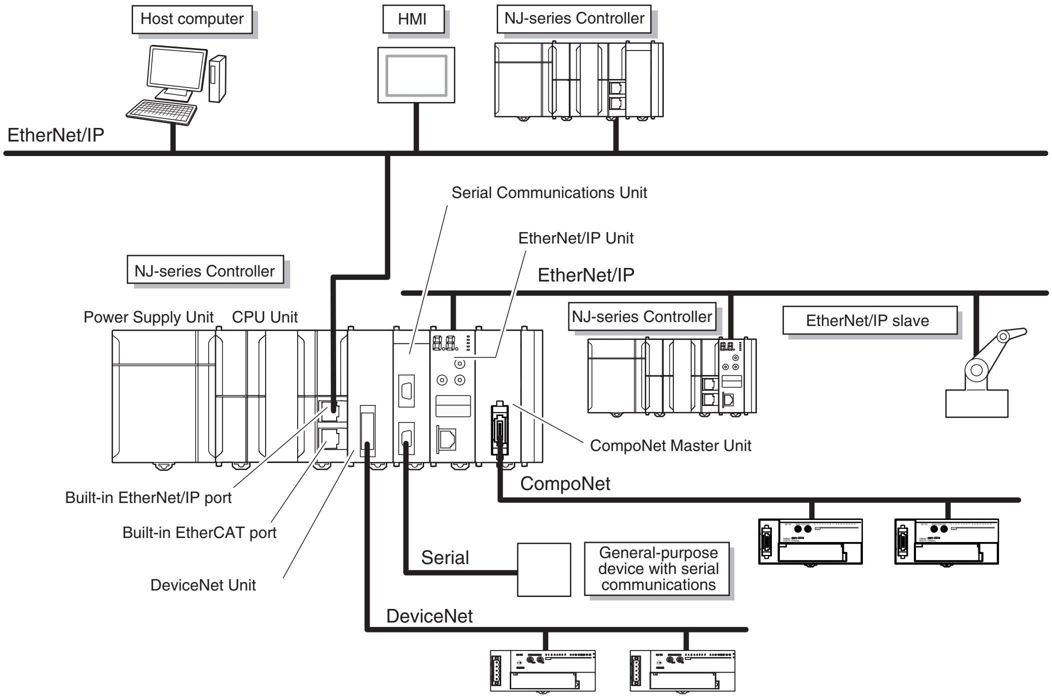


Additional Information

You can connect the Sysmac Studio directly to the Communications Coupler Unit to set up the Slave Terminal. Refer to the *NX-series EtherCAT Coupler Units User's Manual* (Cat. No. W519) for details.

Network Configurations

- Host computers, HMIs, and other NJ-series Controllers are connected to the built-in EtherNet/IP port or to a CJ1W-EIP21 EtherNet/IP Unit.
- A DeviceNet network is connected to a DeviceNet Unit. A CompoNet network is connected to a CompNet Unit. A serial communications network is connected to a Serial Communications Unit.



Refer to *Section 10 Communications Setup* for details on the network configuration.

Support Software

You can use the following Support Software to set up, monitor, and debug an NJ-series Controller.

- Sysmac Studio

  The Sysmac Studio is the main Support Software that you use for an NJ-series Controller. On it, you can set up the Controller configurations, parameters, and programs, and you can debug and simulate operation.

- Other Support Software

  The following Support Software is also included in the Sysmac Studio Software Package Standard Edition.

Configuration software	Application
Sysmac Studio	The Sysmac Studio is used for sequence control, motion control, and all other operations except those described below.
Network Configurator	The Network Configurator is used for tag data links on EtherNet/IP ports or Units.*1
CX-Integrator	The CX-Integrator is used for remote I/O communications with a DeviceNet Unit or CompoNet Master Unit.
CX-Protocol	The CX-Protocol is used for protocol macros with Serial Communications Units.
CX-Designer	The CX-Designer is used to create screens for NS-series PTs.

*1 If the NJ/NX-series Controller is a target device, you may also use Sysmac Studio version 1.10 or higher. Use the Network Configurator if a CS/CJ-series PLC operates as the originator device.

1-2 Main Specifications

This section gives the main specifications of the NJ/NX-series Controllers. Refer to *A-1 Specifications* for general specifications, performance specifications, and function specifications.

Item				NX701-		NJ501-			NJ301-		NJ101-	
				17□□	16□□	15□□	14□□	13□□	12□□	11□□	10□□	90□□
Program-ming	Program capacity*1	Size		80 MB		20 MB			5 MB		3 MB	
		Quantity	Number of POU definitions	6,000		3,000			750		450	
			Number of POU Instances	48,000		9,000 (*)			3,000 (*)		1,800	
	Memory capacity for variables	Retain attributes*2	Size	4 MB		2 MB			0.5 MB			
			Number of variables	40,000		10,000			5,000 (*)			
		No Retain attributes*3	Size	256 MB		4 MB			2 MB			
			Number of variables	360,000		90,000			22,500			
	Data types	Number of data types		8,000		2,000			1,000			
	Memory for CJ-series Units (Can be specified with AT specifications for variables.)	CIO Area		---		6,144 words (CIO 0 to CIO 6143)						
		Work Area		---		512 words (W0 to W511)						
		Holding Area		---		1,536 words (H0 to H1535)						
		DM Area		---		32,768 words (D0 to D32767)						
		EM Area		---		32,768 words × 25 banks (E0_00000 to E18_32767)			32,768 words × 4 banks (E0_00000 to E3_32767)			
Motion control	Number of controlled axes	Maximum number of controlled axes*4		256 axes	128 axes	64 axes	32 axes	16 axes	15 axes (*)		6 axes	---
		Maximum number of used real axes*5		256 axes	128 axes	64 axes	32 axes	16 axes	8 axes	4 axes	2 axes	---
		Maximum number of axes for single-axis control		256 axes	128 axes	64 axes	32 axes	16 axes	15 axes (*)		6 axes	---
		Maximum number of axes for linear interpolation axis control		4 axes per axes group								---
		Number of axes for circular interpolation axis control		2 axes per axes group								---
	Maximum number of axes groups			64 axes groups		32 axes groups						---
	Motion control period			The same control period as that is used for the process data communications cycle for EtherCAT.								---
	Cams	Number of cam data points	Maximum points per cam table	65,535 points								---
			Maximum points for all cam tables	1,048,560 points					262,140 points			---
		Maximum number of cam tables		640 tables					160 tables			---
Peripheral USB port	Supported services			Sysmac Studio connection								
	Physical layer			USB 2.0-compliant B-type connector								
	Transmission distance			5 m max.								

Item			NX701-		NJ501-			NJ301-		NJ101-	
			17□□	16□□	15□□	14□□	13□□	12□□	11□□	10□□	90□□
Built-in EtherNet/IP port	Number of ports		2		1						
	Physical layer		10BASE-T, 100BASE-TX, or 1000BASE-T		10BASE-T or 100BASE-TX						
	Frame length		1,514 bytes max.								
	Media access method		CSMA/CD								
	Modulation		Baseband								
	Topology		Star								
	Baud rate		1 Gbps (1000BASE-T)		100 Mbps (100BASE-TX)						
	Transmission media		STP (shielded, twisted-pair) cable of Ethernet category 5, 5e or higher								
	Maximum transmission distance between Ethernet switch and node		100 m								
	Maximum number of cascade connections		There are no restrictions if an Ethernet switch is used.								
	CIP service: Tag data links (cyclic communications)	Maximum number of connections	256 per port 512 total		32						
		Packet interval*6	Can be set for each connection. 0.5 to 10,000 ms in 0.5-ms increments		Can be set for each connection. 1 to 10,000 ms in 1-ms increments (*)						
		Permissible communications band	40,000 pps*7 (including heartbeat)		3,000 pps*7 (including heartbeat)(*)						
		Maximum number of tag sets	256 per port 512 total		32						
		Tag types	Network variables		Network variables, CIO, Work, Holding, DM, and EM Areas						
		Number of tags per connection (= 1 tag set)	8 (7 tags if Controller status is included in the tag set.)								
		Maximum number of tags	256 per port 512 total		256						
		Maximum link data size per node (total size for all tags)	369,664 bytes		19,200 bytes						
		Maximum data size per connection	1,444 bytes		600 bytes						
		Maximum number of registrable tag sets	256 per port 512 total (1 connection = 1 tag set)		32 (1 connection = 1 tag set)						
		Maximum tag set size	1,444 bytes (Two bytes are used if Controller status is included in the tag set.)		600 bytes (Two bytes are used if Controller status is included in the tag set.)						
		Multi-cast packet filter*8	Supported.								
Built-in EtherCAT port	Communications standard		IEC 61158 Type12								
	EtherCAT master specifications		Class B (Feature Pack Motion Control compliant)								
	Physical layer		100BASE-TX								
	Modulation		Baseband								
	Baud rate		100 Mbps (100BASE-TX)								
	Duplex mode		Auto								
	Topology		Line, daisy chain, and branching								
	Transmission media		Twisted-pair cable of category 5 or higher (double-shielded straight cable with aluminum tape and braiding)								
	Maximum transmission distance between nodes		100 m								
	Maximum number of slaves		512		192					64	

Item				NX701-		NJ501-			NJ301-		NJ101-	
				17□□	16□□	15□□	14□□	13□□	12□□	11□□	10□□	90□□
Unit configuration	Maximum number of connectable Units	Maximum number of CJ Units per CPU Rack or Expansion Rack		---		10						
		Maximum number of CJ Units for entire controller		---		40						
		Maximum number of NX Units for entire controller		4,000 (On EtherCAT Slave Terminals)								
	Maximum number of Expansion Racks			0		3						
	I/O capacity	Maximum number of I/O points on CJ-series Units		---		2,560						
	Power Supply Unit for CPU Rack and Expansion Racks	Model		NX-PA9001 NX-PD7001		NJ-P□3001						
		Power OFF detection time	AC power supply	30 to 45 ms								
			DC power supply	5 to 20 ms		22 to 25 ms						

*1 This is the capacity for the execution objects and variable tables (including variable names).

*2 Does not include Holding, DM, and EM Area memory for CJ-series Units.

*3 Does not include CIO and Work Area memory for CJ-series Units.

*4 This is the total for all axis types.

*5 This is the total number of axes that are set as servo axes or encoder axes and are also set as used axes.

*6 Data will be refreshed at the set interval, regardless of the number of nodes.

*7 “pps” means packets per second, i.e., the number of communications packets that can be sent or received in one second.

*8 As the EtherNet/IP port implements the IGMP client, unnecessary multi-cast packets can be filtered by using an Ethernet switch that supports IGMP Snooping.

Note Items that are marked with asterisks in the table are improvements that were made during version upgrades. Refer to *A-13 Version Information for NX-series Controllers* and *A-14 Version Information for NJ-series Controllers* for information on version upgrades.

1-3 Overall Operating Procedure for the NJ/NX-series Controller

This section gives the overall operating procedure of the NJ/NX-series Controllers and then describes it in more detail.

1-3-1 Overall Procedure

The overall procedure to use an NJ/NX-series Controller is given below.

Step 1. Software Design

Design the overall system configuration, task configuration, programs, and variables.

- Step 1-1 Designing I/O and Processing
- Step 1-2 Designing Tasks
- Step 1-3 Designing Programs

↓

Step 2. Software Setups and Programming

Create the system configurations that you designed in step 1 on the Support Software and assign the variables. Create the tasks and programs, and debug them, e.g., with simulations.

- Step 2-1 Slave and Unit Configurations
- Step 2-2 Controller Setup
- Step 2-3 Programming
- Step 2-4 Offline Debugging

↓

Step 3. Mounting and Setting Hardware

Mount the Units and make the required hardware settings.

↓

Step 4. Wiring

Connect the network cables and wire the I/O.

↓

Step 5. Checking Operation and Starting Actual Operation on the Actual System

Connect the Support Software to the physical system and download the project. Check operation on the physical system and then start actual system operation.

1-3-2 Procedure Details

Step 1. Software Design

Step	Description	Reference
Step 1-1 Designing I/O and Processing	• External I/O devices and unit configuration • Refresh periods for external devices • Program contents	*NX-series CPU Unit Hardware User's Manual* (Cat No. W535) *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500)

Step 1-2 Designing Tasks	• Task configuration • Relationship between tasks and programs • Task periods • Slave and Unit refresh times • Exclusive control methods for variables between tasks	*4-2-3 Task Settings*

Step 1-3 Designing Programs		
POU (Program Organization Unit) Design	• Programs • Functions and function blocks • Determining the algorithm languages	*Section 6 Programming*
Variable Design	• Defining variables that you can use in more than one POU and variables that you use in only specific POUs • Defining the variables names for the device variables that you use to access slaves and Units • Defining the attributes of variables, such as the Name and Retain attributes • Designing the data types of variables	*6-3 Variables*

Step 2. Software Setups and Programming

Step	Description	Sysmac Studio Operations	Reference
Project Creation	1. Create a project in the Sysmac Studio. 2. Insert a Controller.	**New Project** Button ***Insert*** – ***Controller***	*Sysmac Studio Version 1 Operation Manual* (Cat. No. W504)

The following *Controller Configurations and Setup* and the *Programming and Task Settings* can be performed in either order.

Step 2-1 Slave and Unit Configurations			
1) Creating the Slave and Unit Configurations	1. Creating the slave configuration and Unit configuration either offline or online. (For online configuration, make the online connection that is described in step 5.) 2. Setting up any Slave Terminals that are used.	EtherCAT Slave Setting Editor Unit Editor	*3-2-1 Creating the EtherCAT Slave Configuration* *3-2-2 Creating the Unit Configuration* *NX-series EtherCAT Coupler Unit User's Manual* (Cat. No. W519)

2) Assigning Device Variables to I/O Ports	Registering device variables in variable tables (Variable names are user defined or automatically created.)	I/O Map	*3-3 I/O Ports and Device Variables*

(The following step is for motion control.)

3) Creating the Axes and Assigning Them to the Servo Drive/Encoder Input Slaves	Creating the axes and setting them as real axes or virtual axes. Creating axes groups to perform interpolated axes control.	**Configurations and Setup – Motion Control Setup**	*3-5 Creating the Axes and Assigning Them to the Servo Drives/Encoder Input Slaves*

Step 2-2 Controller Setup	Setting the following parameters from the Sysmac Studio		*Section 4 Controller Setup*
	Setting the initial values for the PLC Function Module	**Configurations and Setup – Controller Setup – Operation Settings**	*4-2 Initial Settings for the PLC Function Module*
	(For NJ-series CPU Units) Initial settings for Special Units	**Configurations and Setup – CPU/Expansion Racks**	*4-3 Initial Settings for Special Units*
	(To use motion control) Setting the initial settings for the Motion Control Function Module	**Configurations and Setup – Motion Control Setup**	*4-4 Initial Settings for the Motion Control Function Module*
	Setting the initial values for the EtherCAT Function Module	**Configurations and Setup – EtherCAT**	*4-5 Initial Settings for the EtherCAT Master Function Module*
	Setting the initial values for the EtherNet/IP Function Module	**Configurations and Setup – Controller Setup – Built-in EtherNet/IP Port Settings**	*4-6 Initial Settings for the EtherNet/IP Function Module*

Step 2-3 Programming			
1) Registering Variables	• Registering the variables used by more than one POU in the global variable table with Sysmac Studio • Registering the local variable table for each program • Registering the local variable table for each function block and function	Global Variable Table Editor Local Variable Table Editor	*Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) *6-3 Variables*
2) Writing Algorithms for POUs	Writing the algorithms for the POUs (programs, function blocks, and functions) in the required languages	Programming Editor	*Section 6 Programming* *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) and *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508)
3) Setting the Tasks	Making task settings	**Configurations and Setup** – **Task Settings**	*4-2-3 Task Settings*

Step 2-4 Offline Debugging	Checking the algorithms and task execution times on the Simulator (virtual controller)		*Section 7 Checking Operation and Actual Operation*

Step 3. Mounting and Setting Hardware

Step	Description	Reference
1. Mounting	• Connecting adjacent Units • Mounting to DIN Track	*NX-series CPU Unit Hardware User's Manual* (Cat No. W535) *NJ-series CPU Unit Hardware User's Manual* (Cat No. W500)
2. Setting Hardware	• Setting the node addresses of the EtherCAT slaves • Setting unit numbers on the rotary switches on the front of the Special Units	Operation manuals for the EtherCAT slaves and Special Units

Step 4. Wiring

Step	Description	Reference
1. Connecting Ethernet Cable	• Connecting the built-in EtherCAT port • Connecting the built-in EtherNet/IP port	*NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500)
2. Wiring I/O	• Wiring I/O to EtherCAT slaves • Wiring Basic I/O Units and Special Units	Operation manuals for the EtherCAT slaves and *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500)
	• Checking wiring	*Sysmac Studio Version 1 Operation Manual* (Cat. No. W504)
3. Connecting the Computer That Runs the Sysmac Studio	• Connecting USB Cable • Connecting the built-in EtherNet/IP port	*Sysmac Studio Version 1 Operation Manual* (Cat. No. W504)

Step 5. Checking Operation and Starting Operation on the Actual System

Step	Description	Sysmac Studio Operations	Reference
1. Online Connection to Sysmac Studio and Project Download	Turn ON the power supply to the Controller and place the Sysmac Studio online. Then, download the project.* (Perform this step before you create the slave configuration or Unit configuration from the mounted Units in step 2-1.)	***Controller*** – ***Communications Setup*** ***Controller*** – ***Synchronization***	*Section 7 Checking Operation and Actual Operation*
2. Operation Check on Controller	1. Check the wiring by using forced refreshing of real I/O from the I/O Map or Watch Tab Page. 2. For motion control, use the MC Test Run operations in PROGRAM mode to check the wiring. Then check the motor rotation directions for jogging, travel distances for relative positioning (e.g., for electronic gear settings), and homing operation. 3. Change the Controller to RUN mode and check the operation of the user program.		*Section 7 Checking Operation and Actual Operation*
3. Actual Controller Operation	Start actual operation.		

* Use the Synchronize Menu of the Sysmac Studio to download the project.

2

CPU Unit Operation

This section provides information that is necessary to use the CPU Unit, including how the CPU Unit works and the operations that it performs depending on the status of the CPU Unit.

2-1 Overview of CPU Unit Operation

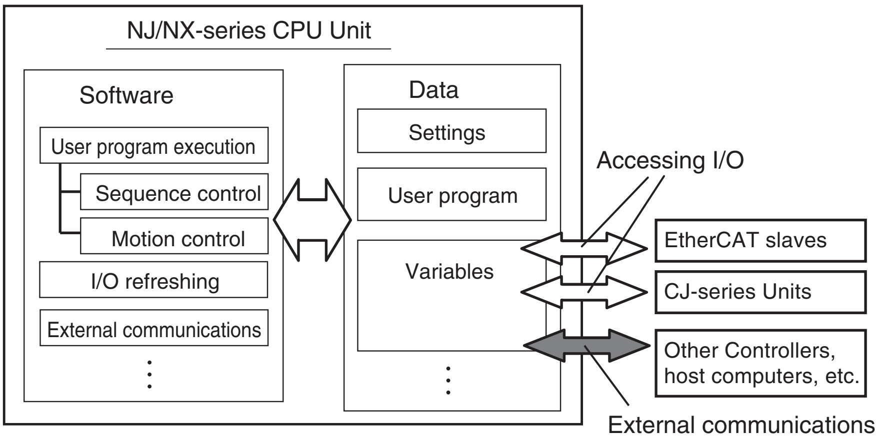
This section describes the operation of the CPU Unit and gives an overview of how it operates depending on the status of the CPU Unit.

2-1-1 Introduction to CPU Unit

The NJ/NX-series CPU Unit executes the user program for sequence control and motion control. It also performs other processing, such as I/O refreshing and external communications. These processes are performed by the software in the CPU Unit.

The CPU Unit also contains settings, the user program, variables, and other data. The CPU Unit uses this data to perform processing. Of this data, variables are used to access the CPU Unit and I/O, and for external communications.

The internal software and the use of variables for I/O access enable the CPU Unit to execute both sequence control and motion control.



Note You can use CJ-series Units only with NJ-series CPU Units.

This section describes the following items to provide you with a basic understanding of how the CPU Unit performs sequence control and motion control.

Item	Reference
Software	P.2-4
Accessing I/O	P.2-10
Sequence control and motion control	P.2-18
Overview of CPU Unit data	P.2-24
Operation for CPU Unit status	P.2-26

Additional Information

Refer to the following manuals for details on the use of variables for external communications.

Communications	Manual
EtherNet/IP	• For information on using variables with the built-in EtherNet/IP port, refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506). • For information on using variables with an EtherNet/IP Unit, refer to the *CJ-series EtherNet/IP Units Operation Manual for NJ-series CPU Unit* (Cat. No. W495).
Serial communications	*CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit* (Cat. No. W494)

2-1-2 Overview of Operation According to CPU Unit Status

The status of the CPU Unit changes when an error occurs or when you change the operating mode. Changes in the status of the CPU Unit affect user program execution, I/O refreshing, and the processing of external communications.

2-6 Operation for CPU Unit Status describes CPU Unit operation according to the status of the CPU Unit.

2-2 Software

This section describes the software configuration of the CPU Unit, and how the software components operate.

2-2-1 Software Configuration

The software in the CPU Unit is divided into four modules. These functional units are called function modules.

The function modules and the processing that they perform are described in the following table.

Function module name	Processing
PLC Function Module	Performs the following services: task scheduling, commands for other function modules, event logging, execution of the user program, I/O refreshing*1 for the CJ-series Units, USB port services, SD Memory Card services, and data trace processing.
Motion Control Function Module	Performs motion control processing.*2
EtherCAT Master Function Module	Performs communications with EtherCAT slaves as the EtherCAT master, including I/O refreshing*3 for the EtherCAT slaves, EtherCAT message communications*4, etc.
EtherNet/IP Function Module	Performs processing for EtherNet/IP communications, including tag data link processing, built-in EtherNet/IP port servicing, etc.

*1 Some CJ-series Units can also be connected to an NJ-series CPU Unit.

*2 This function module executes motion processing based on target values (such as the position or velocity target value) from the motion control instructions. It outputs command values, controls status, and obtains information through the EtherCAT Master Function Module.

*3 I/O refreshing for EtherCAT slaves is performed by using process data communications (also called PDO communications). In PDO communications, the master and slaves exchange data cyclically at regular intervals.

*4 This module communicates with the EtherCAT slaves as the EtherCAT master.

2-2-2 Operation of Software

The software in the CPU Unit performs the following four processes. Which process is performed depends on the status of the CPU Unit and the execution conditions of the process itself.

Type of CPU Unit processing	Status of CPU Unit	Execution conditions	Processing example
Initialization	Startup state	Initialization is performed only when the power supply is turned ON.	Self diagnosis at startup
Processing executed with tasks	Normal operation and error states	The processing is executed within the assigned task. These tasks are executed either periodically or only once when the specified condition is met.	User program execution and I/O refreshing
Tag data link service		These services are performed only upon requests from the hardware or other external devices.	Tag data links
System services			USB port service and SD Memory Card service

Refer to *2-6-1 CPU Unit Status* for information on the CPU Unit status.

This section describes the operation of the processes.

Initialization

Initialization is performed only when the power supply is turned ON.

The following processing is performed for initialization.

Processing	Description
Self diagnosis at startup	Operation is monitored for the following errors: Power Supply Error, CPU Unit Reset, CPU Unit Watchdog Timer Error, and Incorrect Power Supply Unit Connected.*1
Data check	The *_RetainFail* (Retention Failure Flag) system-defined variable changes to TRUE at the following time: when the values of variables for which the Retain attribute was set to retain the values and the values in DM, EM, and HR Areas in the memory used for CJ-series Units*2 were not retained after a power interruption.
Detecting CJ-series Units*3	The CJ-series Units mounted in the Controller are detected.
Recording Power Turned ON and Power Interrupted events	The Power Turned ON and Power Interrupted events are recorded.

*1 Refer to *12-1-2 Fatal Errors in the CPU Unit* for information on the following errors: Power Supply Error, CPU Unit Reset, CPU Unit Watchdog Timer Error, and Incorrect Power Supply Unit Connected.

*2 You can use memory for CJ-series Units only with NJ-series CPU Units.

*3 You can use CJ-series Units only with NJ-series CPU Units.

Processing Executed with Tasks

Types of Processing That are Executed with Tasks

The following processing is performed with tasks.

Processing	Description
I/O refreshing	Data I/O for EtherCAT slaves, CJ-series*1 Basic I/O Units, and CJ-series Special Units is performed.
User program execution	The user programming for sequence control is executed. It also sends commands to the motion control process.
Motion control	Motion control is executed based on commands from the user program.
System common processing	System common processing, such as data trace processing and tag data link processing, is performed.

*1 You can use CJ-series Units only with NJ-series CPU Units.

Refer to *5-3-3 Basic Operation of Tasks for NX-series Controllers* and *5-4-3 Basic Operation of Tasks for NJ-series Controllers* for details on the processing that is executed with tasks.

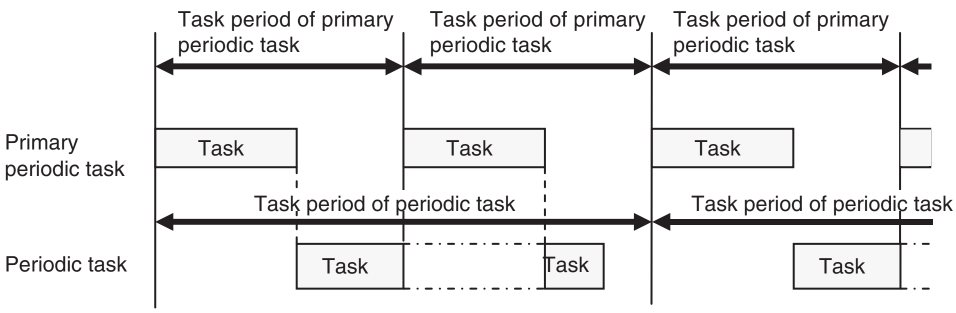
Task Operation

Processing is assigned to tasks. There are three kinds of tasks, as shown in the following table. They are defined by their execution priorities and execution conditions.

Type of periodic task	Execution priority (smaller values indicate higher priority)	Execution condition	Main processing content
Primary periodic task	4 This task has the highest execution priority.	The primary periodic task is periodically executed.	I/O refreshing, user program execution, and motion control
Periodic tasks	5*1 This task has the next highest execution priority after the primary periodic task.	The primary periodic task is periodically executed. The task period is an integer multiple of the task period of the primary periodic task.	I/O refreshing, user program execution, and motion control
	16, 17, or 18	The primary periodic task is periodically executed. The task period is an integer multiple of the task period of the primary periodic task.	The processing that can be performed depends on the task execution priority. Execution priority 16: I/O refreshing and user program execution Execution priority 17 or 18: User program execution
Event tasks	8 or 48	An event task is executed only once when the specified condition is met.	User program execution

*1 You can use the priority-5 periodic task only with NX-series CPU Units.

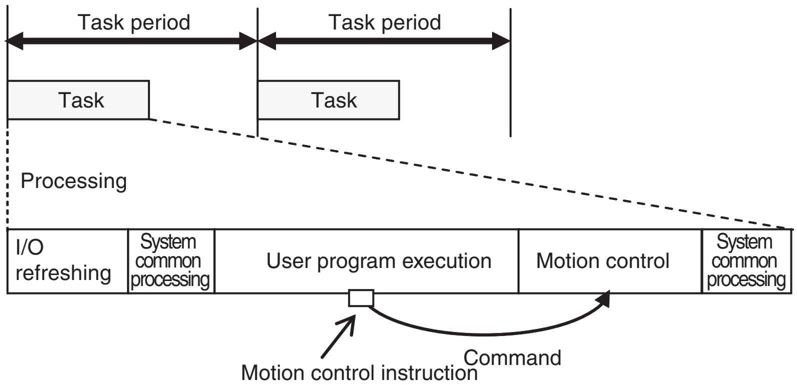
The CPU Unit executes the task with the highest execution priority first. The following operation example is for the primary periodic task and a periodic task. If the primary periodic task is ready for execution while a periodic task is in execution, execution of the primary periodic task is prioritized.



The operation of tasks with lower priority when a task with higher priority is in execution differs between NX-series and NJ-series CPU Units. Refer to *5-2 Overview of Tasks* for details on task operation.

● Operation of Processing with Tasks

Processing that is assigned to a task is executed within the task in the order shown in the following diagram. If the program contains a motion control instruction, the execution process for the program in the task will send a command to the motion control process. The motion control process is executed based on commands.



Note The CPU Unit executes motion control in the primary periodic task and in the priority-5 periodic task. Refer to *5-10-3 System Input and Output Response Times* for details.

Additional Information

With an NX-series CPU Unit, you can execute motion control in the primary periodic task and in the priority-5 periodic task. If these two motion controls need to be identified, the motion control in the primary periodic task is called motion control 1, while the motion control in the priority-5 periodic task is called motion control 2.

Tag Data Link Service

● Processing Performed by the Tag Data Link Service

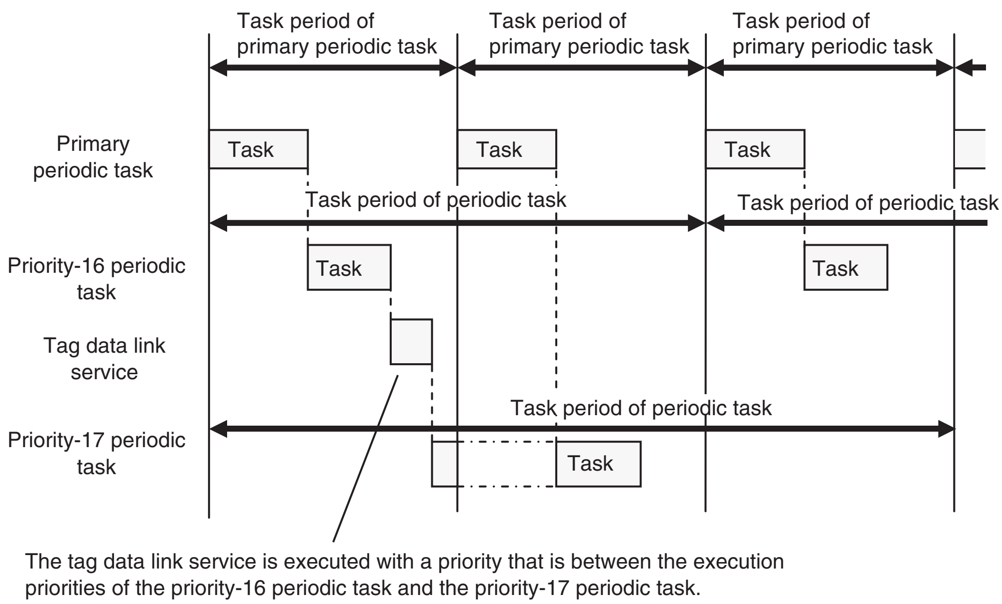
The tag data link service processes communications that use tags with other controllers or devices on an EtherNet/IP network. You can use the built-in EtherNet/IP port in the CPU Unit or a CJ-series CJ1W-EIP21 EtherNet/IP Unit to connect to an HMI.

Note You can use the CJ1W-EIP21 EtherNet/IP Unit only with NJ-series CPU Units.

● Operation of the Tag Data Link Service

The tag data link service is executed periodically. The period and the time that is required for each execution depend on the model of the CPU Unit and on the tag data link settings.

The execution priority of the tag data link service is between the execution priorities of the priority-16 periodic task and the priority-17 periodic task. Therefore, the tag data link service will be given priority over execution of the priority-17 periodic task, but if the primary periodic task or the priority-16 periodic task is executed, the execution of the primary periodic task or the priority-16 periodic task is prioritized.



For details on the tag data link service, refer to *5-5 Tag Data Link Service and System Services*.

System Services

- System Services

System services include the following processing.

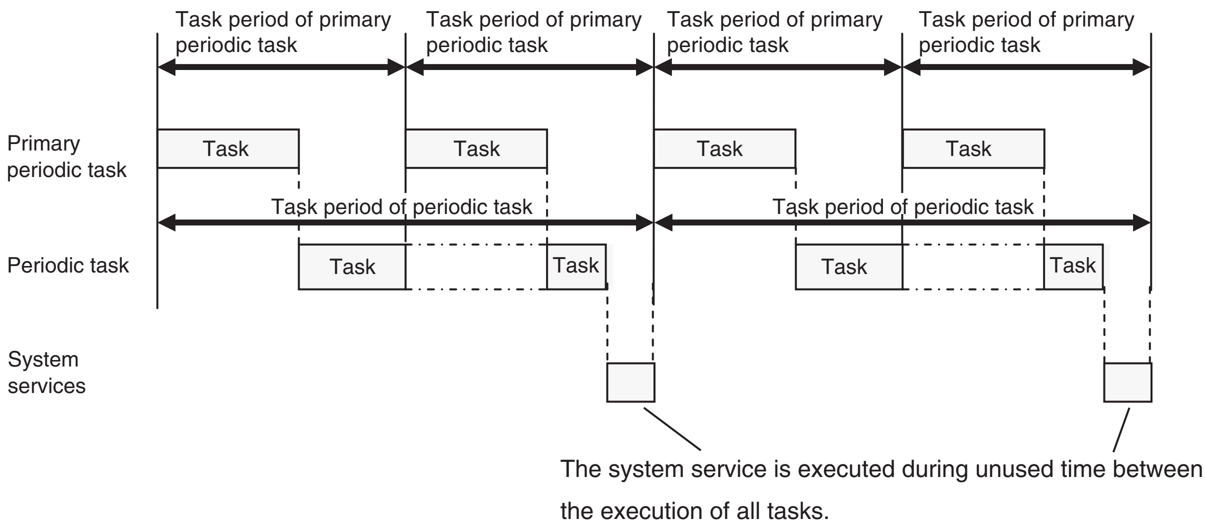
Processing	Contents
USB port service	• Processing of service requests from the Sysmac Studio or host computers
Built-in EtherNet/IP port service	• Processing of message service requests, such as CIP commands, from the Sysmac Studio, an HMI, host computers, or other Controllers • Execution of communications instructions for CIP and socket communications
Built-in EtherCAT port service	• Execution of EtherCAT message communications
CJ-series Special Unit service *	• Event servicing for CJ-series Special Units • Execution of communications instructions (CIP)
SD Memory Card service	• Access from FTP client • SD Memory Card operations from the Sysmac Studio • Execution of SD Memory Card instructions
Self-diagnosis	• Hardware error detection

* The CPU Unit exchanges data between CJ-series Special Units and the memory words that are allocated to them during I/O refreshing.
You can use CJ-series Special Units only with NJ-series CPU Units.

- **System Service Operations**

For NX-series CPU Units, if a request comes from the hardware or from outside of the CPU Unit, system services are executed in parallel with other processes. System services are executed at the required time without being affected by the execution of tasks.

For NJ-series CPU Units, if a request comes from the hardware or from outside of the CPU Unit, system services are executed during the unused time between the execution of all tasks.



Refer to *5-5 Tag Data Link Service and System Services* for details on the system services.

2-3 Accessing I/O

The CPU Unit uses variables to access I/O. This section describes how variables are used to access I/O.

In this manual, I/O on EtherCAT slaves and CJ-series Units are treated as I/O. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506) for details on how to access data in other Controllers with tag data links.

2-3-1 Types of Variables

In an NJ/NX-series CPU Unit, you use variables in the user program to access I/O and memory in the CPU Unit.

The type of a variable depends on whether it has attributes that are set by the user, and what it can access.

<table>
<tr><th colspan="3">Variables</th><th>Attribute settings</th><th>Accessed data</th></tr>
<tr><td colspan="3">User-defined variables</td><td>All attributes can be set.</td><td>CPU Unit</td></tr>
<tr><td rowspan="3">Semi-user-defined variables</td><td rowspan="2">Device variables</td><td>Device variables for EtherCAT slaves</td><td rowspan="3">Some attributes can be set.*1</td><td>EtherCAT slaves to which axes are not assigned*2*3</td></tr>
<tr><td>Device variables for CJ-series Units*4</td><td>CJ-series Basic I/O Units and CJ-series Special Units</td></tr>
<tr><td colspan="2">Cam data variables</td><td>Servo Drives, encoder input slaves, and CPU Unit</td></tr>
<tr><td rowspan="6">System-defined variables</td><td colspan="2">System-defined variables for PLC Function Module</td><td rowspan="6">No attributes can be set.</td><td>CPU Unit</td></tr>
<tr><td rowspan="3">System-defined variables for motion control</td><td>MC Common Variable</td><td rowspan="3">Servo Drives, encoder input slaves, and CPU Unit</td></tr>
<tr><td>Axis Variables</td></tr>
<tr><td>Axes Group Variables</td></tr>
<tr><td colspan="2">System-defined variables for EtherNet/IP</td><td>Built-in EtherNet/IP port</td></tr>
<tr><td colspan="2">System-defined variables for EtherCAT master</td><td>Built-in EtherCAT master port</td></tr>
</table>

*1 Refer to *Device Variable Attributes* on page 3-11 for the attributes that can be set.

*2 "EtherCAT slaves" includes any NX Units on EtherCAT Slave Terminals.

*3 With the Sysmac Studio version 1.09 or higher, the EtherCAT slaves to which axes are assigned can also be accessed via EtherCAT slave device variables.

*4 You can use CJ-series Units only with NJ-series CPU Units.

User-defined Variables

The user defines all of the attributes of a user-defined variable. Refer to *6-3 Variables* for details on user-defined variables.

Semi-user-defined Variables

Semi-user-defined variables have some attributes that you can set. These variables are used to access specific data. A semi-user-defined variable can either be a device variable or a cam data variable, depending on what it can access.

● Device Variables

Device variables are used to access data in devices. A device is a general term for any Unit or slave that is refreshed by the I/O refreshing that is performed by the CPU Unit. Specifically, it refers to EtherCAT slaves and CJ-series Units.

The device and the data to access in that device determine the type of device variable, as shown below.

Type of device variable	Device	Accessed data
Device variables for EtherCAT slaves	EtherCAT slaves to which axes are not assigned*1	Process data for EtherCAT slaves*2
Device variables for CJ-series Units*3	CJ-series Basic I/O Units	Real I/O data in Basic I/O Units
	CJ-series Special Units	Operating data*4 and setup data for Special Units*5

*1 With the Sysmac Studio version 1.09 or higher, the EtherCAT slaves to which axes are assigned can also be accessed via EtherCAT slave device variables.

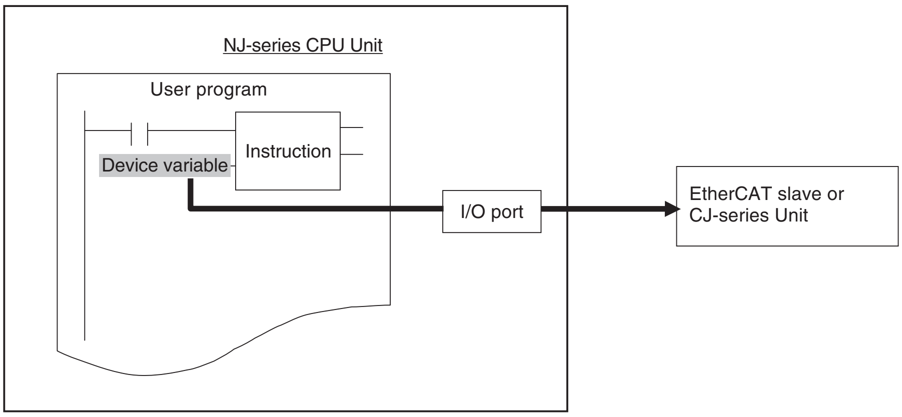
*2 This refers to I/O data that is exchanged during the process data communications cycle between the master and slaves.

*3 You can use CJ-series Units only with NJ-series CPU Units.

*4 This data is used in the operation of CJ-series Units. The CIO Area portion of the memory used for CJ-series Units is used.

*5 This data is used to set up the CJ-series Units. The DM Area portion of the memory used for CJ-series Units is used.

Device variables are used to access data for EtherCAT slaves and CJ-series Units through the I/O ports. The I/O ports are logical ports that are used to access devices.



Refer to *3-3-1 I/O Ports* for details on I/O ports and device variables.

● Cam Data Variables

Cam data variables are used to access data in cam tables, which are used for motion control. For details, refer to the *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

System-defined Variables

System-defined variables are defined in advance in an NJ-series Controller. The names and all attributes are defined by the system. They have specific functions. You cannot change the variable names or any other attributes.

The system-defined variables are specific to a function module. There are system-defined variables for each function module. The types of system-defined variables are listed in the following table.

Function module	Type of system-defined variable
PLC Function Module	System-defined variables for PLC Function Module
Motion Control Function Module	System-defined variables for motion control
EtherNet/IP Function Module	System-defined variables for EtherNet/IP
EtherCAT Master Function Module	System-defined variables for EtherCAT master

The system-defined variables for motion control are classified according to what the Motion Control Function Module does, as listed in the following table.

System-defined variables for motion control	Description
MC Common Variable	Common processing for the entire Motion Control Function Module
Axis variables	Control of individual axes
Axes Group variables	Control of axes groups*1

*1 An axes group consists of multiple axes. An axes group is used for interpolation.

Refer to *A-4 System-defined Variables* for details on system-defined variables.

2-3-2 Accessing I/O with Variables

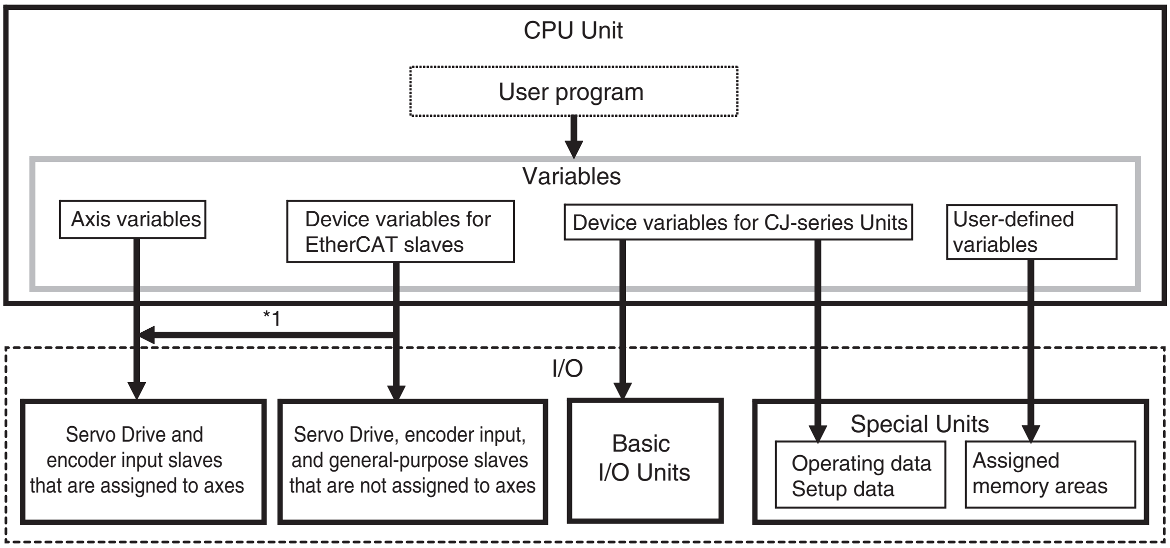
In the CPU Unit, variables are used in the user program. Variables access the data of the assigned I/O. The following table shows how I/O and variables are assigned in the CPU Unit. The type of variable that is used by a CJ-series Special Unit depends on the data to access.

I/O		Data	Variables
EtherCAT slaves	EtherCAT slaves to which axes are not assigned*1	---	Device variables for EtherCAT slaves
	EtherCAT slaves to which axes are assigned	---	Axis variables
CJ-series Units*2	Basic I/O Units	---	Device variables for CJ-series Units
	Special Units	• Operating data • Setup data	Device variables for CJ-series Units
		Assigned memory area data*3	User-defined variables

*1 With the Sysmac Studio version 1.09 or higher, the EtherCAT slaves to which axes are assigned can also be accessed via EtherCAT slave device variables.

*2 You can use CJ-series Units only with NJ-series CPU Units.

*3 This data is for extended functions and slave I/O that you assign by specifying addresses in memory. You cannot access assigned memory area data with device variables.



Accessing EtherCAT Slaves

The method that is used to access an EtherCAT slave depends on the type of EtherCAT slave.

Type of EtherCAT slave	Access method
• Servo Drive and encoder input slaves that are not assigned to axes*1 • General-purpose slaves	These slaves are accessed through I/O ports by using device variables for EtherCAT slaves.
Servo Drive and encoder input slaves that are assigned to axes	These slaves are accessed directly with Axis variables.*2

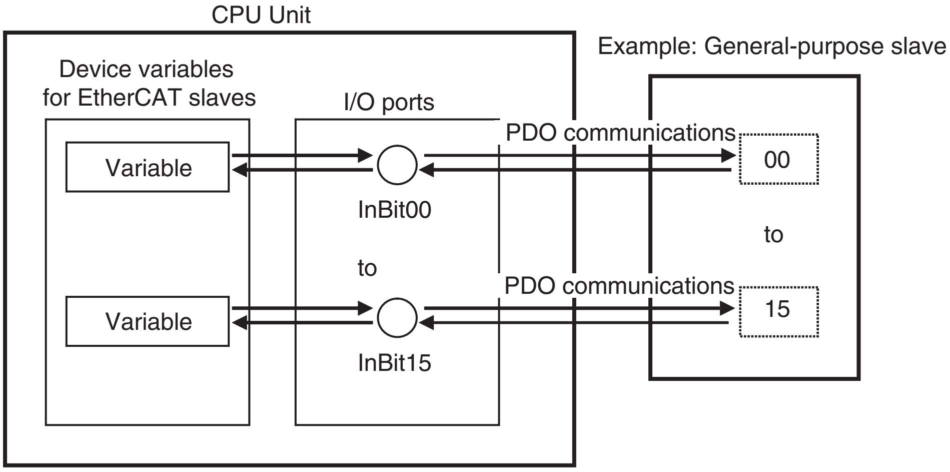
*1 With the Sysmac Studio version 1.09 or higher, the EtherCAT slaves to which axes are assigned can also be accessed via EtherCAT slave device variables.

*2 For a Servo Drive, one Servomotor is assigned as one axis to one Axis variable. For an encoder input slave, one counter is assigned as one axis to one Axis variable.

Note EtherCAT slaves that cannot be assigned to axes are called general-purpose slaves. EtherCAT slaves that can be assigned to axes are called Servo Drive and encoder input slaves. Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for details on Servo Drive and encoder input slaves.

● Accessing Servo Drive, Encoder Input, and General-purpose Slaves That Are Not Assigned to Axes

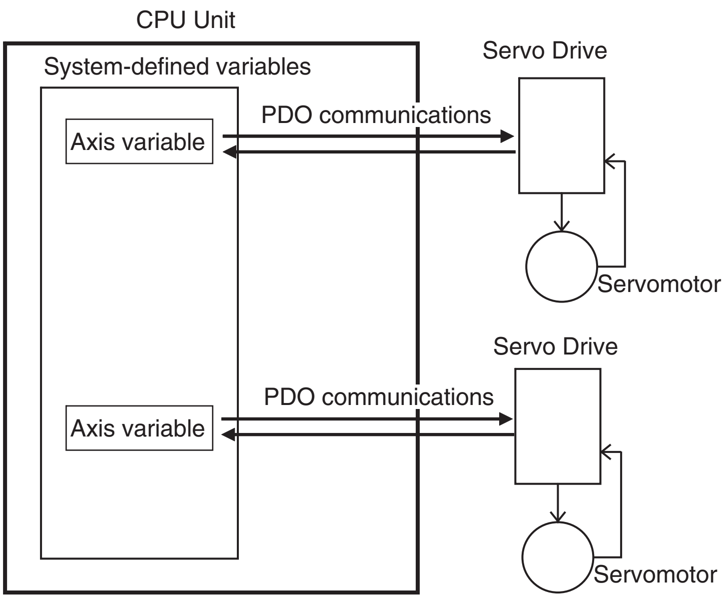
These slaves are accessed through I/O ports for device variables for EtherCAT slaves. PDO communications are used to access data from I/O ports.



● Accessing Servo Drive and Encoder Input Slaves That Are Assigned to Axes

Servo Drive and encode input slaves that are assigned to axes are accessed directly through the Axis variable. PDO communications are used to access data from Axis variables.

For example, if a Servomotor is controlled with a Servo Drive, the control commands for the Servomotor that is assigned to an Axis variable are sent to the Servo Drive. The feedback from the Servomotor is sent from the Servo Drive to the CPU Unit by using the Axis variable.



Refer to *3-5-2 Axis Variables and Axes Group Variables* for details on Axis variables.

Version Information

With the Sysmac Studio version 1.09 or higher, device variables can be assigned to the I/O ports of Servo Drive and encoder input slaves to which axes are assigned.
The I/O port to which a device variable can be assigned must meet either of the following conditions.

- The value of the R/W attribute is R (Read only).
- The value of the R/W attribute is W (Write only), and <Not assigned> is set for the process data field under **Detailed Settings** on the Axis Basic Settings Display in the Sysmac Studio.

Precautions for Correct Use

If you perform the following steps, the system will clear the assignment of the device variable to the I/O port of a Servo Drive and encoder input slave to which an axis is assigned.

(1) With the Sysmac Studio version 1.09 or higher, assign device variables to the I/O ports of Servo Drive and encoder input slaves to which axes are assigned.

(2) Save the project data.

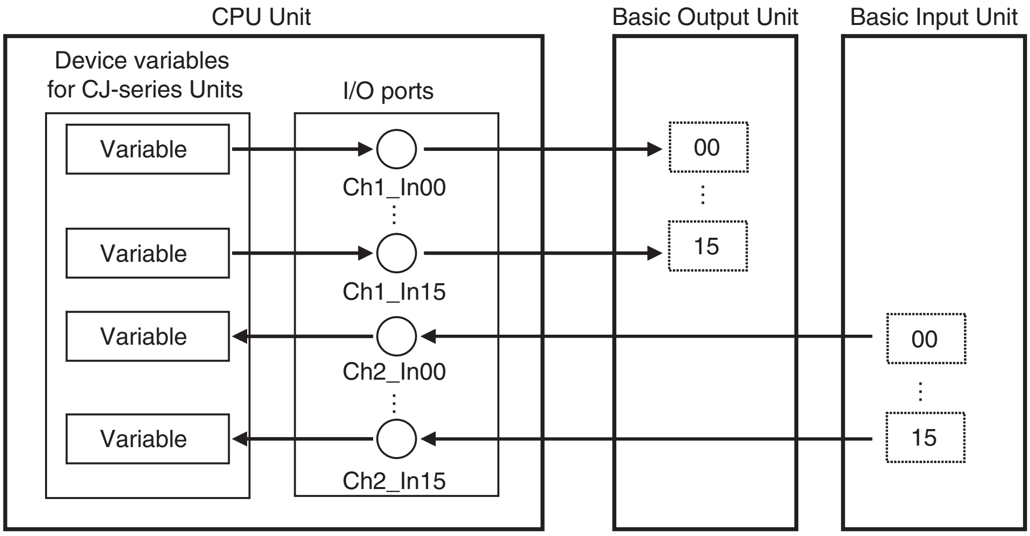
(3) Open the saved project data with the Sysmac Studio version 1.08 or lower.

Additional Information

There are two types of EtherCAT communications, PDO communications and SDO communications. PDO communications are used for commands to refresh I/O data, such as data for Servomotor position control, on a fixed control period. SDO communications are used for commands to read and write data at specified times, such as for parameter transfers.
Refer to the *NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual* (Cat. No. W505) for details.

Accessing Basic I/O Units

With an NJ-series CPU Unit, you access Basic I/O Units through the I/O ports for device variables for the CJ-series Unit.



Refer to *3-3-1 I/O Ports* for details.

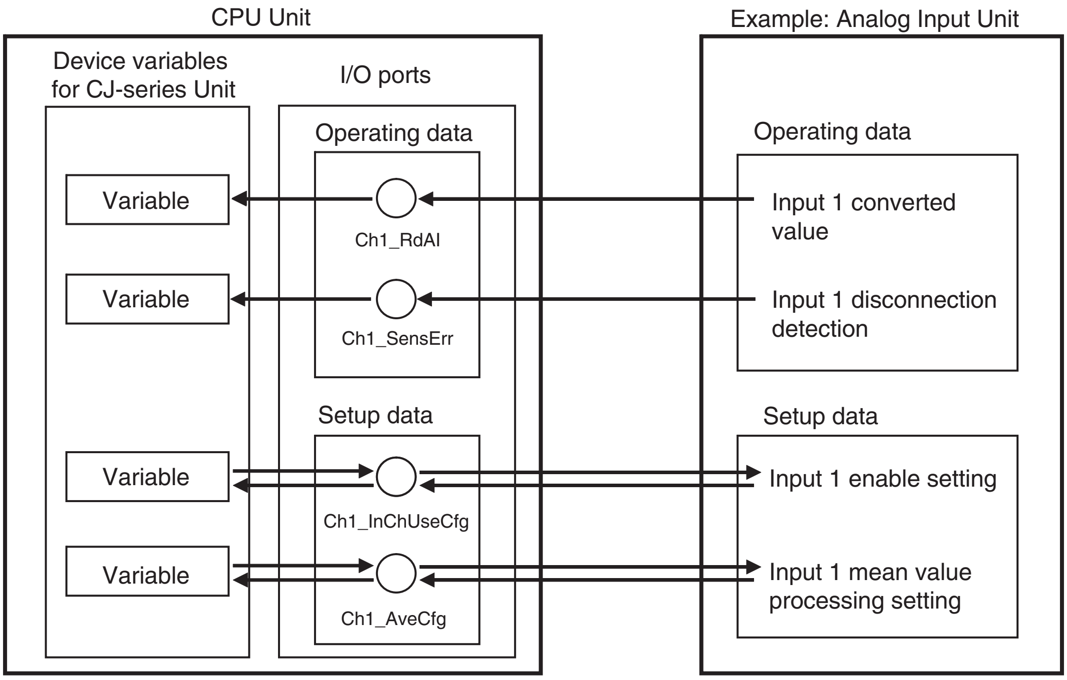
Accessing Special Units

With an NJ-series CPU Unit, there are two methods that you can use to access Special I/O Units. Which method is used depends on the data to access.

Access method	Data
Accessing Special Units through I/O ports by using device variables for CJ-series Units	• Operating data • Setup data
Accessing Special Units by using user-defined variables with AT specifications	Assigned memory area data

● Accessing Special Units through I/O ports by Using Device Variables for CJ-series Units

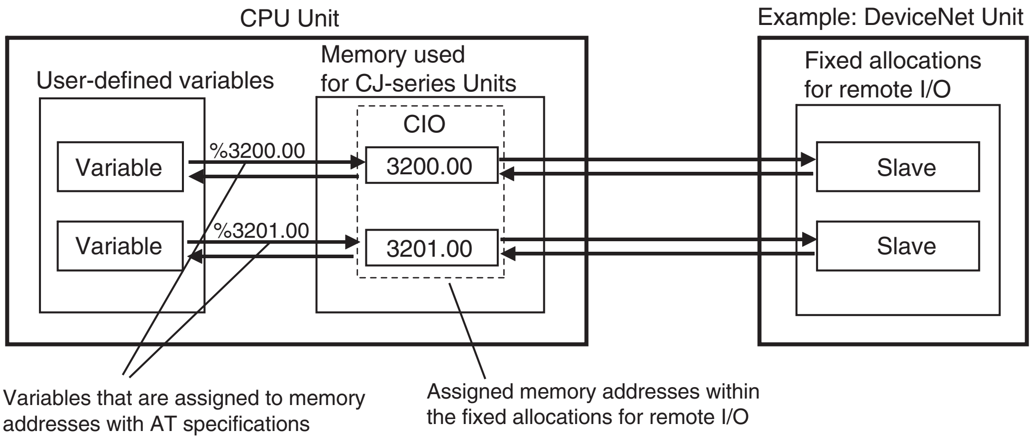
The operating data and setup data are accessed through the I/O ports for the device variables for the CJ-series Unit. The data is exchanged during I/O refreshing.



Refer to *3-3-1 I/O Ports* for details.

Accessing Special Units by Using User-defined Variables with AT Specifications

The assigned memory area addresses are accessed by using AT specifications for user-defined variables to the memory addresses that are used for the CJ-series Units. The data in the memory used for CJ-series Units is exchanged with the data in the CJ-series Units during I/O refreshing.



The assigned memory addresses including the following addresses.

- Addresses in fixed allocations for DeviceNet Units
- Addresses in user-specified allocations for DeviceNet Units or CompoNet Master Units from the CX-Integrator (A CompoNet Master Unit must be set to communications mode 8 to use the CX-Integrator.)
- Addresses in expansion memory for High-speed Counter Units
- Addresses in expansion memory for Process I/O Units

Refer to *A-7 Contents of Memory Used for CJ-series Units* for information on the memory used for CJ-series Units. Refer to *6-3-8 Variable Attributes* for information on AT specifications.

2-4 Sequence Control and Motion Control

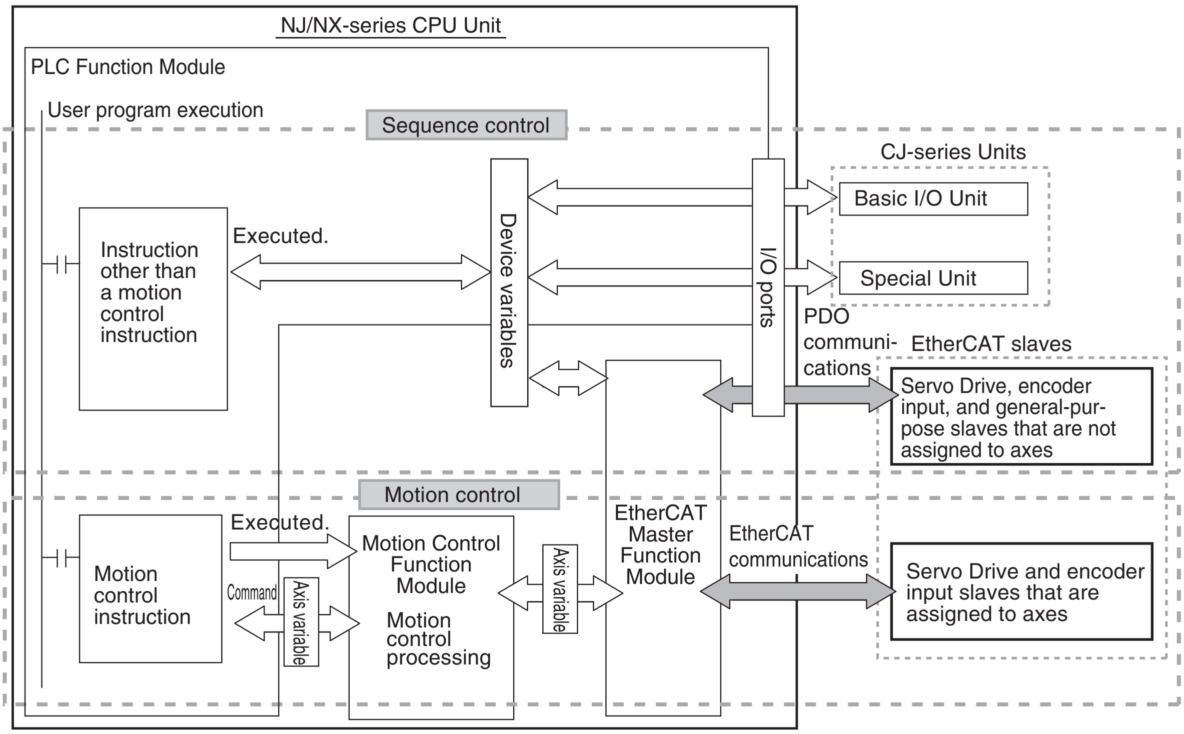
This section describes the sequence control and motion control systems that are used by the CPU Unit.

2-4-1 Overview of Control

The NJ/NX-series CPU Unit can perform both sequence control and motion control.

You execute sequence control with instructions other than motion control instructions in the user program. Sequence control is for EtherCAT slaves and CJ-series Units that are not assigned to axes. Control is performed by the PLC Function Module and the EtherCAT Master Function Module.

You perform motion control with motion control instructions in the user program for EtherCAT Servo Drive and encoder input slaves that are assigned to axes. Control is performed by the PLC Function Module, Motion Control Function Module, and the EtherCAT Master Function Module.



Note You can use CJ-series Units only with NJ-series CPU Units.

Additional Information

Instruction Types in Terms of Control Systems

In terms of the controls, the instructions can be broadly separated into the following two types of instructions.

Type of instruction	Definition
All instructions other than motion control instructions (sequence control)	These instructions are executed in the user program in the PLC Function Module and processing for them is completed there.
Motion control instructions	These instructions are executed in the user program in the PLC Function Module to send commands to the Motion Control Function Module. MC_Home (Homing), MC_Move (Positioning), MC_CamIn (Start Cam Operation), and other instructions for motion control operations

For details on motion control instructions, refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508). For details on other instructions, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

Version Information

With the Sysmac Studio version 1.09 or higher, device variables can be assigned to the I/O ports of Servo Drive and encoder input slaves to which axes are assigned.
The I/O port to which a device variable can be assigned must meet either of the following conditions.

- The value of the R/W attribute is R (Read only).
- The value of the R/W attribute is W (Write only), and <Not assigned> is set for the process data field under **Detailed Settings** on the Axis Basic Settings Display in the Sysmac Studio.

Precautions for Correct Use

If you perform the following steps, the system will clear the assignment of the device variable to the I/O port of a Servo Drive and encoder input slave to which an axis is assigned.

(1) With the Sysmac Studio version 1.09 or higher, assign device variables to the I/O ports of Servo Drive and encoder input slaves to which axes are assigned.

(2) Save the project data.

(3) Open the saved project data with the Sysmac Studio version 1.08 or lower.

2-4-2 Sequence Control System

The way that the sequence control works depends on the device to control. This section describes the operation of the function modules and the control period as part of the sequence control system.

Device	Sequence Control System	
	Operation of the function module	Control period
Servo Drive, encoder input*1, and general-purpose slaves	• The PLC Function Module executes the user program and refreshes the device variables. • The EtherCAT Master Function Module exchanges data with the slaves through the I/O ports for device variables.	The task period of the task to which the program is assigned (i.e., the task period of the primary periodic task or a periodic task)*2
CJ-series Units*3	The PLC Function Module executes the user program, refreshes the device variables, and exchanges data with the CJ-series Units.	The task period of the task to which the program is assigned (i.e., the task period of the primary periodic task or a periodic task)*4

*1 With the Sysmac Studio version 1.09 or higher, a Servo Drive and encoder input slave to which an axis is assigned can also be a part of sequence controls if you assign the device variable to the I/O port of the slave.

*2 The data refresh period in the slave depends on settings in the slave.

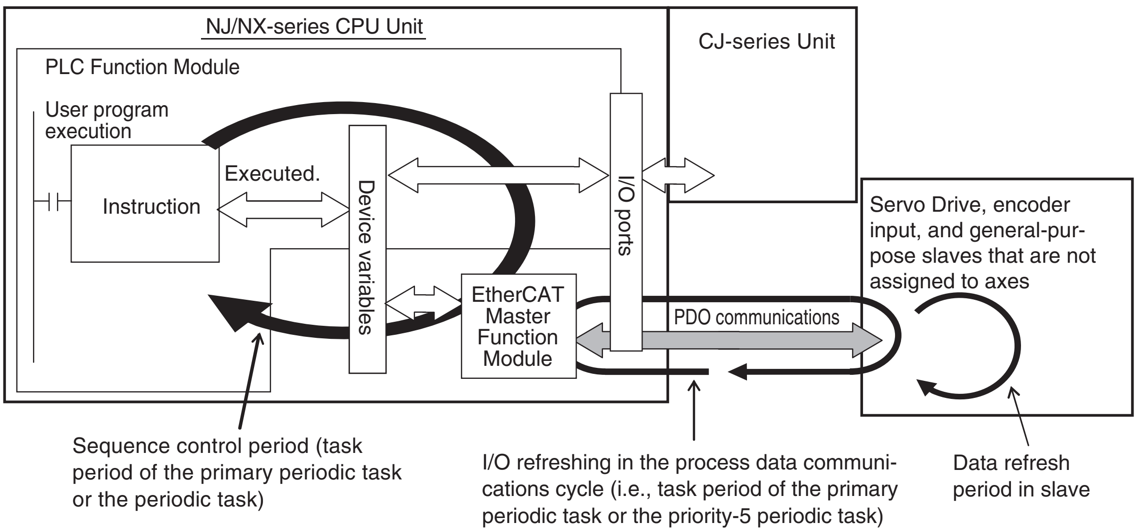
*3 You can use CJ-series Units only with NJ-series CPU Units.

*4 The data exchange period with a CJ-series Unit is the task period to which I/O refreshing for the CJ-series Unit is assigned.

Servo Drive, encoder input, and general-purpose slaves that are not assigned to axes are refreshed in the process data communications cycle. This means that I/O refreshing takes place in the task period of the primary periodic task or the priority-5 periodic task. However, execution of the programs and refreshing of the device variables take place in the task period of the task to which the programs are assigned. Therefore, the slave values are not reflected and not controlled by the device variables until the task period of the task to which the programs are assigned.

If it is necessary to control a slave in the process data communications cycle, assign the program that controls the slave to the primary periodic task or the priority-5 periodic task.

For details, refer to *5-10-3 System Input and Output Response Times.*



Additional Information

- You can use the priority-5 periodic task only with NX-series CPU Units.
- With an NX-series CPU Unit, you can perform process data communications in the primary periodic task and the priority-5 periodic task. If these two process data communications cycles need to be identified, the communications cycle for the primary periodic task is called process data communications cycle 1, while the communications cycle for the priority-5 periodic task is called process data communications cycle 2.
- NJ-series CPU Units perform process data communications only in the primary periodic task.

2-4-3 Motion Control System

This section describes the operation of the function modules and the control period as part of the motion control system.

● Operation of Function Modules

- The PLC Function Module executes motion control instructions in the user program and sends commands for motion control to the Motion Control Function Module. Axis variables are used for these commands.
- The Motion Control Function Module performs motion control processing based on commands from the PLC Function Module. It then reflects the results of this processing in the Axis variables.
- The EtherCAT Master Function Module sends the command values of the Axis variable to the Servo Drive or other slave by using EtherCAT communications.

● Control Period

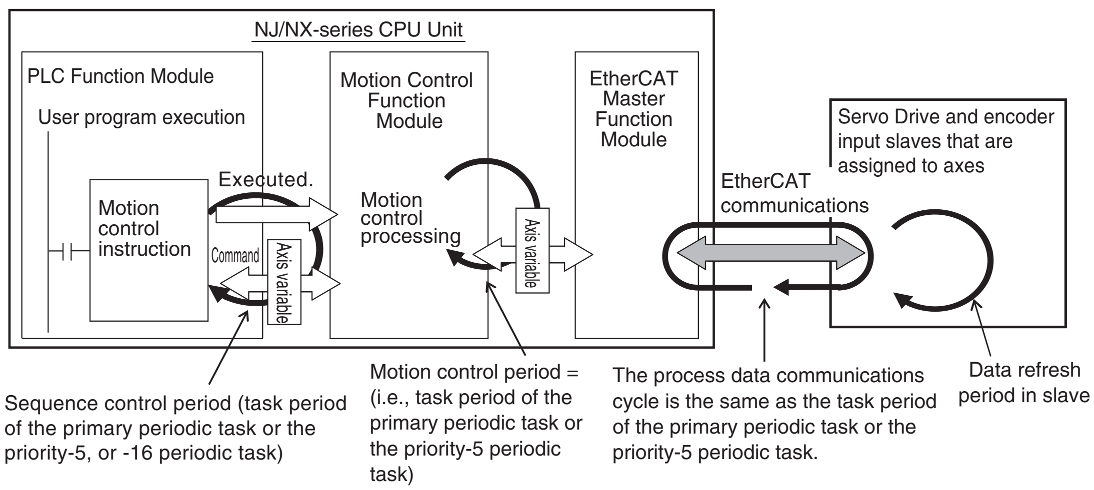
The motion control period is the task period of the primary periodic task or the priority-5 periodic task.

Motion control processing in the Motion Control Function Module is executed in the task period of the primary periodic task or the priority-5 periodic task. The Motion Control Function Module also exchanges data with Servo Drive and encoder input slaves that are assigned to the axes to control in the process data communications cycle of the primary periodic task or the priority-5 periodic task. The process data communications cycle is synchronized with the primary periodic task or the priority-5 periodic task.

This makes the motion control period the same as the task period of the primary periodic task or the priority-5 periodic task, which allows complete synchronization of multiple axes.

However, the following restrictions apply:

- The motion control instruction is executed and the command for motion control is sent in the sequence control period.
- The data refresh period in the EtherCAT slave depends on settings in the slave.



Precautions for Correct Use

- You can use the priority-5 periodic task only with NX-series CPU Units.
- With an NX-series CPU Unit, you can execute motion control in the primary periodic task and in the priority-5 periodic task. If these two motion controls need to be identified, the motion control in the primary periodic task is called motion control 1, while the motion control in the priority-5 periodic task is called motion control 2.
- NJ-series CPU Units perform motion control only in the primary periodic task.

Additional Information

- You must use the Sysmac Studio to assign an axis to an EtherCAT slave to control it from the Motion Control Function Module. This allows the PLC Function Module to send commands to the Motion Control Function Module for motion control instructions that are executed in the user program. It also allows the PLC Function Module to obtain information from the Motion Control Function Module through the Axis variables.
- The task to which the program that contains the motion control instructions is assigned determines the I/O response time of the motion control system. For details, refer to *5-10-3 System Input and Output Response Times.*

2-4-4 Synchronizing Sequence Control and Motion Control

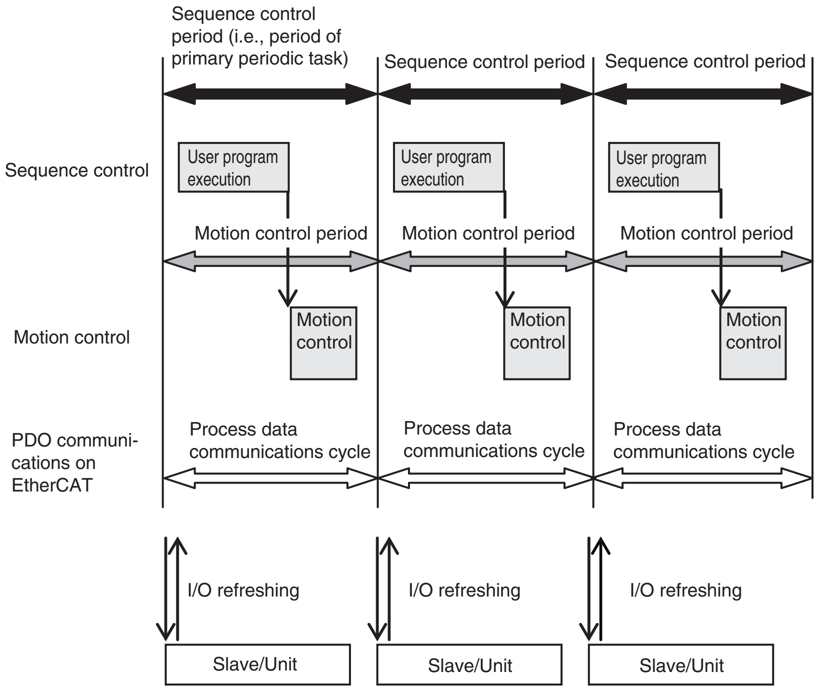
The sequence control period is the task period of the task to which the program and I/O refreshing are assigned. However, motion control is always executed in the task period of the primary periodic task or the priority-5 periodic task. The process data communications cycle for the EtherCAT slave to use for motion control is synchronized with the primary periodic task or the priority-5 periodic task.

With an NX-series CPU Unit, you can set motion control to execute motion processing for each axis and each axes group. For axes or axes groups for which you set motion control 1, motion control and process data communications are executed in the task period of the primary periodic task. For axes or axes groups for which you set motion control 2, motion control and process data communications are executed in the task period of the priority-5 periodic task.

NJ-series CPU Units perform motion control and process data communications for axes and axes groups in the task period of the primary periodic task.

If you assign the sequence control program to the task that motion control is executed, you can synchronize the sequence and motion control periods with the process data communications cycle for EtherCAT.

The following diagram shows a program assigned to the primary periodic task. In the following diagram, the period of sequence control, motion control, and process data communications on EtherCAT are all synchronized.



Additional Information

- Relationship among motion controls, tasks, and process data communications cycles for NX-series CPU Units

Motion control	Task to execute	Process data communications cycle
Motion control 1	Primary periodic task	Process data communications cycle 1
Motion control 2	Priority-5 periodic task	Process data communications cycle 2

- Relationship among motion controls, tasks, and process data communications cycles for NJ-series CPU Units

Motion control	Task to execute	Process data communications cycle
Motion control	Primary periodic task	Process data communications cycle

2-5 Overview of CPU Unit Data

The CPU Unit also contains settings, the user program, present values, and other data. The main data is described in the following table.

Refer to *A-6 Attributes of CPU Unit Data* for information on all of the data in the CPU Unit.

Type of data			Description
Settings	**EtherCAT Configuration**	**EtherCAT Slave Configuration**	This is information on the EtherCAT slave configuration.
		EtherCAT Master Settings	The EtherCAT Master Settings contain parameter settings for the EtherCAT Master Function Module, such as the communications cycle.
	Unit Configuration and Unit Setup		The Unit Configuration and Unit Setup contain information on the Unit configuration that enables the CPU Unit to recognize the Units, and the initial settings of the Special Units.
	I/O Map		The I/O Map contains assignment information between the variables and the I/O ports that are automatically created based on the Unit Configuration.
	Controller Setup	**Operation Settings**	The Operation Settings include the Startup Mode setting, Security Settings, and System Service Monitoring Settings.
		Built-in EtherNet/IP Port Settings	The Built-in EtherNet/IP Port Settings contain the following settings: TCP/IP settings, Ethernet settings, DHCP settings, DNS settings, FTP settings, NTP settings, and SNMP settings
	Motion Control Setup		The Motion Control Setup consists of settings for Axis variables and Axes Group variables for axis and axes groups, and motion control parameter settings.
	Cam Data Settings		The cam data includes cam tables that consist of phase/displacement data for use in cam operation for motion control instructions.
	Event Setup		These settings are for user-defined errors and user-defined information.
	Task Setup		The Task Setup contain settings for the task types, number of tasks, task execution conditions, task names, programs executed in the task, and other task settings.
	Data Trace Settings		The Data Trace Settings include settings for trigger conditions.
	Tag Data Link Tables		The Tag Data Link Tables contain the tag data link settings for EtherNet/IP.
	Controller Name		The Controller name is the name of the CPU Unit.
	Operation Authority Verification		This data contains the operation authority passwords to perform Sysmac Studio operations for the CPU Unit.
	Built-in Clock	**Set Time**	This is the time information that is used inside the CPU Unit.
		Time Zone Setting	This is the time zone that is set for the clock in the CPU Unit.
User Program	**POUs (program organization units)**		These are the definitions of the programs, functions, and function blocks. The local variable tables and the initial values of the variables are also included.
	Data	**Data Types**	This data contains the definitions of the data types.
		Global Variables	This data gives the attribute information of the global variables. It includes the Initial Value and Retain attributes.
Present Values	**Values of Variables**		This data contains the values of the variables.
	Contents of Memory Used for CJ-series Units*1		These are the values of the CIO, Working, Holding, DM, and EM Areas in the memory for CJ-series Units.

Type of data		Description
Other Data	**Event Logs**	The event logs include the error log for the Controller, and logs of events other than errors, such as when the power supply was turned ON and OFF and when operation started.
	Absolute Encoder Home Offsets	This data is used to restore the actual position of a Servo Drive with an absolute encoder in motion control. The offset is the difference between the command position after homing and the absolute data that is read from the absolute encoder.

*1. You can use memory for CJ-series Units only with NJ-series CPU Units.

2-6 Operation for CPU Unit Status

This section describes the processing that is performed for user program execution, I/O refreshing, and external communications according to the status of the CPU Unit. It also describes the operating modes that change the execution status of the user program when the CPU Unit is in the normal operation state.

2-6-1 CPU Unit Status

The CPU Unit can be in any of three states: startup state, normal operation state, or error state. These states are defined as follows:

State	Definition
Startup state	The software is initializing the system.
Normal operation	The software is executing processing for instructions that are executed in a task or it is executing a system service. A Controller error has not occurred.
Error state	A Controller error occurred when the software was executing processing for instructions that are executed in a task or it was executing a system service.

The normal operation state has these three states for operation: PROGRAM mode, RUN mode, and downloading. A CPU Unit in the normal operation state changes to the other states due to user interaction. This status is defined as follows:

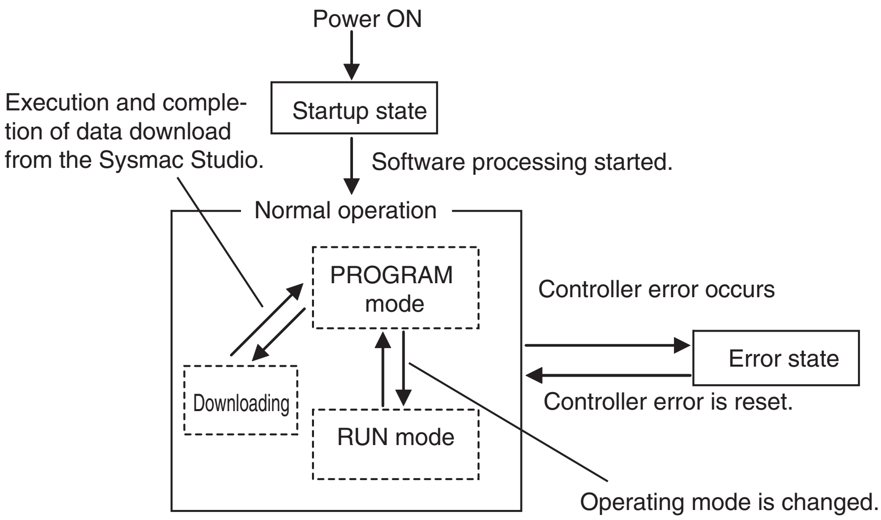
State	Definition
PROGRAM mode	The operating mode is PROGRAM mode.
RUN mode	The operating mode is RUN mode.
Downloading	Data is being downloaded from the Sysmac Studio.

Note Refer to *2-6-3 Operating Modes* for details on PROGRAM mode and RUN mode.

CPU Unit status

The CPU Unit enters the startup state after the power supply is turned ON. About 10 to 20 seconds after the CPU Unit enters the startup state, software processing begins and the CPU Unit changes to normal operation. If a Controller error occurs during normal operation, the CPU Unit changes to the error state. When you reset the Controller error, the CPU Unit returns to normal operation.

When the CPU Unit changes from startup state to normal operation, it will change to the operating mode that you specify in the Controller Setup. You can set the operating mode at startup to PROGRAM mode or RUN mode. Thereafter, changing the operating mode causes the CPU Unit to change between PROGRAM mode and RUN mode. If you download data from the Sysmac Studio during PROGRAM mode, the CPU Unit will change to the downloading state. The CPU Unit will return to PROGRAM mode when the download is completed.



Additional Information

- You can check the operating status of the CPU Unit with the status indicators on the front panel of the CPU Unit. Refer to *12-1 Operation after an Error* for troubleshooting procedures using the status indicators.
- Refer to *A-6 Attributes of CPU Unit Data* for information on data operations when the CPU Unit status changes.
- Refer to *6-3-9 Changes to Variables for Status Changes* for the values that variables take when the status of the CPU Unit changes.

2-6-2 Operation for CPU Unit Status

Changes in the status of the CPU Unit affect user program execution, I/O refreshing, and the operation of external communications. The following table shows how each process operates in startup state and during normal operation.

Refer to *12-1-3 Non-fatal Errors in the CPU Unit* for information on the error state.

CPU Unit processing	Operation during execution	Operation during normal operation		
		PROGRAM mode	RUN mode	Downloading
User program	Stopped.	Stopped.	Executed.	Stopped.
I/O Refreshing for EtherCAT slaves	Stopped.	Executed.		EtherCAT communications changes to safe-operational state. *1
I/O Refreshing for CJ-series Units*2	Stopped.	Executed.		
External communications	Stopped.	Executed.		Executed. *3

*1 Only the input values are refreshed.

*2 You can use CJ-series Units only with NJ-series CPU Units.

*3 The tag data links remain in effect, but the values of those links are not refreshed. The output tags retain the values from before the download was started. The values in the input tags are not reflected in the variables.

● Values of Outputs in I/O Refreshing

The following table shows the values of the outputs in each state after I/O refresh processing.

Outputs	Operation during startup	Operation during normal operation		
		PROGRAM mode	RUN mode	Downloading
Outputs from EtherCAT slaves	Controlled by the slave settings. *1	The outputs have the values of the device variables for EtherCAT slaves. *2		Controlled by the slave settings.
Outputs from CJ-series Basic Output Units*3	Turned OFF.	The output have the values of the device variables for CJ-series Units. *4		

*1 Refer to the manual for each slave for information on the slave settings that apply until EtherCAT communications starts after the power supply is turned ON.

*2 When the download is completed, initialization of the EtherCAT slaves starts. When initialization is in progress, the outputs reflect the settings for the slave.

*3 You can use CJ-series Units only with NJ-series CPU Units.

*4 When the download is completed and when the operating mode is changed: the values in the device variables for CJ-series Units are initialized to the values of the Initial Value attributes.

Refer to *6-3-8 Variable Attributes* for information on the Initial Value attribute for variables.

Additional Information

Servo Drive Response to Changes in Operating Mode

If the operating mode changes from RUN to PROGRAM mode during a motion control operation, the axes will decelerate to a stop at the maximum deceleration rate.

Changing the Operating Mode during Initialization of EtherCAT Slaves

You can change the operating mode of the CPU Unit to RUN mode while EtherCAT slaves initialization is in progress. If you do, provide programming to confirm that communications are established before you attempt to use slave data in control operations. Your program can use the *_EC_PDSlavTbl* (Process Data Communicating Slave Table) system-defined variable to see if the process data inputs and outputs are valid for all of the slaves.

2-6-3 Operating Modes

You can change the operating mode according to the purpose of operation, such as functional testing or actual operation. You can set the operating mode to RUN mode or PROGRAM mode, depending on the purpose. The execution status of the user program is different in each operating mode. The following table gives the purpose for each operating mode and the execution status of the user program.

Operating mode	Application	User program execution status
RUN mode[*1]	RUN mode is for trial operation or actual operation.	Executed.
PROGRAM mode	PROGRAM mode is for checking I/O wiring and other functional testing without executing the user program.	Not executed.

*1 For the default setting, the CPU Unit will enter RUN mode when the CPU Unit changes from startup state to normal operation.

Additional Information

The CPU Unit performs various operations when the operating mode is changed, i.e., the axes are stopped, and motion control instructions are aborted. For details on how the Motion Control Function Module operates when the operating mode is changed, refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508).

Operations Allowed from the Sysmac Studio or an HMI in Each Operating Mode

The major operations that you can perform from the Sysmac Studio or an HMI in each operating mode are listed in the following table.

<table>
<tr><th colspan="2">Operation</th><th>RUN mode</th><th>PROGRAM mode</th></tr>
<tr><td rowspan="4">Sysmac Studio</td><td>Synchronization</td><td>Not possible.</td><td>Possible.</td></tr>
<tr><td>Online editing</td><td colspan="2">Possible.</td></tr>
<tr><td>Forced refreshing</td><td colspan="2">Possible.</td></tr>
<tr><td>Changing the values of variables or memory used for CJ-series Units[*1]</td><td colspan="2">Possible.</td></tr>
<tr><td>HMI</td><td>Changing the values of variables or memory used for CJ-series Units[*1]</td><td colspan="2">Possible.</td></tr>
</table>

*1. You can use memory for CJ-series Units only with NJ-series CPU Units.

Retention of Variable Values during Changes in Operating Mode

The following table shows how the Retain attribute affects the variable values when the operating mode is changed between RUN mode and PROGRAM mode.

Retain attribute of variable	Values of variables
Non-retain	If initial values are set, the variables change to the initial values. If no initial values are set, the variables change to the system-defined initial values.*1
Retain	The values before the operating mode changed are retained.

*1 The system-defined initial values of variables depend on the data types of the variables. Refer to *When the Initial Value Specification Is Left Blank* on page 6-60.

Refer to *6-3-9 Changes to Variables for Status Changes* for the values that variables take when the status of the CPU Unit changes.

Setting and Changing the Operating Mode

When operation starts after the power supply is turned ON, the CPU Unit operates in the operating mode that you specify in the Controller Setup. During normal operation, you change the operating mode for different purposes. You use the Sysmac Studio to set and change the operating mode.

● Operating Mode Setting after the Power Supply Is Turned ON

When the CPU Unit starts operating after the power supply is turned ON, the CPU Unit operates in the operating mode that you set as the Startup Mode. Specify RUN mode or PROGRAM mode in the *Startup Mode* setting in the **Operation Settings** in the Controller Setup. Refer to *4-2-2 Controller Setup* for details on the Startup Mode setting.

● Changing the Operating Mode during Operation

You can change the operating mode from the Sysmac Studio. Select the RUN mode or PROGRAM mode from ***Controller - Operating Mode*** on the menu bar.

Precautions for Safe Use

Always confirm the safety of the controlled system before you change the setting of the Startup Mode or the current operating mode.

Checking the Operating Mode

You can check the operating mode with the RUN indicator on the CPU Unit or the Sysmac Studio.

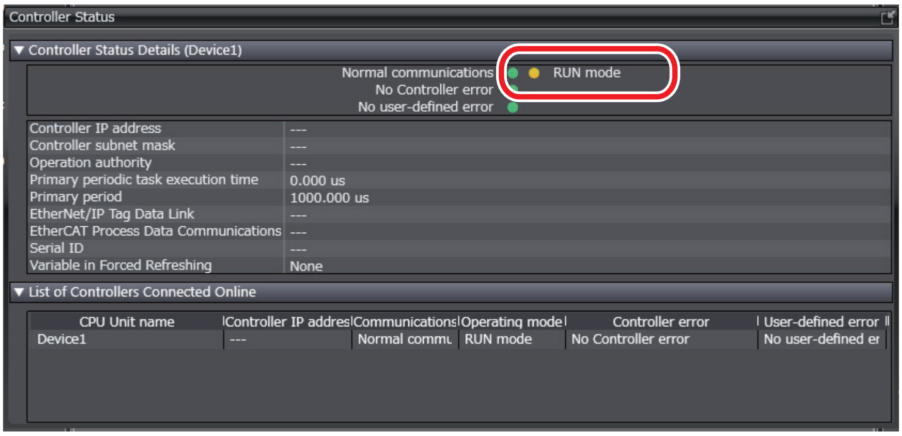
● Checking the RUN Indicator

The RUN indicator on the CPU Unit indicates the operating mode as given below.

RUN indicator status	Operating mode
Not lit	PROGRAM mode
Lit	RUN mode

● Checking the Operating Mode from the Sysmac Studio

You can check the operating mode from the Controller Status Pane of the Sysmac Studio. The following Controller Status Pane indicates that the CPU Unit is in RUN mode.



Additional Information

If you want to output a signal when the CPU Unit is in RUN mode, use the RUN output on the Power Supply Unit. Refer to the *NX-series CPU Unit Hardware User's Manual* (Cat. No. W535) and the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) for details on the RUN output on the Power Supply Unit.

3

I/O Ports, Slave Configuration, and Unit Configuration

This section describes how to use I/O ports, how to create the Slave and Unit Configurations, and how to assign functions.

3-1 Procedure to Create the Slave and Unit Configurations

This section provides the procedures for the Slave and Unit Configurations.

The shaded steps in the overall procedure that is given below are for the Slave and Unit Configurations.

Step 1. Software Design

- Step 1-1 Designing I/O and Processing
- Step 1-2 Designing Tasks
- Step 1-3 Designing Programs

Step 2. Software Setups and Programming

- Step 2-1 Slave and Unit Configurations
- Step 2-2 Controller Setup
- Step 2-3 Programming
- Step 2-4 Offline Debugging

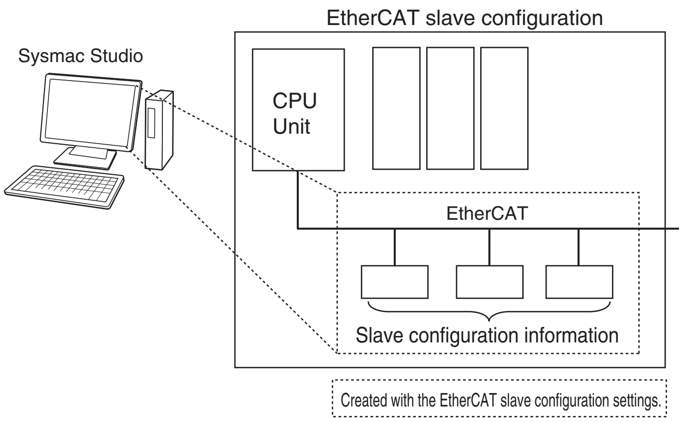
Step 3. Mounting and Setting Hardware

Step 4. Wiring

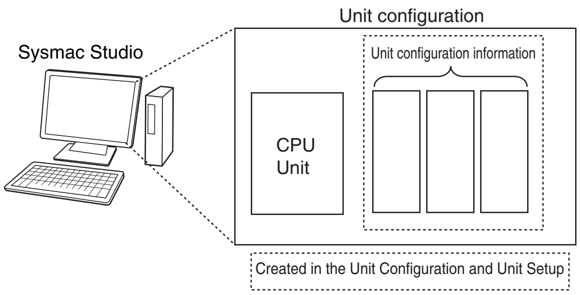
Step 5. Checking Operation and Starting Operation on the Actual System

For details, refer to *1-3 Overall Operating Procedure for the NJ/NX-series Controller*.

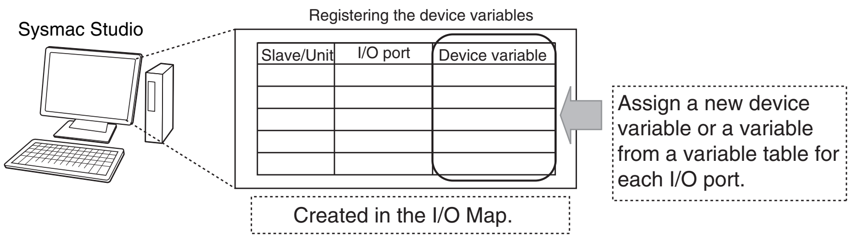
Step 1 Create the EtherCAT Slave Configuration (if EtherCAT is used) and the Unit Configuration (if CJ-series Units are used).	Reference
• Create the EtherCAT Slave Configuration.	*3-2-1 Creating the EtherCAT Slave Configuration* *3-2-2 Creating the Unit Configuration*



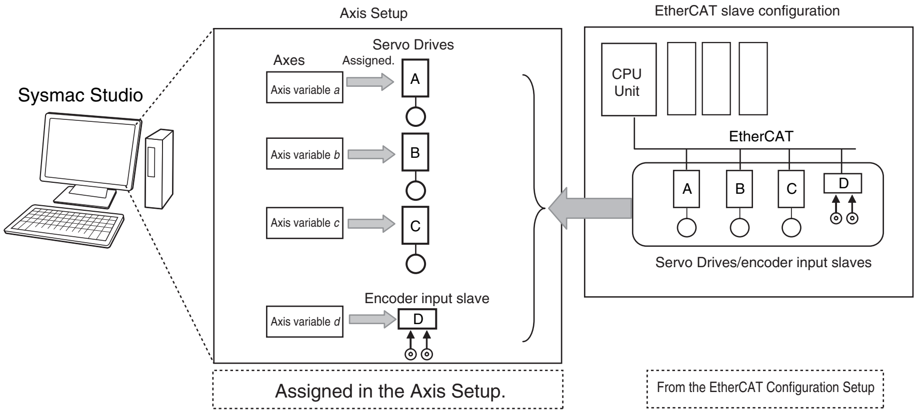
• Create the Unit Configuration.



Step 2 Assign device variables to I/O ports.	Reference
• Register the device variables.	*2-3-1 Types of Variables* *3-3 I/O Ports and Device Variables*



Step 3 Create the axes and assigning them to the Servo Drive and encoder input slaves (if motion control is used).	Reference
1. Create the axes. 2. Assign the axes to the Servo Drive and encoder input slaves in the EtherCAT configuration.	*3-5 Creating the Axes and Assigning Them to the Servo Drives/Encoder Input Slaves*

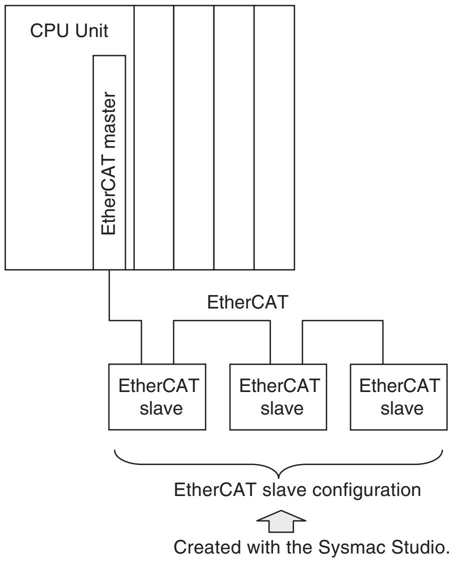


3-2 Creating and Comparing the Slave and Unit Configurations

To enable accessing the slaves and Units in the Controller, you create a Slave Configuration and a Unit Configuration on the Sysmac Studio. You can also compare the Unit Configuration that was created on the Sysmac Studio with the physical Unit configuration.

3-2-1 Creating the EtherCAT Slave Configuration

In the EtherCAT Tab Page of the Sysmac Studio, create the EtherCAT Slave Configuration that is detected as "correct" by the CPU Unit.



The I/O ports are automatically registered for the slaves in the configuration. Later, the user assigns device variables to the I/O ports. You can specify device variables in the user program to access the slaves.

Refer to *EtherCAT Configuration and Settings* in the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific procedures to create the EtherCAT slave configuration.

Additional Information

If you connect EtherCAT Slave Terminals, create the EtherCAT slave configuration, create the Slave Terminal configuration, and set the operation settings. Refer to the *NX-series EtherCAT Coupler Unit User's Manual* (Cat. No. W519) for information on the Slave Terminal configuration and operation settings.

Version Information

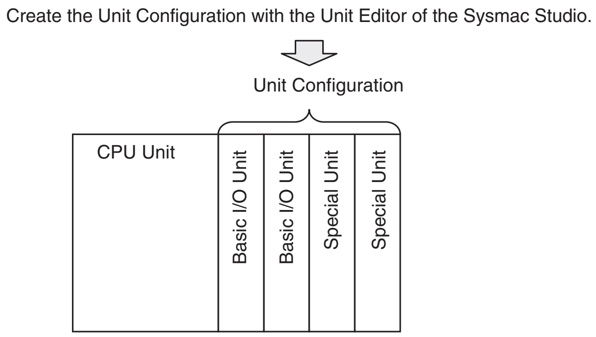
A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use EtherCAT Slave Terminals.

Additional Information

If the EtherCAT slaves are Servo Drives or encoder input slaves, after they are registered in the EtherCAT slave configuration, Axis Variables are registered automatically by creating the axes. Refer to *3-5 Creating the Axes and Assigning Them to the Servo Drives/Encoder Input Slaves* for details.

3-2-2 Creating the Unit Configuration

Use the Unit Editor in the Unit Configuration and Setup Tab Page of the Sysmac Studio to create the Unit Configuration that is recognized as correct by the CPU Unit.



When the power is turned ON, an automatic check is performed to determine whether the “correct” Unit Configuration matches the physical Unit configuration. The I/O ports are automatically registered for Units that are specified in the Unit Configuration. Later, the user assigns device variables to the I/O ports. The device variables are used in the user program to access the Units in the Unit Configuration.

Refer to *CPU/Expansion Rack Configuration and Setup* in the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific procedures to create the Unit Configuration.

Precautions for Correct Use

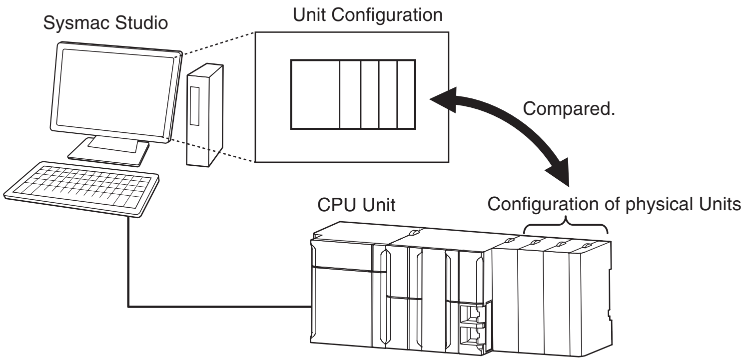
- You can start the Controller without creating the Unit Configuration. However, if you do so, the I/O ports and device variables will not be registered automatically, so you will not be able to access the Units from the user program.
- Create the Unit Configuration to use CJ-series Units with an NJ-series CPU Unit.
  NX-series CPU Units do not have the Unit configuration because Units are not used.

3-2-3 Verifying the Unit Configuration

You can use either of the following two methods to compare the Unit Configuration.

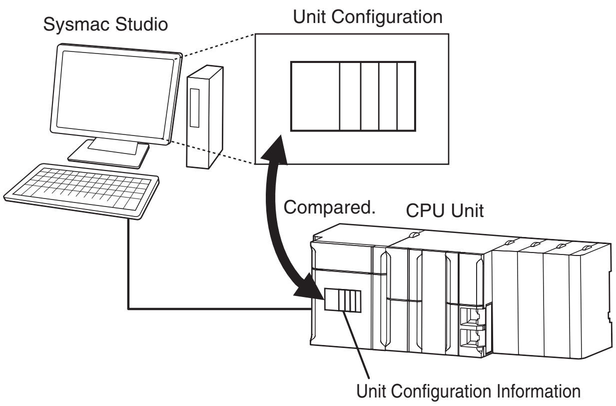
Comparison between the Unit Configuration on the Sysmac Studio and the Physical Unit Configuration

You can verify if the Unit Configuration on the Sysmac Studio and the physical Unit configuration are the same. You can compare the Unit Configuration on the Sysmac Studio with the physical Unit configuration to see if they are the same before the first time you download the Unit Configuration to the CPU Unit from the Sysmac Studio.



Comparison between the Unit Configuration on the Sysmac Studio and the Unit Configuration in the CPU Unit

You can verify if the Unit Configuration on the Sysmac Studio and the Unit Configuration that is saved in the CPU Unit are the same. You can compare the Unit Configuration on the Sysmac Studio with the Unit Configuration Information that is stored in the CPU Unit to see if they match before you download the Unit Configuration to the CPU Unit from the Sysmac Studio.



3-3 I/O Ports and Device Variables

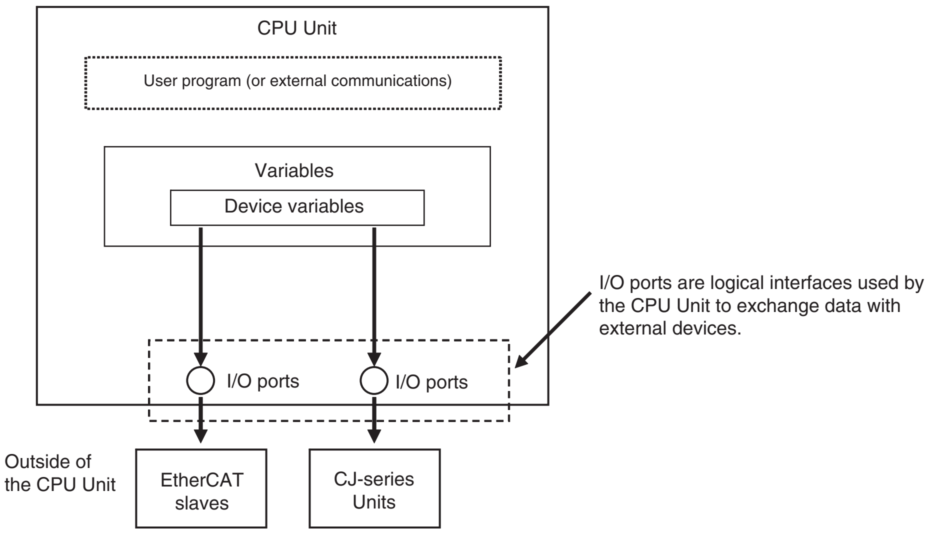
This section describes the I/O ports and device variables that you use to access the EtherCAT slaves and CJ-series Units of an NJ/NX-series Controller.

Precautions for Correct Use

You can use CJ-series Units only with NJ-series CPU Units.

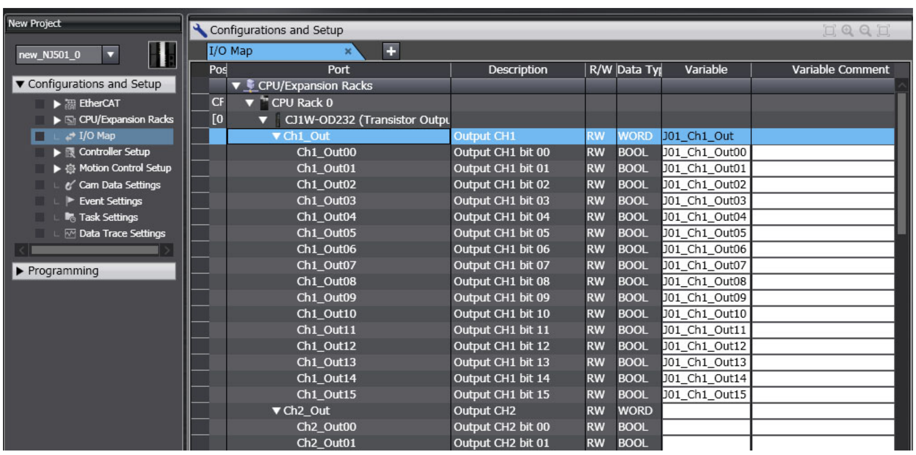
3-3-1 I/O Ports

An I/O port is a logical interface that is used by the CPU Unit to exchange data with external devices (slaves and Units). I/O ports are automatically created when you create the Slave and Unit Configurations on the Sysmac Studio. You assign device variables to I/O ports to enable accessing the slaves and Units from the user program.



I/O ports are automatically registered in the I/O Map when you create the EtherCAT Slave Configuration or Unit Configuration in the Sysmac Studio, or when you read either of these configurations from the physical Controller from the Sysmac Studio. You can check the I/O ports that were registered in the I/O Map of the Sysmac Studio.

I/O Map



3-3-2 I/O Port Names

The I/O port names are registered automatically. The I/O port names differ as given below depending on whether the device (i.e., Unit or slave) is an EtherCAT slave, CJ-series Basic I/O Unit, or CJ-series Special Unit.

EtherCAT Slave Devices

The following I/O port names are used for Remote I/O Terminals that are EtherCAT slave devices.

Example for a 16-point Remote I/O Terminal: Bit00 to Bit15

For other slaves, all or part of the object names that are defined in the EtherCAT object dictionary are used.

Example for Analog Input Unit: CH0_input16-bit

Examples for the R88D-KN50H-ECT: Position actual value and Digital inputs

CJ-series Basic I/O Unit Devices

If the device is a CJ-series Basic I/O Unit, I/O port names are created according to the following rules.

● Rules for I/O Port Names for Basic I/O Units

Inputs	Outputs
Ch□_In□□ Terminal number: 00 to 15 16-bit words: 1 to 4	Ch□_Out□□ Terminal number: 00 to 15 16-bit words: 1 to 4

● Example of I/O Port Names for Specific Numbers of I/O Points

Number of input points Number of output points	I/O port names			
	Inputs	Data type	Outputs	Data type
32 points	Ch1_In	WORD	Ch1_Out	WORD
	Ch1_In00 to Ch1_In15	BOOL	Ch1_Out00 to Ch1_Out15	BOOL
	Ch2_In	WORD	Ch2_Out	WORD
	Ch2_In00 to Ch2_In15	BOOL	Ch2_Out00 to Ch2_Out15	BOOL

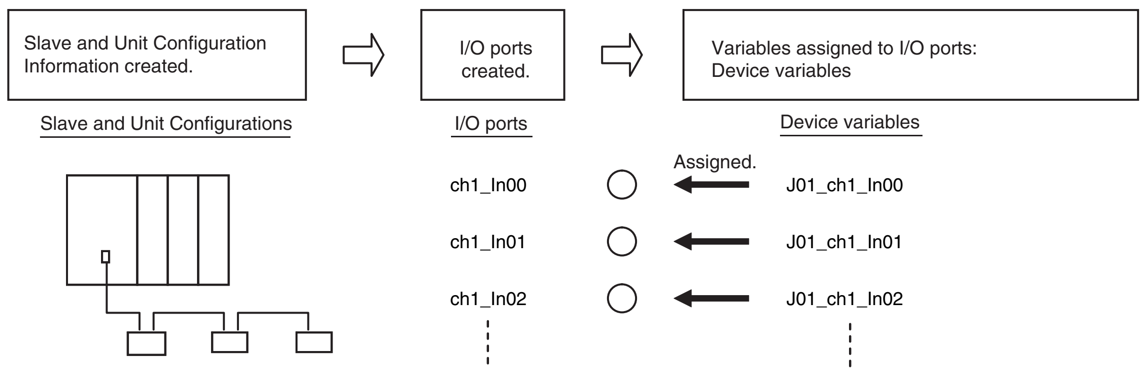
CJ-series Special Unit Devices

If the device is a CJ-series Special Unit, I/O port names are determined by the model number of the Unit and the functionality.

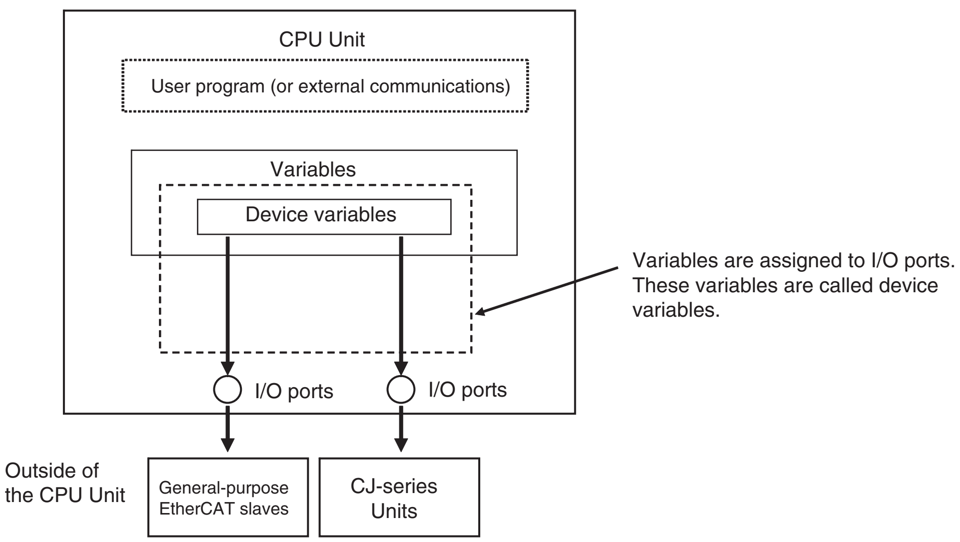
Examples for a CJ1W-AD041-V1 Analog Input Unit:
Ch1_PkHdCmd, *Ch1_AveCfg*, etc.

3-3-3 Device Variables

In an NJ/NX-series Controller, external devices (slaves and Units) are not assigned to specific memory addresses in the CPU Unit. Rather, variables are assigned to the I/O ports. These variables are called device variables.



You can specify device variables in the user program or in external communications to access the devices (slaves or Units).



Refer to *2-3-1 Types of Variables* for the relationship of device variables to other variables.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details on registering device variables with the Sysmac Studio.

Device Variable Attributes

The attributes of the device variables are described in the following table. You can change the settings of some of the attributes, but not all of them.

Attribute	Setting	Changes to settings
Variable Name	Automatically generated variables: [*device_name*] + [*I/O_port_name*] The default device names are as follows: • For EtherCAT slaves, an E followed by a sequential number starting from 001. • For CJ-series Units, the device names start with a J followed by a sequential number starting from 01. Refer to *3-3-2 I/O Port Names* for more information on I/O port names. If entered manually, the variable name is the string you enter.	Allowed.
Data Type	According to the data type of the I/O port.	Allowed.
AT Specification	• NX Units connected through an EtherCAT Coupler Unit: *ECAT://node#[EtherCAT_Coupler_Unit_node_address.NX_Unit_number]/[I/O_port_name]* • EtherCAT slaves: *ECAT://node#[node_address]/[I/O_port_name]* • CJ-series Units: IOBus://rack#[*rack_number*]/slot#[*slot_number*]/[*I/O_port_name*]	Not allowed.
Retain	• Device variables for EtherCAT slaves: Not retained. • Device variables for CJ-series Units assigned to the Operating Data (CIO Area): Not retained Device variables for CJ-series Units assigned to the Setup Data (DM Area): Retained	Not allowed.
Initial Value	None	Allowed.
Constant	None	Allowed.
Network Publish	Do not publish.	Allowed.
Edge	None	Not allowed.

Note You can use CJ-series Units only with NJ-series CPU Units.

Refer to *6-3-4 Attributes of Variables* for the meanings of the attributes.

Additional Information

- You can specify forced refreshing for I/O ports in the I/O Map. You can force real I/O to turn ON or OFF to check the wiring.
- You can choose the variable table (global variable table or local variable table for one POU) in which to register a device variable in the I/O Map.

3-4 Allocating Variables to Units

For some instructions, the Units on EtherCAT Slave Terminals are specified by using variables. Therefore, you must assign variables to the Units in advance. After you assign variables to the Units, the connection locations of the Units are automatically updated in the variables even if you change the locations. This means that you do not have to assign variables again every time you change the Unit connection locations.

Version Information

A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to assign variables to Units.

Additional Information

You can assign variables to EtherCAT slaves other than Slave Terminals. This applies to EtherCAT slaves from other manufacturers. The variables are assigned to the EtherCAT slaves in the same way as they are assigned to EtherCAT Coupler Units and NX Units.

3-4-1 Procedure to Assign Variables to Units

The variables assigned to the Units are not created automatically when you make configuration settings for an EtherCAT Slave Terminal on the Sysmac Studio. You must make the following settings to assign the variables to the Units.

1. On the Sysmac Studio, select **Configurations and Setup** – **EtherCAT** and make configuration settings for EtherCAT Slave Terminals.
2. Select **Configurations and Setup** – **I/O Map** to display the I/O Map.

   The I/O Map is displayed for the Units of the set EtherCAT Slave Terminals.
3. Right-click the model of Unit to which you want to assign variables and select ***Display Node Location Port*** from the menu.

   The ***Node location information*** port will be added on the I/O Map.
4. Right-click the ***Node location information*** and select ***Create Device Variable***.

   The variable name will be written automatically to the *Variable* Field of the ***Node location information*** port.

The data type of variables assigned to the Units is _sNXUNIT_ID structure. The details on the _sNXUNIT_ID structure data type are given in the following table.

Variable	Name	Meaning	Data type
User specified	Specified Unit	Specified Unit	_sNXUNIT_ID
NodeAdr	Node address	Node address of the EtherCAT Coupler Unit	UINT
IPAdr	IP address*1	IP address of the EtherCAT Coupler Unit	BYTE[5]
UnitNo	Unit number	Unit number of the specified Unit	UDINT
Path	Path*1	Path information to the specified Unit	BYTE[64]
PathLength	Valid path length	Valid path length	USINT

*1 This information is used only inside the Controller. You cannot access or change it.

Precautions for Correct Use

The values of variables assigned to the Units will be set automatically when you register the variables. Do not change the values of the variables. If you change the value, the Controller may not perform the intended operation.

Additional Information

The data type of variables assigned to EtherCAT slaves other than Slave Terminals is _sECAT_ID structure. The details are given in the following table.

Variable	Name	Meaning	Data type
User specified	Specified slave	Specified slave	_sECAT_ID
NodeAdr	Node address	Node address of the EtherCAT Coupler Unit	UINT

3-4-2 Using Variables Assigned to Units

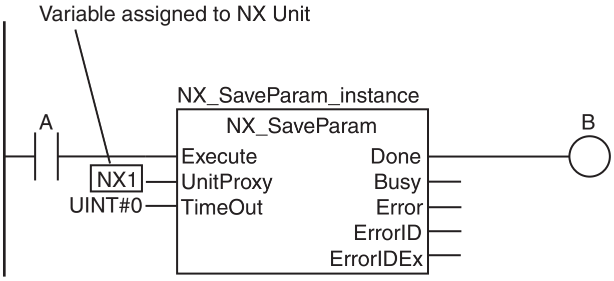
This section describes how to use the variables assigned to the Units in the user program. In any cases, the variable with the same name as the variable assigned to the Unit on the I/O Map must be registered in the variable table in advance. The data type of the variable is _sNXUNIT_ID structure.

Designating Units

The variables assigned to the Units are passed as parameters to the instructions for which specify the Units.

Example: Executing the NX_SaveParam Instruction

In the following example, the NX Unit to which the *NX1* variable is assigned is specified when the NX_SaveParam instruction is executed. *NX1* is passed to the *UnitProxy* variable.



Designating Unit Attributes

You can specify members of the variables that you assign to Units to specify some of the Unit attributes.

Example: Executing an Instruction with the Unit Number of an NX Unit

The following programming example reads a data object from an NX Unit if the NX Unit number of the NX Unit to which the *NX1* variable is assigned is 2. The *NX1.UnitNo* member gives the NX Unit number.

```
IF (NX1.UnitNo = UINT#2) THEN
  NX_ReadObj_instance(Execute:=TRUE, UnitProxy:=NX1, Obj:=S_Obj, ReadDat:=Rdat);
END_IF;
```

Designating More Than One Unit

To designate more than one Unit, you can specify the elements of an array of the variables that are assigned to the Units. This allows you to use loop processing to perform the same process for more than one Unit.

Example: The following programming example changes multiple NX Units to the mode that enables writing data.

NX0, NX1, and *NX2* are the variables that were assigned to the NX Units. The variables are assigned to the elements of the *NXTable[0..2]* and then the NX_ChangeWriteMode instruction is executed in order for each.

- Variable Table

Variable	Data type
NXTable	ARRAY[0..2] OF_sNXUNIT_ID

- ST Program

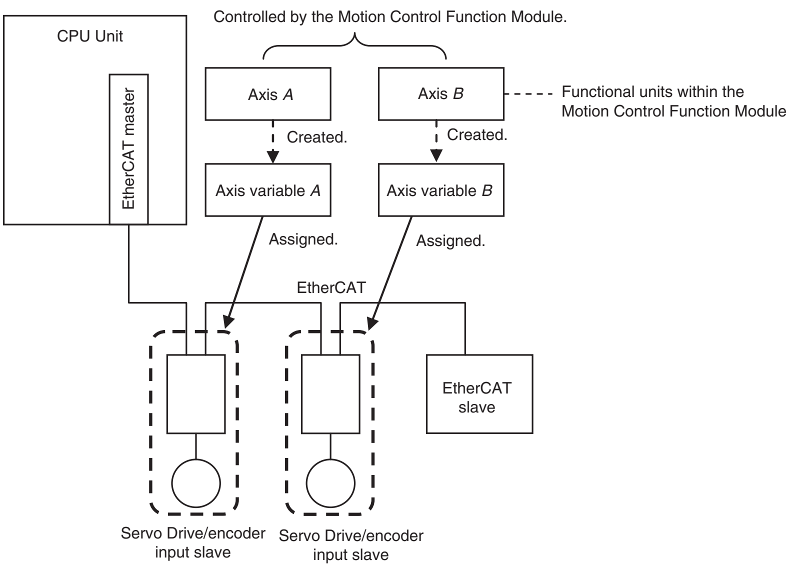
```
FOR i:= 0 TO 2 DO
  NX_ChangeWriteMode_instance[i](Execute:=FALSE);
END_FOR;

NXTable[0] := NX0;
NXTable[1] := NX1;
NXTable[2] := NX2;

FOR i:= 0 TO 2 DO
  NX_ChangeWriteMode_instance[i](Execute:=TRUE, UnitProxy:=NXTable[i]);
END_FOR;
```

3-5 Creating the Axes and Assigning Them to the Servo Drives/Encoder Input Slaves

This section describes how to create axes in the NJ/NX-series Controller and how to assign the axes to the Servo Drive and encoder input slaves.

3-5-1 Introduction

When you use the Motion Control Function Module for operation with EtherCAT Servo Drive or encoder input slaves, create axes in the Sysmac Studio and define them as EtherCAT servo axes or encoder axes. At a result, Axis Variables are automatically created as system-defined variables.



You can specify an Axis Variable in a motion control instruction in the user program to easily access and perform operations with Servo Drive and encoder input slaves.

3-5-2 Axis Variables and Axes Group Variables

The following table lists the types of Axis Variables and Axes Group Variables.

Type of variable		Application	Device to access	Creation method
Axis Variables	System-defined axis variables	An Axis Variable is used to control a single axis.	The EtherCAT slave (Servo Drive or encoder input slave) that is assigned to the axis	Provided by the system.
	Axis Variables automatically created when axes are created with the Sysmac Studio			You must create an axis with Sysmac Studio and assign the device to the axis.
Axes Group Variables	System-defined axes group variables	An Axes Group Variable is used for multi-axes coordinated control.	The EtherCAT slaves (Servo Drive or encoder input slaves) that are assigned to the axes group	Provided by the system.
	Axes Group Variables automatically created when axes groups are created with the Sysmac Studio			You must create an axes group with the Sysmac Studio.

Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on Axis Variables and Axes Group Variables.

Specifying Axis and Axes Group Variables

The variables can be specified with variable names that are created with the Sysmac Studio or with system-defined variable names.

Type	Names	
	Axis Variables	Axes Group Variables
Variable names created with the Sysmac Studio	MC_Axis*** ("***" is assigned in ascending order from 000 in the order the variables are created.) You can change the names as required.	MC_Group*** ("***" is assigned in ascending order from 000 in the order the variables are created.) You can change the names as required.
System-defined variable names	_MC_AX[***]*1 (The array element numbers are assigned in ascending order from 0 in the order the variables are created.)	_MC_GRP[***]*1 (The array element numbers are assigned in ascending order from 0 in the order the variables are created.)

*1 With the NX-series CPU Unit, you can also use *_MC1_AX[***]*, *_MC2_AX[***]*, *_MC1_GRP[***]*, and *_MC2_GRP[***]*. For details, refer to the *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

Application

There are two ways to use Axis Variables and Axes Group Variables.

1) Specifying Axes and Axes Groups in Motion Control Instructions:
If you specify an axis or axes group for an I/O variable for a motion control instruction, you can perform operations for the OMRON Servo Drive or encoder input slave.

2) Monitoring Axis Variable Members:
You can use instructions to monitor the actual position, error information, or other information on the Servo Drive and encoder input slaves.

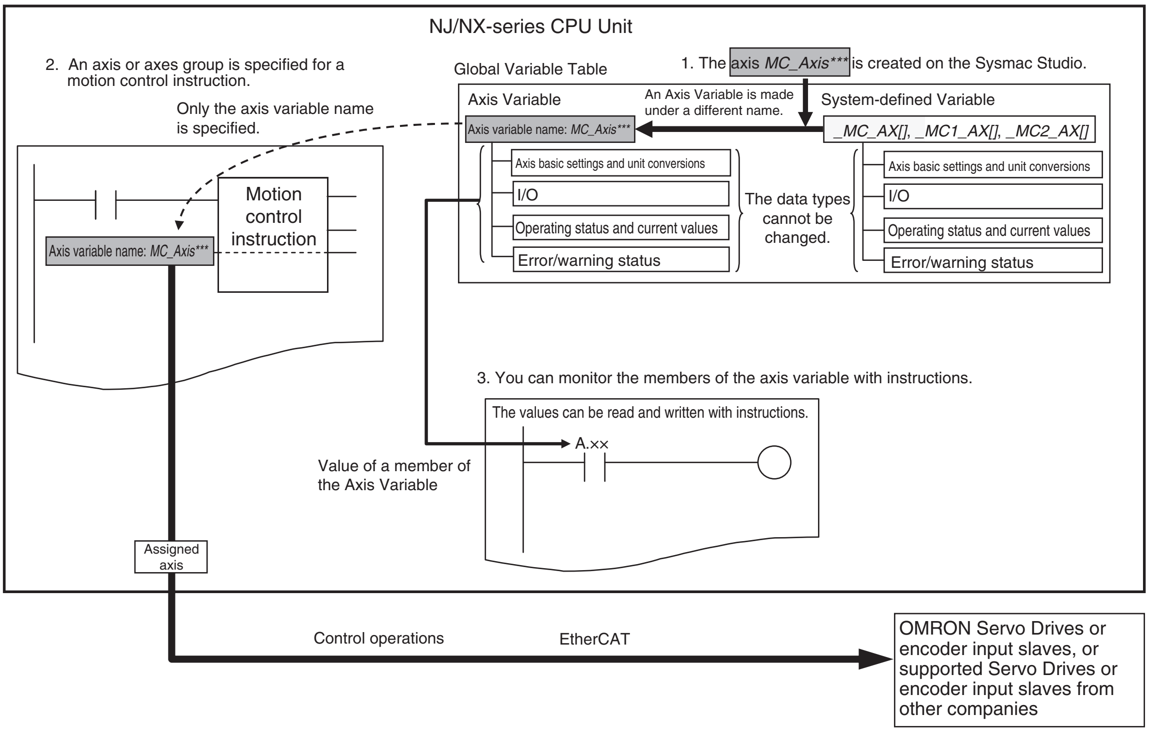
Additional Information

Details on Axis Variables

1. Assume that you create an axis with an axis name of A on the Sysmac Studio. An Axis Variable with a variable name of *A* is created automatically based on the system-defined axis variable. The Axis Variable consists of Axis Basic Settings, Unit Conversion Settings, I/O, operating status, current values, error status, and warning status.

2. You specify the axis variable name *A* for the in-out variable of a motion control instruction. With the axis variable name, you can access the OMRON Servo Drive or encoder input slave, or supported Servo Drive or encoder input slave from another company and perform operations for it.

3. You can specify the Axis Variable to use instructions as required to monitor the actual position, error information, or other information on the Servo Drive or encoder input slave.



3-5-3 Creating and Using Axes and Axis Variables

You can create and use axes and Axis Variables as described below.

1 Right-click **Axis Settings** under **Configurations and Setup** – **Motion Control Setup** in the Multiview Explorer and select ***Add*** – ***Axis Settings*** from the menu.

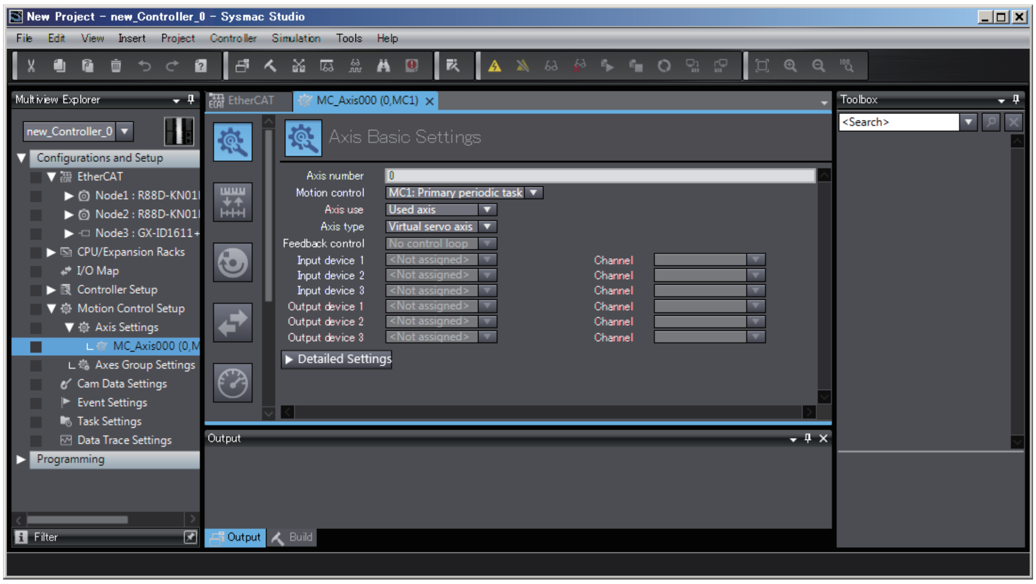
If necessary, you can change the axis variable names from the default names of *MC_Axis***.* ("***" is incremented from 000 in the order that the axis variables are created.)

2 Assign the axes that you created to Servo Drives or encoder input slaves in the EtherCAT Slave Configuration of the Sysmac Studio.

Set the Axis Basic Settings from the Sysmac Studio.

Classification	Parameter name	Setting
Axis Basic Settings	Axis Number	Axis numbers are automatically set in the order that the axes are created.
	Motion Control*1	Select Primary periodic task.
	Axis Use	Select Used axis.
	Axis Type	Select a servo axis or encoder axis.
	Input Device/ Output Device	Specify the node address of the EtherCAT slave that is assigned to the axis.

*1 You can select this parameter for NX-series CPU Units.

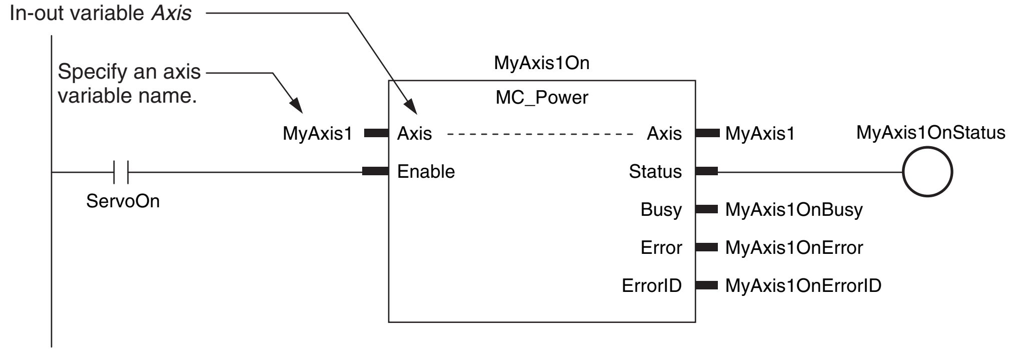


3 Use the Sysmac Studio to specify the settings required for Test Mode operation (Unit Conversion, Count Mode, Limits, etc.) and the settings required for actual system operation. Then transfer the settings to the CPU Unit with the project.

4 In the user program, an axis variable name is specified for the in-out variable *Axis* in motion control instructions.

For the axis variable name, specify the axis name (axis variable name) that was specified in the Motion Control Setup or a system-defined variable. You can execute motion control for the assigned Servo Drive or encoder input slave. An example that specifies the axis variable name *MyAxis1* is shown below.

Example:



Refer to *3-5-2 Axis Variables and Axes Group Variables* for information on Axis Variables.

Precautions for Correct Use

- For an NX-series CPU Unit, use the system-defined variables that correspond to the Motion Control parameter for Axis Settings or Axes Group Settings. A building error will occur if you use the system-defined variables that do not correspond.

Motion Control setting	Corresponding system-defined variable
MC1: Primary periodic task	_MC_AX[***], _MC1_AX[***], _MC_GRP[***], _MC1_GRP[***]
MC2: Priority-5 periodic task	_MC2_AX[***], _MC2_GRP[***]

4

Controller Setup

This section describes the initial settings of the function modules.

4-1 Overview of the Controller Setup

This section provides an overview of the Controller Setup.

The shaded steps in the overall procedure that is shown below are related to the Controller Setup.

Step 1. Software Design

- Step 1-1 Designing I/O and Processing
- Step 1-2 Designing Tasks
- Step 1-3 Designing Programs

↓

Step 2. Software Setups and Programming

- Step 2-1 Slave and Unit Configurations
- Step 2-2 Controller Setup
- Step 2-3 Programming
- Step 2-4 Offline Debugging

↓

Step 3. Mounting and Setting Hardware

↓

Step 4. Wiring

↓

Step 5. Checking Operation and Starting Operation on the Actual System

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series Controller* for details.

Controller Setup	Reference
● **Initial Settings Related to the PLC Function Module:** Controller Setup: Startup Mode, Write Protection, System Service Monitoring Settings, and other settings	*4-2 Initial Settings for the PLC Function Module*
● **Initial Settings for Special Units:** Unit Configuration and Setup: Initial settings for the Special Units	*4-3 Initial Settings for Special Units*
● **Initial Settings for the Motion Control Function Module:** • Axis Parameters: Motion control parameters for single-axis operation • Axes Group Parameters: Motion control parameters for multi-axes coordinated operation • Cam data: Phase and displacement setting tables for cam motions	*4-4 Initial Settings for the Motion Control Function Module*
● **Initial Settings for the EtherCAT Master Function Module:** EtherCAT Master Parameters in the EtherCAT Configuration: Parameter settings for the EtherCAT master process data communications cycle, and other settings	*4-5 Initial Settings for the EtherCAT Master Function Module*
● **Initial Settings for the EtherNet/IP Function Module:** Ethernet Port Setup: EtherNet/IP Port TCP/IP Settings, Ethernet Settings, and other settings	*4-6 Initial Settings for the EtherNet/IP Function Module*

4-2 Initial Settings for the PLC Function Module

This section describes the initial settings that are required for the PLC Function Module.

4-2-1 Introduction

The initial settings for the PLC Function Module are listed below.

- Controller Setup
- Task Settings

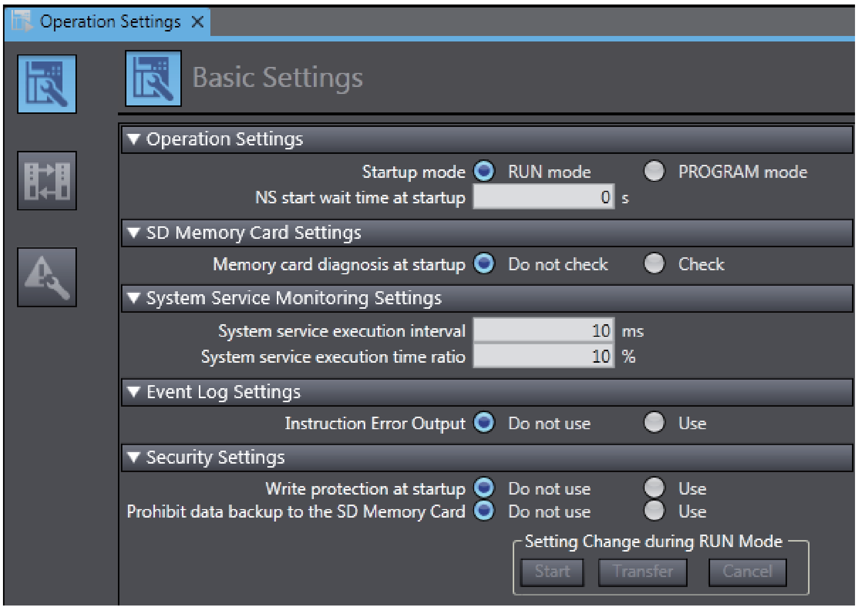
Select **Configurations and Setup** – **Controller Setup** and **Configurations and Setup** – **Task Settings** on the Sysmac Studio to make these settings

4-2-2 Controller Setup

Operation Settings Tab Page

Basic Settings

The Operation Settings are for functions supported by the CPU Unit, such as the definitions of operations when the power is turned ON or when the operating mode changes.



Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
Operation Settings	Startup Mode	Sets the CPU Unit's operating mode at startup.	RUN or PROGRAM mode	RUN mode	When downloaded to CPU Unit	Not allowed.
	NS start wait time at startup*1	Sets the time to wait for an NS-series PT to perform the tag verifications with priority during startup after the power supply is turned ON.*2	0 to 10 s	0 s	When downloaded to CPU Unit	Not allowed.
SD Memory Card Setting	Memory Card Diagnosis at Startup	Sets whether to execute self-diagnosis (file system check and recovery) on the inserted SD Memory Card when the power is turned ON.	Do not check. Check.	Do not check.	When downloaded to CPU Unit	Not allowed.
System Service Monitoring Settings*3	System Service Execution Interval [ms]	Sets the interval of system service execution.	10 ms to 1 s	10 ms	When downloaded to CPU Unit	Not allowed.
	System Service Execution Time Ratio [%]	Sets the ratio of execution for monitoring system services in relation to overall processing of the CPU Unit.	5% to 50%	10%	When downloaded to CPU Unit	Not allowed.
Event Log Settings*4	Instruction Error Output	Sets whether to output events to an event log when instruction errors occur.	Do not use. Use.	Do not use.	When transferred to CPU Unit	Not allowed.
Security Setting	Write Protection at Startup	Sets whether to automatically enable or disable write protection when you turn ON the power supply to the Controller.	Do not use. Use.	Do not use.	When power is turned ON	Supported.
	Prohibit data backup to the SD Memory Card*5	Sets whether to enable or disable SD Memory Card backups.	Do not use. Use.	Do not use.	When downloaded to CPU Unit	Supported.

*1 This setting is enabled when an NS-series PT is connected to the built-in EtherNet/IP port on the CPU Unit, and the power supplies for these devices are turned ON simultaneously.
A CPU Unit with unit version 1.10 or later and Sysmac Studio version 1.12 or higher are required to set the NS start wait time at startup.

*2 The processing time for verifying tags of an NS-series PT can be reduced with this setting. Set the value to 10 if you want to give priority to the tag verifications. Otherwise, set the value to 0.
If you set the value to 10, after the power supply is turned ON, the CPU Unit gives priority to the tag verifications of the NS-series PT for approximately 10 seconds during startup before the Unit changes the startup state to the normal operation state. The time to complete the tag verifications can be reduced by performing a part of processing of the tag verifications with priority during startup.
If you specify the value between 1 and 10, the time until the CPU Unit changes the state to the normal operation state is increased because the Unit gives priority to the tag verifications for the specified time regardless of whether an NS-series PT is used. Set the value to 0 if an NS-series PT is not connected, or if you do not turn ON the power supplies for the NS-series PT and the CPU Unit simultaneously.

*3 For NX-series CPU Units, the System Service Monitoring Settings are not provided.

*4 A CPU Unit with unit version 1.02 or later and Sysmac Studio version 1.03 or higher are required to use the Event Log Settings.

*5 A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to disable backups.

Operation item	Description	
Setting Change during RUN Mode	You can change the set values of parameters that can be changed during RUN mode.	
	Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for the operations.	
	Start:	Enables writing set values.
	Transfer:	Transfers set values to the Controller. The set values are overwritten.
	Cancel:	Prohibits writing set values.

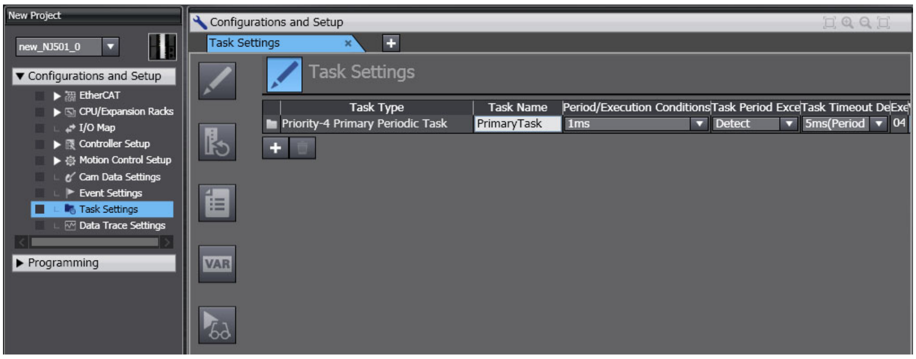
Precautions for Correct Use

If *Use* is selected for *Event Log Settings – Instruction Error Output*, an instruction error is output each time an error occurs when an instruction with an error is executed repetitively. This may cause the event log to exceed the maximum number of events. If this occurs, older events are overwritten.

4-2-3 Task Settings

● Task Settings

The Task Settings are used to add and set up tasks.



Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
Task Type		Sets the task type.	Primary periodic task Priority-5 periodic task*1 Priority-16 periodic task Priority-17 periodic task Priority-18 periodic task Priority-8 event task Priority-48 event task	Primary periodic task	When downloaded to CPU Unit	Not allowed.
	Execution Priority	Sets the task execution priority.	Automatically set according to the task type.	Primary periodic task: 4	When downloaded to CPU Unit	Not allowed.
Task Name		Sets the task name.	Text string	Primary periodic task: PrimaryTask Periodic tasks: PeriodicTask 0 Event tasks: EventTask0	When downloaded to CPU Unit	Not allowed.

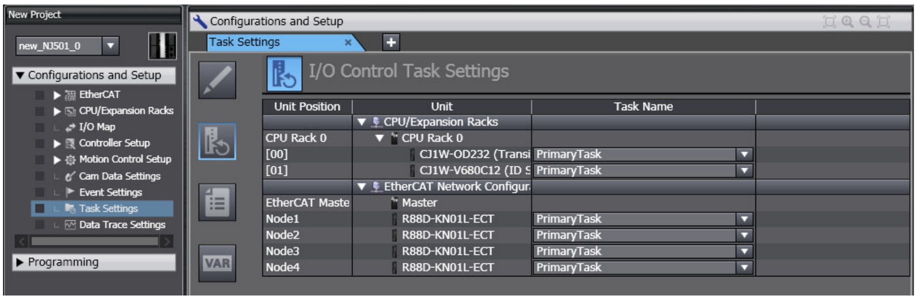
Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
Period/Execution Conditions		Sets the task period.	Primary periodic task: Refer to *5-3-1 Specifications of Tasks for NX-series Controllers*. Periodic tasks: Refer to *5-3-1 Specifications of Tasks for NX-series Controllers*. Event tasks: Executed with an instruction or when a variable expression is satisfied.	Primary periodic task: 1 ms*2 Periodic tasks: 2 ms (Priority 5) 10 ms (Priority 16, 17, and 18) Event tasks: Execution with an instruction	When downloaded to CPU Unit	Not allowed.
Task Period Exceeded Detection		Sets whether to detect an error when the task period is exceeded.	• Detect. (Minor fault level Controller error generated.) • Do not detect. (Store an observation level log record.)	Primary periodic task and periodic tasks: Detect Event tasks: Do not detect (cannot be changed).	When downloaded to CPU Unit	Not allowed.
Task Timeout Detection Time		Sets the task execution timeout time. A Task Execution Timeout occurs when the timeout time is exceeded.	Primary periodic task and periodic tasks: Task period × 1 to Task period × 5 Event tasks: Execution priority of 8: 1 to 500 ms Execution priority of 48: 1 ms to 10 s	Primary periodic task and periodic tasks: Task period × 5 Event tasks: Execution priority of 8: 200 ms Execution priority of 48: 1 s	When downloaded to CPU Unit	Not allowed.
Variable Access Time [%]		Sets the percentage of the task period to assign to variable access from outside the Controller.	Primary periodic task and periodic tasks: 1% to 50% Event tasks: None	3%	When downloaded to CPU Unit	Not allowed.

*1 You can use the priority-5 periodic task only with NX-series CPU Units.

*2 For an NJ101-□□□□ CPU Unit, the default of the primary periodic task is 2 ms.

● I/O Control Task Settings

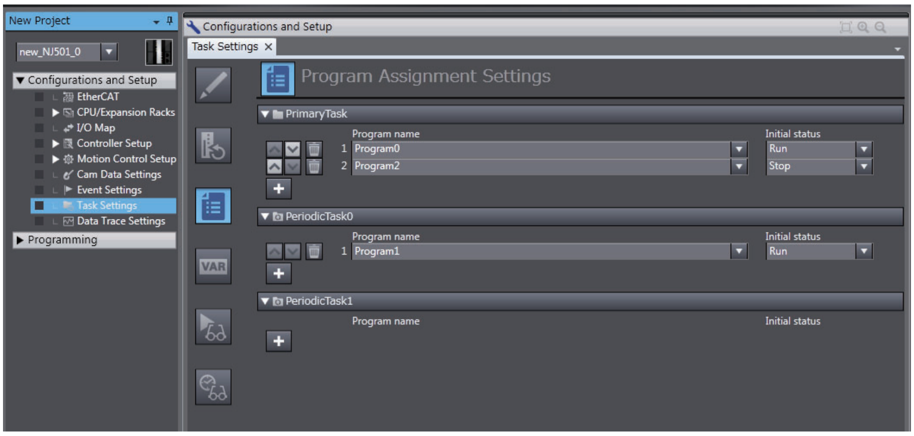
The I/O Control Task Settings are used to set the timing of refresh execution of inputs and outputs.



Parameter	Description	Set value	Default	Update timing	Changes in RUN mode
Task Name	Sets the task to use to refresh the specified Units or slaves.	PrimaryTask or PeriodicTask	PrimaryTask	When downloaded to CPU Unit	Not allowed.

● Program Assignment Settings

The Program Assignments Settings are used to assign the programs to tasks, set the program execution order, and set the operation of the programs at the start of operation.



Parameter	Description	Set value	Default	Update timing	Changes in RUN mode
Program Execution Order	Assigns the programs to the specified tasks and sets the order of program execution within the tasks.	Assign the programs in the order to execute them from top to bottom.	Program0	When downloaded to CPU Unit	Not allowed.
Initial Status of Program*1	Set whether to run the program at the start of operation.	Run Stop	Run	When downloaded to CPU Unit	Not allowed.

*1 A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required.

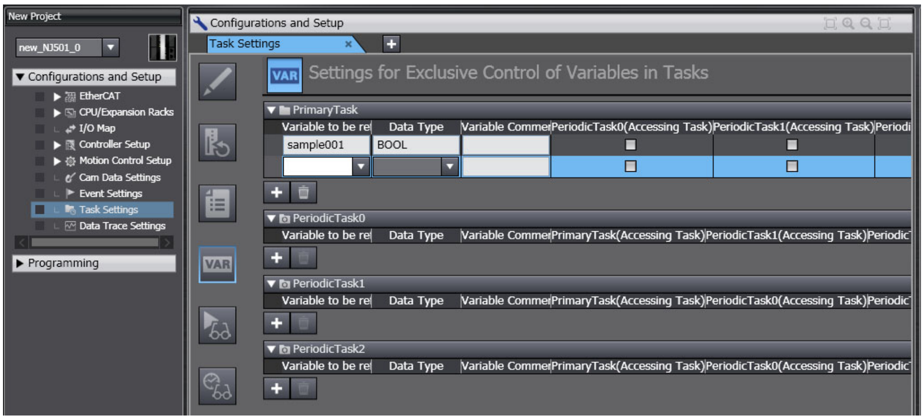
Precautions for Correct Use

- A program that contains the device variables used to access slaves and Units must be assigned to the same task as the one to which the slaves and Units that are set in the I/O Control Task Settings are assigned. A building error will occur if you assign the program to other tasks.
- The default task set in the I/O Control Task Settings is the primary periodic task. If you want to assign the program to a task other than the primary periodic task, change the setting in the I/O Control Task Settings in advance to prevent a building error.
- For an NX-series CPU Unit, a program used to control slaves and Units that are assigned to axes must be assigned to the tasks that correspond to the Motion Control parameter for Axis Settings or Axes Group Settings. A building error will occur if you assign the program to the tasks that do not correspond.
  Use the system-defined variables that correspond to the setting values of the Motion Control parameter when you specify axes or axes groups in programs.

Motion Control setting	Corresponding task to program assignment	Corresponding system-defined variable
MC1: Primary periodic task	Primary periodic task Priority-16 periodic task	_MC_AX[], _MC1_AX[], _MC_GRP[], _MC1_GRP[]
MC2: Priority-5 periodic task	Priority-5 periodic task	_MC2_AX[], _MC2_GRP[]

● Settings for Exclusive Control of Variables in Tasks

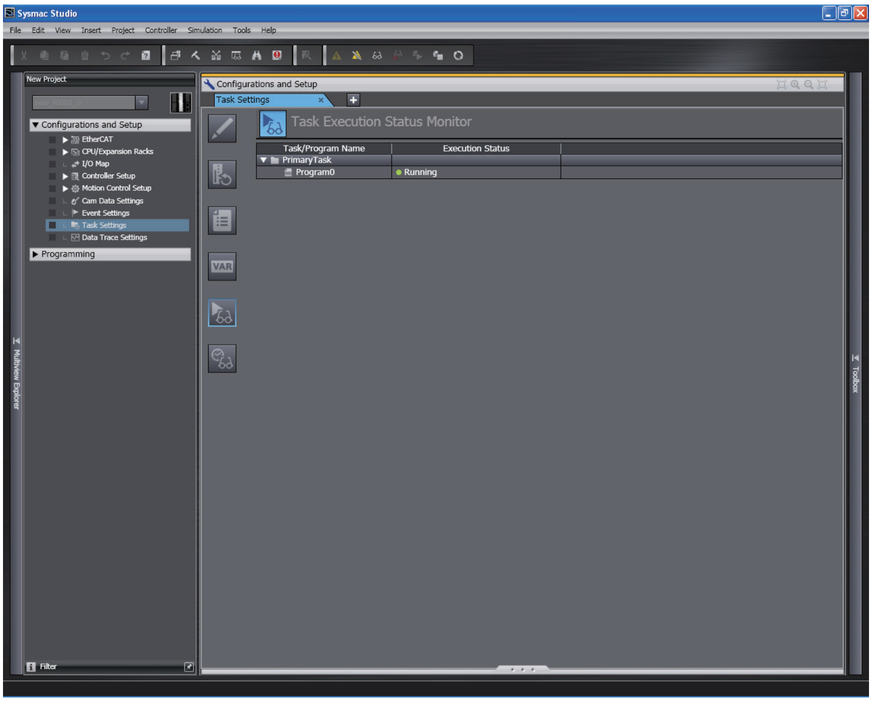
The Settings for Exclusive Control of Variables in Tasks are used to set the tasks that refresh specified global variables and the tasks that access specified global variables.



Item	Parameter	Description	Set value	Default	Update timing	Changes in RUN mode
Each Task	Variables to be refreshed	Sets the variables to refresh in the primary periodic task or periodic task.		None	When downloaded to CPU Unit	Not allowed.
	Data Type	Sets the data type of variable.	None			
	Variable Comment	Sets a comment for the variable.	None			
	Accessing Task	Sets the tasks that access the variable.				

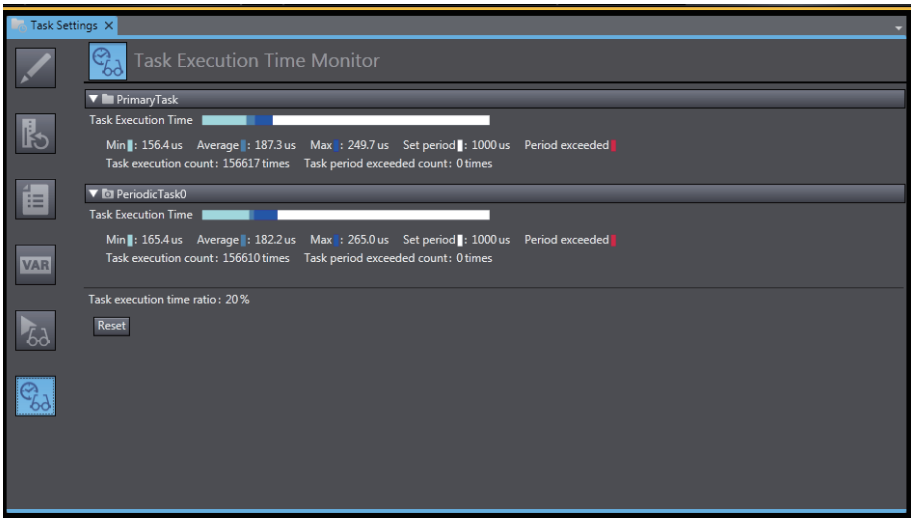
● Task Execution Status Monitor

The Task Execution Status Monitor displays the execution status of the programs.



● Task Execution Time Monitor

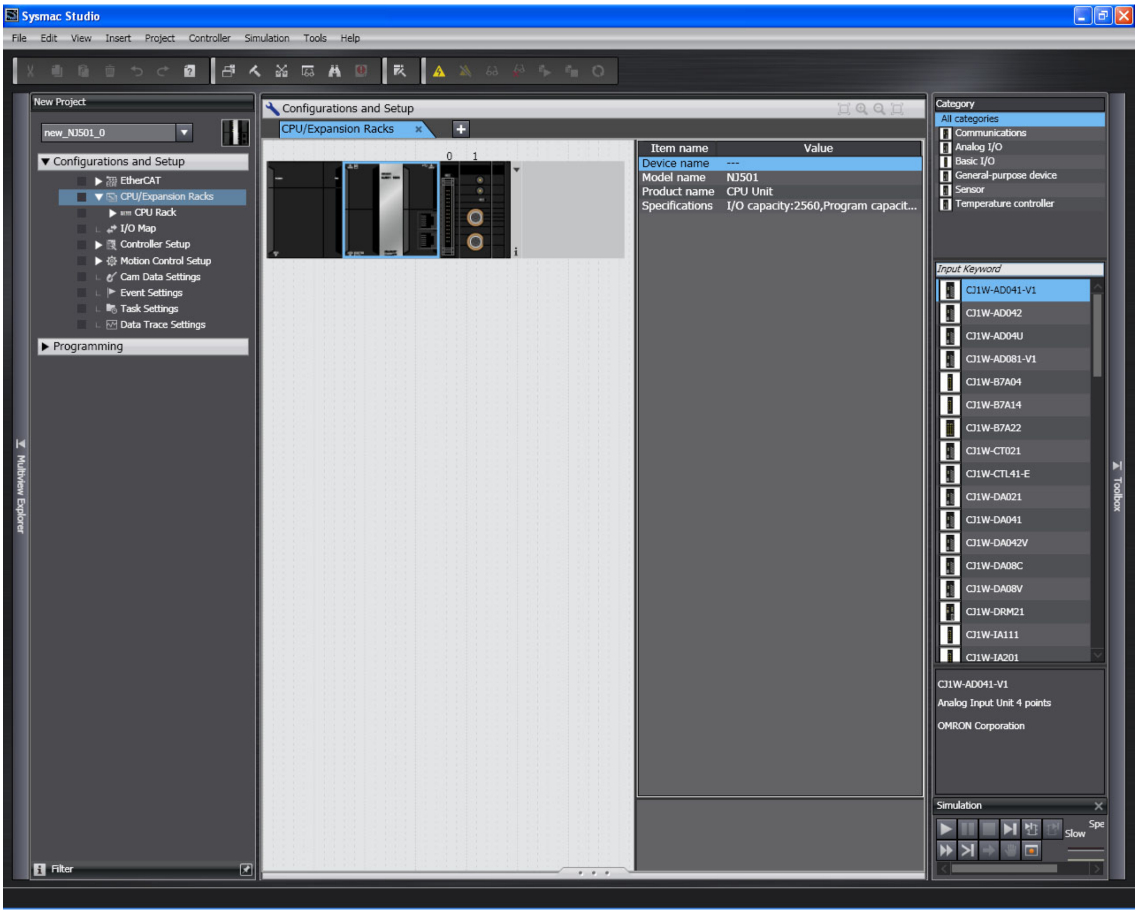
The Task Execution Time Monitor displays the execution times of the tasks.



4-2-4 Unit Configuration and Unit Setup

This section describes the Unit settings for NJ-series CPU Units. These settings are not provided for NX-series CPU Units.

● Unit Information



● Settings for All Units

Set the device names.

Device names are automatically created when Units are added in the Unit Editor.

Default names: “J” followed by serial numbers that start from 01

We recommend that you change the name to one that is suitable to the device.

Additional Information

The device names that are set here are placed before the I/O port name when device variables are automatically created.

● Special Units

Set the unit numbers of the Special Units.

Precautions for Correct Use

Make sure you set the same unit numbers as the unit numbers that are set on the rotary switches on the front of the Special Units. If they are not the same, operation will be according to the unit numbers that are set on the front-panel rotary switches.

● Basic I/O Units

The following settings are made in the Unit Information of the Basic I/O Units.

Access point	Setting group	Description	Set values	Default	Update timing	Changes in RUN mode
Unit Information **Note** Set the information for each slot.	Basic Input Unit Input Response Time	Sets the input response time (ON response time = OFF response time) of the Basic Input Unit. You can set no filter or you can set the value in increments from 0 to 32 ms. You can increase the value to reduce chattering and the effects of external noise. If you decrease the value, shorter input pulses are received (but the pulses must be longer than the task period).	No filter 0.5 ms, 1 ms, 2 ms, 4 ms, 8 ms, 16 ms, or 32 ms	8 ms	When power is turned ON or the CPU Unit is reset	Not allowed.

4-3 Initial Settings for Special Units

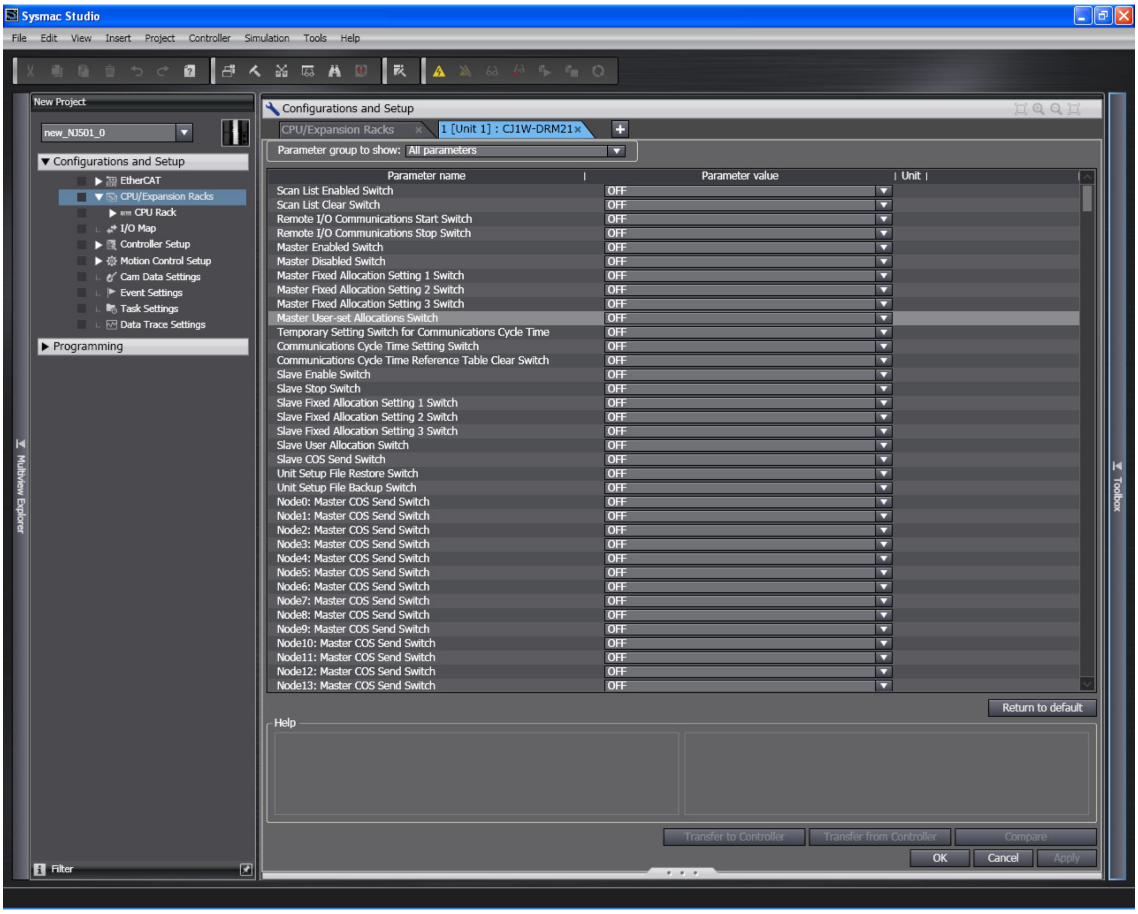
This section describes the initial settings that are required for the Special Units used with NJ-series CPU Units. These settings are not provided for NX-series CPU Units.

You can use any of the following methods to set the initial settings of the Special Units.

Method 1: Setting from the Unit Setting Pane of the Sysmac Studio

1 Select the Unit in the Unit Configuration and Setup.

2 Specify the settings in the Unit Settings Tab Page shown below.



3 Connect the CPU Unit online and transfer the settings to the CPU Unit.

Method 2: Using the Sysmac Studio to Specify Initial Settings for the I/O Ports in the I/O Map

1 Use the I/O Map in the Sysmac Studio to set values for the I/O ports.

2 Restart the Unit, reset the Controller, or cycle the power supply to the Controller.

Method 3: Using the Sysmac Studio to Specify Initial Settings for the Device Variables of the CJ-series Units

1 Use the Sysmac Studio to specify the initial values for the device variables of the CJ-series Units.

2 Download the variable table from the Sysmac Studio to the CPU Unit.

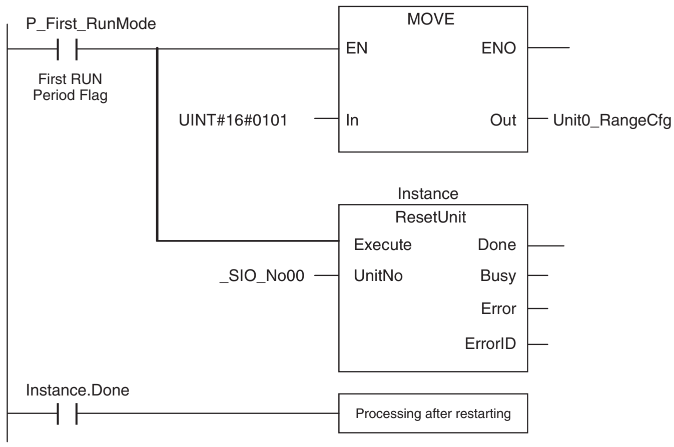
Select the *Clear the present values of variables with Retain attribute* Check Box.

3 Restart the Unit, reset the Controller, or cycle the power supply to the Controller.

Method 4: Using Instructions to Set the Device Variables for the CJ-series Units

1 Set the values for the device variables for the CJ-series Unit at the start of operation from the user program (e.g., use the MOVE instruction) and then restart the Unit.

Example:



Precautions for Safe Use

When you restart a Special Unit after you change the settings, confirm the safety of the devices at the connection target before you restart the Unit.

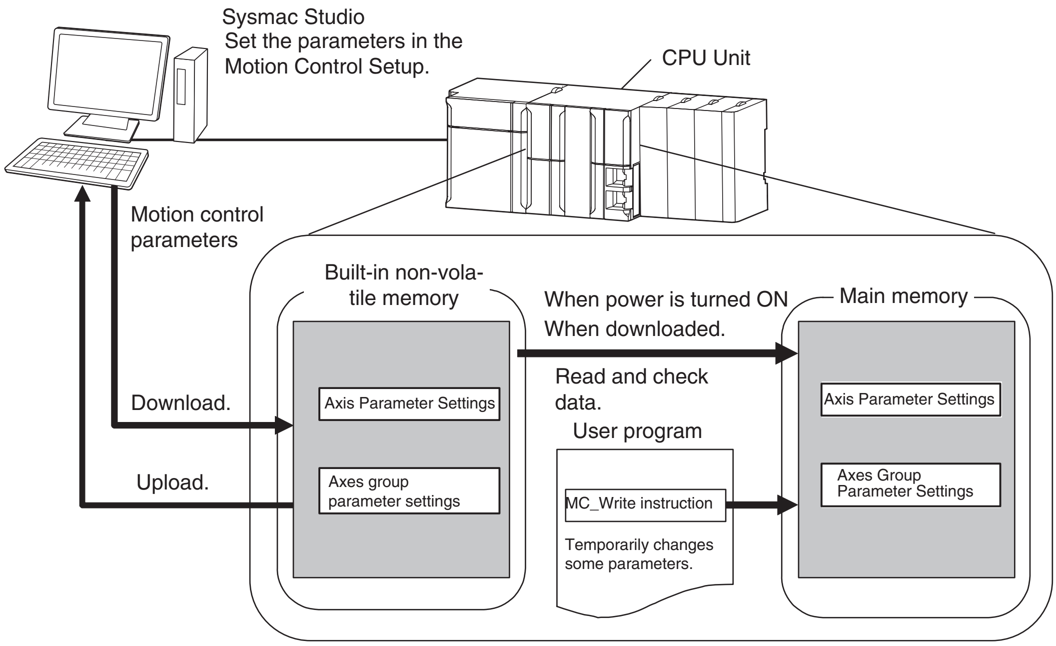
4-4 Initial Settings for the Motion Control Function Module

This section describes the initial settings that are required for the MC Function Module.

4-4-1 Introduction

The initial settings for the Motion Control Function Module are called motion control parameters. Motion control parameters include the following parameters.

- Axis Parameters: Settings for single-axis control
- Axes Group Parameters: Settings for multi-axes coordinated control

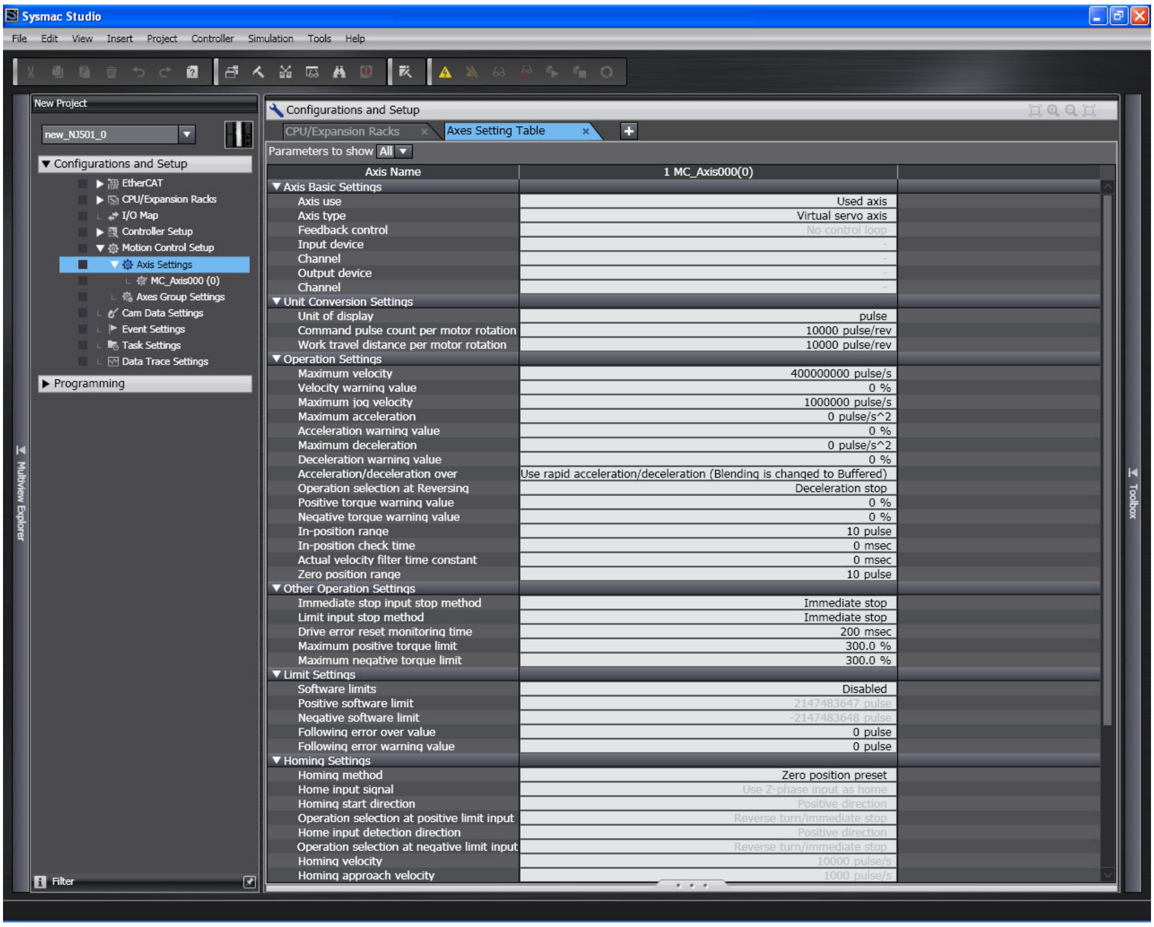


4-4-2 Setting Methods

You can use either of the following methods to set motion control parameters.

Method 1: Setting the Motion Control Setup in the Sysmac Studio

Right-click ***Axis Settings*** from under **Configurations and Setup - Motion Control Setup** in the Sysmac Studio and make the settings in the Axis Setting Table.



Download the motion control parameters to the CPU Unit to save them in the non-volatile memory in the CPU Unit. The downloaded settings are enabled when the power is turned ON or a download is performed.

Method 2: Setting with the MC_Write Instruction

You can temporarily overwrite some motion control parameters with the MC_Write instruction.

Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details.

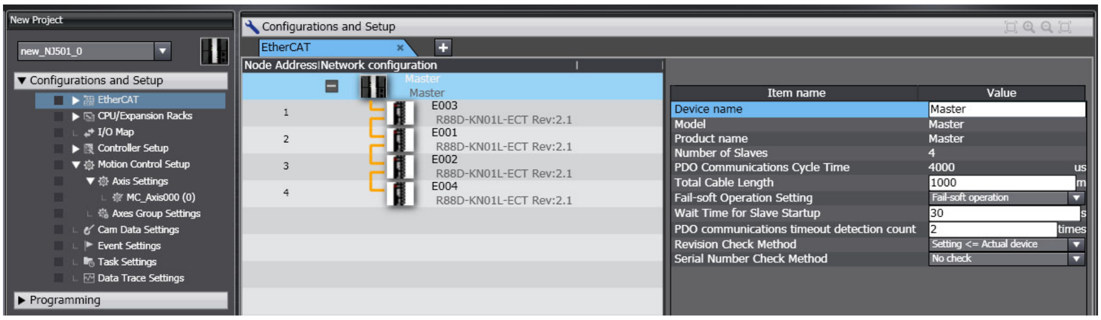
4-5 Initial Settings for the EtherCAT Master Function Module

This section describes the initial settings that are required for the EtherCAT Master Function Module.

The initial settings for the EtherCAT Master Function Module are listed below.

- Device names
- Total Cable Length
- Fail-soft Operation Settings
- Wait Time for Slave Startup
- PDO Communications Timeout Detection Count
- Revision Check Method
- Serial Number Check Method

Double-click **EtherCAT** under **Configurations and Setup** and then select the master on the Sysmac Studio. The Initial Setting Tab Page for the EtherCAT Master Function Module is displayed.



Refer to the *NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual* (Cat. No. W505) for details.

4-6 Initial Settings for the EtherNet/IP Function Module

This section describes the initial settings that are required for the EtherNet/IP Function Module.

The initial settings for the EtherNet/IP Function Module are listed below.

- TCP/IP Settings
- Link Settings
- FTP Settings
- NTP Settings
- SNMP Settings
- SNMP Trap Settings
- FINS Settings*1

*1 For NX-series CPU Units, FINS Settings are not provided.

Select **Configurations and Setup** – **Controller Setup** – **Built-in EtherNet/IP Port Settings** on the Sysmac Studio to make these settings

Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP User's Manual* (Cat. No. W506) for details.

5

Designing Tasks

This section describes the task system and types of tasks.

5-1 Overview of Task Designing Procedure

This section provides an overview of the task designing procedure.

The shaded steps in the overall procedure that is shown below are related to the task designing procedure.

Step 1. Software Design

- Step 1-1 I/O Processing Design
- Step 1-2 Designing Tasks
- Step 1-3 Designing Programs

Step 2. Software Setups and Programming

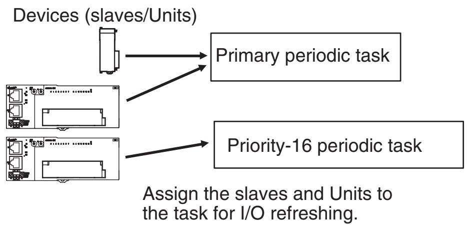
- Step 2-1 Slave and Unit Configurations
- Step 2-2 Controller Setup
- Step 2-3 Programming
- Step 2-4 Offline Debugging

Step 3. Mounting and Setting Hardware

Step 4. Wiring

Step 5. Checking Operation and Starting Operation on the Actual System

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series Controller* for details.

Designing the Tasks	Reference
● **Design the task configuration.** Design the task configuration based on the I/O response performance that is required by the controlled devices.	*5-3-3 Basic Operation of Tasks for NX-series Controllers* *5-4-3 Basic Operation of Tasks for NJ-series Controllers* *5-10 Task Design Methods and I/O Response Times*
● **Determine whether to use the primary periodic task, or the priority-5 or priority-16 periodic task for the I/O refreshing of each Unit and slave.**  	*5-6-1 Assigning I/O Refreshing to Tasks*
● **Determine which programs to assign to the primary periodic task, to the priority-5, priority-16, priority-17, and priority-18 periodic tasks, and to the priority-8 and priority-48 event tasks.**	*5-6-2 Assigning Tasks to Programs*
● **Design the exclusive control methods for variables between tasks.** Design the exclusive control methods for variables between tasks when the same global variables are used in different tasks.	*5-7-1 Ensuring Concurrency of Variable Values between Tasks*
● **Design the tasks to access variables from outside of the Controller.** Design the tasks to enable synchronization of accessing variables in the CPU Unit from outside of the Controller with the execution of a program in a specific task. EtherNet/IP tag data links are included in accessing variables.	*5-7-2 Variable Access from Outside the Controller*

Task Settings on the Sysmac Studio

Setting the Tasks	Reference
● **Initial Settings for the PLC Function Module:** Task Settings: Task Periods, I/O Settings, Program Assignments, Task Interface Settings, and other settings	*4-2 Initial Settings for the PLC Function Module*

Offline Debugging with the Sysmac Studio

Desktop Operation Check	Reference
● **Perform desktop debugging of sequence control and motion control with the Simulator (virtual controller).**	*Section 7 Checking Operation and Actual Operation*
● **Monitor the task execution times in the Task Execution Time Monitor Display.**	*5-9 Monitoring Task Execution Status and Task Execution Times*

5-2 Overview of Tasks

This section provides an overview of tasks.

5-2-1 Tasks

Tasks are used to assign an execution condition and execution order to a series of processes, such as I/O refreshing and user program execution.

There are three kinds of tasks, as shown in the following table. They are defined by their execution conditions and execution priorities.

Type of task	Number of tasks	Task execution priority	Definition	Main processing content
Primary periodic task	1	4	The primary periodic task is executed once every task period. It has the highest execution priority. Motion control instructions and EtherCAT communications of the primary periodic task are executed on the primary periodic task period.	I/O refreshing, user program execution, and motion control
Periodic tasks	0 to 1	5*1	The priority-5 periodic task is executed once every task period. It has the second highest execution priority after the primary periodic task. Motion control instructions and EtherCAT communications of the priority-5 periodic task are executed on the priority-5 periodic task period.	I/O refreshing, user program execution, and motion control
	0 to 3	16, 17, or 18	The priority-16, priority-17, and priority-18 periodic tasks are executed once every task period. Motion control instructions and EtherCAT communications of the priority-16 periodic task are executed on the primary periodic task period.	The processing that can be performed depends on the task execution priority. • Execution priority 16: I/O refreshing and user program execution • Execution priority 17 or 18: User program execution
Event tasks	0 to 32	8 or 48	An event task is executed only once when the specified execution condition is met.	User program execution

*1 You can use the priority-5 periodic task only with NX-series CPU Units.

Version Information

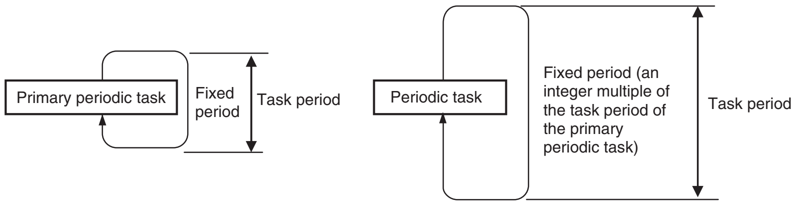
A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to use event tasks.

Additional Information

With an NX-series CPU Unit, you can execute motion control in the primary periodic task and in the priority-5 periodic task. If these two motion controls need to be identified, the motion control in the primary periodic task is called motion control 1, while the motion control in the priority-5 periodic task is called motion control 2.

● Primary Periodic Task and Periodic Tasks

The CPU Unit periodically executes both the primary periodic task and periodic tasks.
(The interval in which the CPU Unit executes the primary periodic task or a periodic task is called the task period.)



From 1 to 128 programs can be assigned to one task. The programs that are assigned to a task are executed in the order that they are assigned. Execution of the all of the programs assigned to each task is called user program execution.

Exchanging data with CJ-series Units or EtherCAT slaves is called I/O refreshing.

You can assign I/O refreshing for each slave and Unit to the primary periodic task or the priority-5 or priority-16 periodic task. By default, I/O refreshing for all slaves and Units is assigned to the primary periodic task.

● Event Tasks

An event task is executed only once when the specified execution condition is met. There are the following two types of execution conditions for event tasks.

Execution condition	Specification
Execution with an instruction	The event task is executed when the ActEventTask (Execute Event Task) instruction is executed.
Execution when a condition for a variable is met	The event task is executed when the specified variable matches a predefined condition.

From 1 to 128 programs can be assigned to one task. The programs that are assigned to a task are executed in the order that they are assigned.

Precautions for Correct Use

- I/O refreshing and motion control are not executed in event tasks. This means that you cannot assign programs to event tasks if the program performs I/O control or executes motion control instructions.
- Event tasks are not executed repeatedly every task period. Therefore, you cannot assign a program to an event task if that program contains an instruction whose execution is not completed within one task period. Instructions that are executed over more than one task period include some of the basic instructions, such as instructions for SD Memory Cards and communications, all motion control instructions, and all simulation instructions. Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for details on the basic instructions that cannot be used in event tasks.

Additional Information

Tasks operate differently between NX-series and NJ-series CPU Units. Refer to *5-3 Specifications and Basic Operation of Tasks for NX-series Controllers* and *5-4 Specifications and Basic Operation of Tasks for NJ-series Controllers* for details.

5-2-2 Instructions Related to Tasks

The following instructions are supported to read the status of the current task, to determine if execution is in progress for other tasks, and to perform exclusive control for regional concurrency between tasks.

<table>
<tr><th>Instruction</th><th>Instruction name</th><th colspan="2">Introduction</th></tr>
<tr><td>GetMyTaskStatus</td><td>Read Current Task Status</td><td colspan="2">Reads the following status of the current task.
Last Task Execution Time, Maximum Task Execution Time, Minimum Task Execution Time, Task Execution Count, Task Period Exceeded Flag, and Task Period Exceeded Count</td></tr>
<tr><td>GetMyTaskInterval</td><td>Read Current Task Period</td><td colspan="2">Reads the task period of the current task.</td></tr>
<tr><td>Task_IsActive</td><td>Determine Task Status</td><td colspan="2">Determines if the specified task is currently in execution.</td></tr>
<tr><td>Lock</td><td>Lock Tasks</td><td>Starts a lock between tasks.</td><td rowspan="2">Execution of any other task with a lock region with the same lock number is disabled.</td></tr>
<tr><td>Unlock</td><td>Unlock Tasks</td><td>Stops a lock between tasks.</td></tr>
<tr><td>ActEventTask</td><td>Activate Event Task</td><td colspan="2">Activates the specified event task.</td></tr>
</table>

5-2-3 System-defined Variables Related to Tasks

The following system-defined variables are provided for each task to show task status.
Do not use these variables in the user program. There may be a delay in updating them and concurrency problems in relation to the error status of the Function Module. It is used only to sample the task status for data tracing from the Sysmac Studio.

You can also use the GetMyTaskStatus and Task_IsActive instructions to read task status from the user program. You cannot access the following variables directly through system-defined variables.

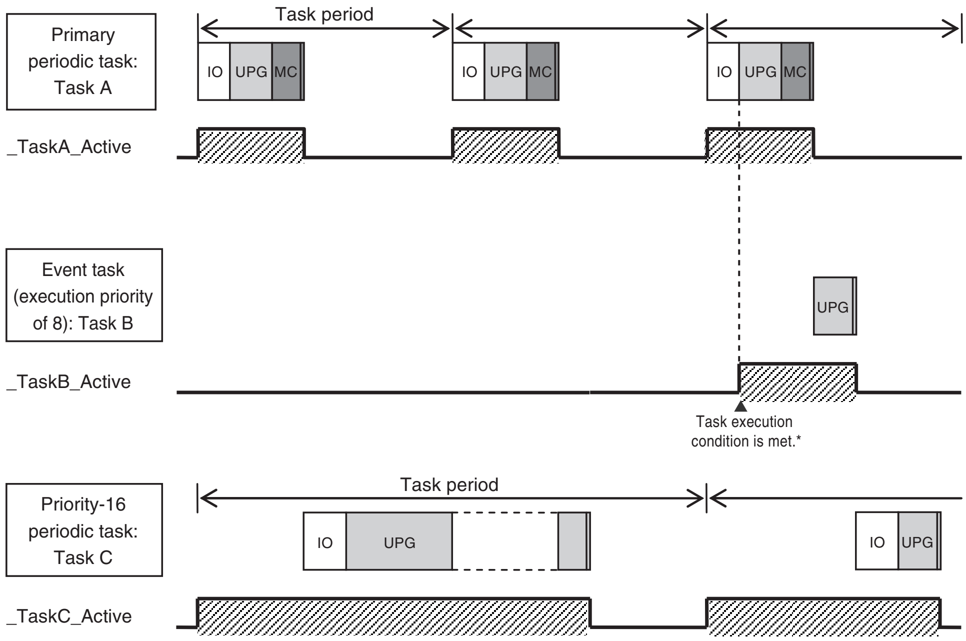
Variable name	Meaning	Description	Data type	R/W
_*TaskName*_Active	Task Active Flag	TRUE during task execution. FALSE from the completion of task execution until the end of the task period.	BOOL	R
_*TaskName*_LastExecTime	Last Task Execution Time	Gives the last execution time of the task.	TIME	R
_*TaskName*_MaxExecTime	Maximum Task Execution Time	Gives the maximum value of the task execution time.	TIME	R
_*TaskName*_MinExecTime	Minimum Task Execution Time	Gives the minimum value of the task execution time.	TIME	R

Variable name	Meaning	Description	Data type	R/W
_*TaskName*_ExecCount	Task Execution Count	Contains the number of executions of the task. If the present value exceeds the maximum value of the data type, the present value returns to 0 and the count is continued.	UDINT	R
_*TaskName*_Exceeded	Task Period Exceeded Flag	TRUE when task execution is completed if the task period is exceeded. FALSE if task execution was completed within the task period.	BOOL	R
_*TaskName*_ExceedCount	Task Period Exceeded Count	Contains the number of times that the task period was exceeded. If the present value exceeds the maximum value of the data type, the present value returns to 0 and the count is continued.	UDINT	R

Note Example: The Task Period Exceeded Flag for the task named MainTask is *_MainTask_Exceeded*.

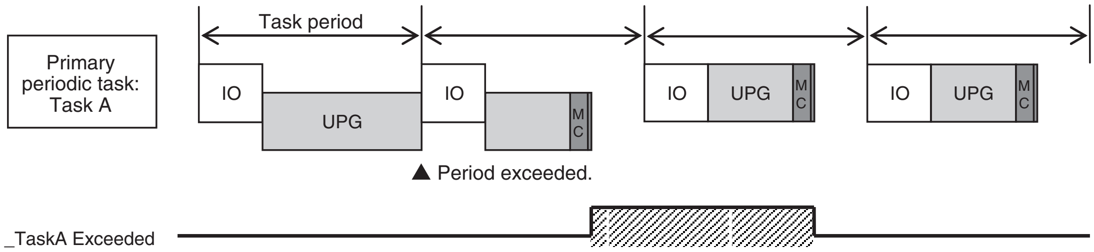
Flag Operation

● Task Active Flag (_*TaskName*_Active)



* When the ActEventTask instruction is used to execute an event task, the Task Active Flag changes to TRUE.

- **Task Period Exceeded Flag (_*TaskName*_Exceeded)**



5-3 Specifications and Basic Operation of Tasks for NX-series Controllers

This section describes the specifications and basic operation of tasks for NX-series CPU Units with a multi-core processor.

5-3-1 Specifications of Tasks for NX-series Controllers

The specifications of tasks are given in the following table.

Item	Specification
Type of task	• Primary periodic task • Periodic task (priority 5, 16, 17, or 18) • Event task (priority 8 or 48)
Numbers of tasks	• Primary periodic task: 1 • Periodic tasks: 0 to 4 tasks*1 • Event tasks: 0 to 32 tasks*2
Number of programs per task	128 max.
Task period of the primary periodic task	125 μs 250 μs to 8 ms (in 250-μs increments)
Task periods of periodic tasks	• Priority 5 125 μs 250 μs to 100 ms (in 250-μs increments) • Priority 16, 17, or 18 1 ms to 100 ms (in 250-μs increments) Set the task period of each periodic task to an integer multiple of the task period of the primary periodic task. You cannot select any combination of task periods whose least common multiple exceeds 600 ms.

*1 There can be no more than one task with each of the following execution priorities: 5, 16, 17, and 18.
*2 There can be up to 32 tasks with each of the following priorities as long as there are no more than a total of 32 tasks with these priorities: 8 and 48.

Precautions for Correct Use

Do not set the task period of primary periodic task to 4 ms or more when you use the priority-5 periodic task. If you set the task period to 4 ms or more, a Slave Application Error may occur.

5-3-2 Guidelines for Separating Tasks for NX-series Controllers

All programs must be assigned to one of the tasks. Use the guidelines in the following table to determine which tasks to assign your programs to based on the requirements of the programs.

Task	Programs that are suitable for this task
Primary periodic task	• Programs that require periodic I/O refreshing, user program execution, motion control, or system common processing at an exact execution period. • Programs that require the highest execution priority and contain controls that need high-speed response. • Programs that contain motion control instructions with the highest execution priority.
Priority-5 periodic task	• Programs that require periodic I/O refreshing, user program execution, motion control, or system common processing at an exact execution period. • Programs that require the second highest execution priority after the primary periodic task and contain controls that need high-speed response. • Programs that contain motion control instructions with a relatively high execution priority. • Programs used for applications where the processes that require the highest-speed processing are controlled with the primary periodic task and the rest is controlled separately with the priority-5 periodic task.
Priority-16 periodic task	• Programs with a relatively low execution priority that require periodic user program execution or system common processing. • Programs that contain controls for some slaves and Units whose I/O refreshing is assigned to the primary periodic task. • Programs that contain motion control instructions with a relatively low execution priority. • Programs used for applications where, among the processes for slaves and Units controlled with the primary periodic task, those with a relatively low execution priority are controlled separately with the priority-16 periodic task.
Priority-17 or priority-18 periodic task	• Programs with a relatively low execution priority that require periodic user program execution or system common processing. • Programs that contain data processing and communications processing controls that do not need high-speed response.
Event Task	• Programs that are executed only when specified conditions are met.

5-3-3 Basic Operation of Tasks for NX-series Controllers

With a multi-core processor, NX-series CPU Units can execute the primary periodic task and the priority-5 periodic task in parallel. The order in which tasks are executed depends on the execution priority that is set for each task.

Additional Information

With an NX-series CPU Unit, you can execute multiple tasks, the tag data link service, and system services in parallel.

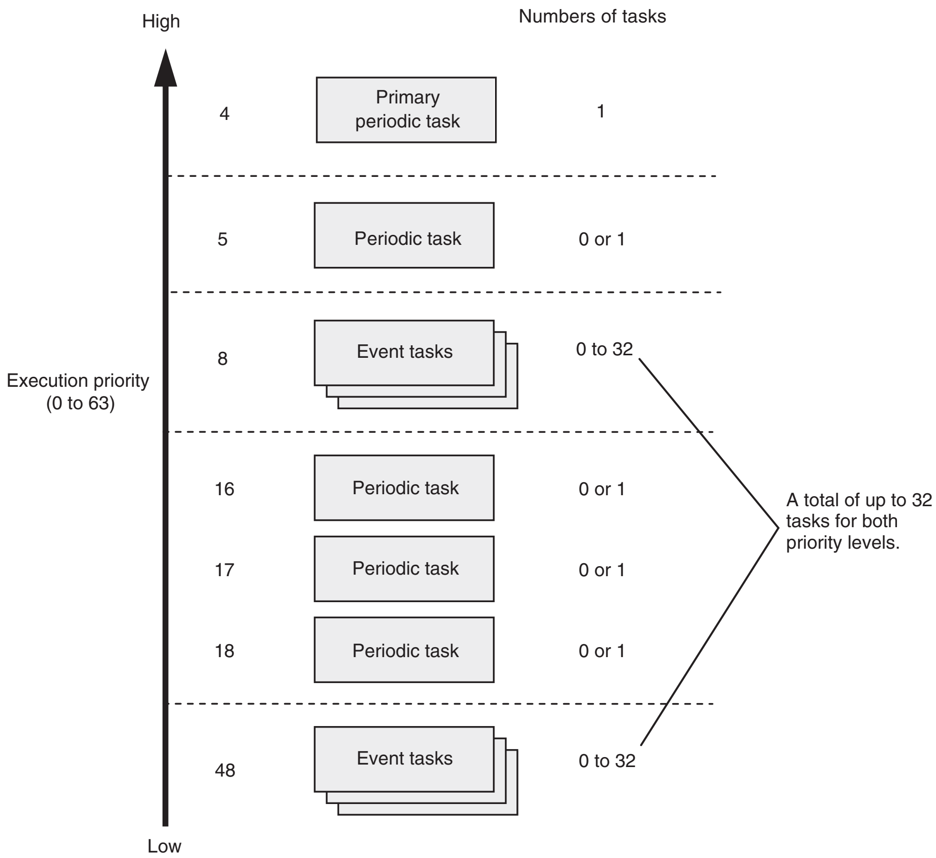
Task Execution Priority

The type of the task determines its execution priority. The CPU Unit executes the task with the highest execution priority first.

If the execution condition is met for another task, Tb, that has a higher execution priority while task Ta execution is in progress, the NX-series CPU Unit will assign Tb to the available core for processing on a priority basis.

The execution priority for each task type is given in the following table. The smaller the value of the execution priority, the higher the priority.

Task	Execution priority	Tasks with the same execution priority
Primary periodic task	4	---
Periodic task	5, 16, 17, or 18	You cannot set the same execution priority for more than one task.
Event task	8 or 48	You can set the same execution priority for more than one event task. Refer to *5-3-5 Event Task Execution Timing for NX-series Controllers* for the order of execution.



Task Periods for the Primary Periodic Task and Periodic Tasks

The CPU Unit repeatedly and cyclically executes the primary periodic task and periodic tasks. The task periods for periodic tasks must be assigned as integer multiples of the task period of the primary periodic task (called the primary period). Therefore, execution of both tasks will be start at the same time every few cycles.

For example, if the primary period is set to 1 ms and the task period of the priority-16 periodic task is set to 4 ms, the execution timing of the primary periodic task and the priority-16 periodic task is synchronized after each four executions of the primary periodic task.

Additional Information

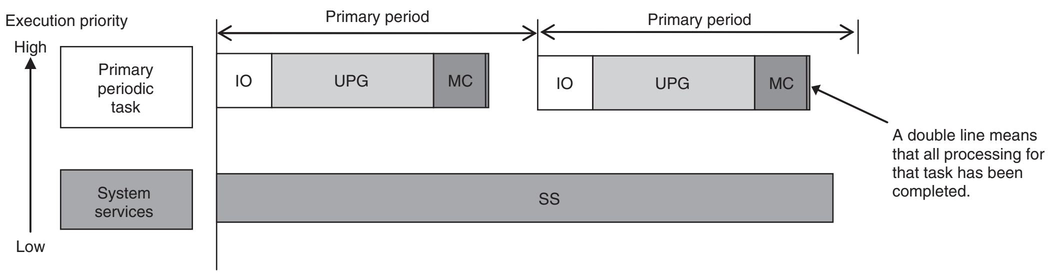
An event task is not executed periodically. Instead, it is executed only once when the specified execution condition is met. Therefore, execution of an event task depends on when its execution condition is met and on its execution priority.

Examples of Execution Order for Tasks

This section gives a few examples of the execution order for the primary periodic task and periodic tasks. Refer to *5-3-5 Event Task Execution Timing for NX-series Controllers* for the order of execution of event tasks.

● Projects with Only the Primary Periodic Task

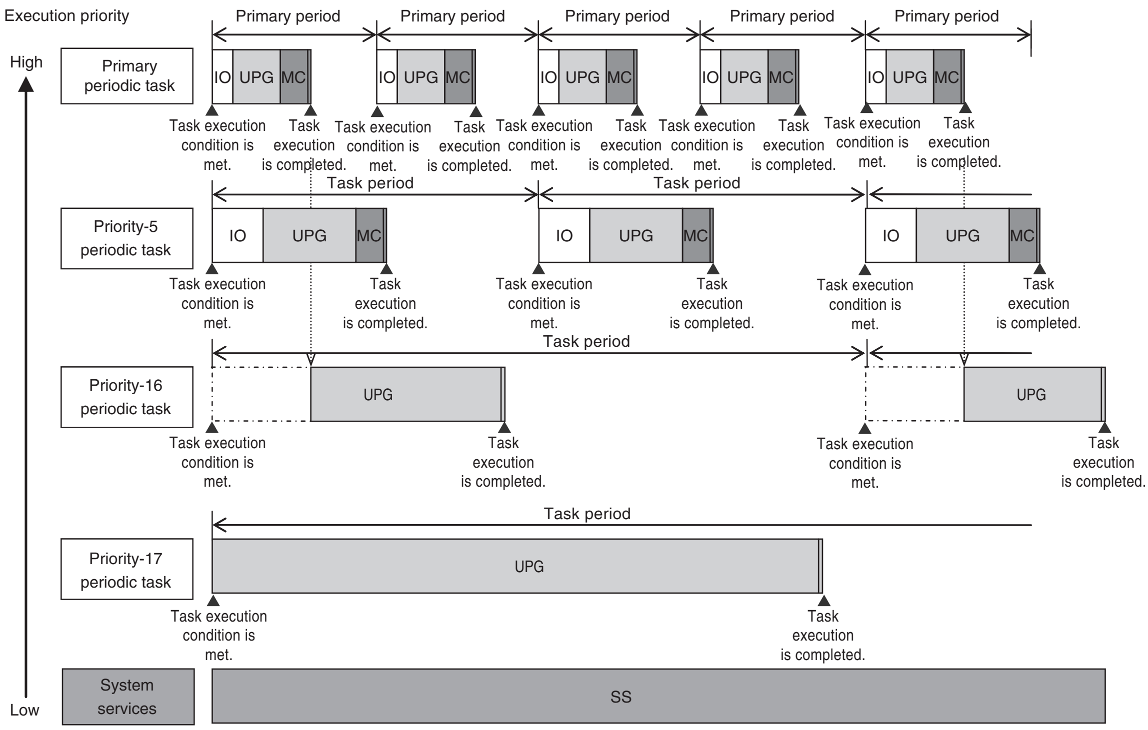
The primary periodic task is executed every primary period. The system service shown in this figure refers to non-task related processing, such as communications processing, that is performed by the CPU Unit. System services are executed in the unused time between execution of the tasks. Refer to *Processing Performed in System Services* on page 5-59 for details on the system services.



IO: Indicates I/O refreshing (data outputs and then data inputs).
UPG: User program execution.
MC: Indicates motion control.
SS: Indicates system services.

● Project with the Primary Periodic Task, Priority-5 Periodic Task, Priority-16 Periodic Task, and Priority-17 Periodic Task

- The primary periodic task has the highest execution priority, so it is always executed in the primary period.
- The priority-5 periodic task is executed in parallel with the primary periodic task.
- The priority-16 periodic task is executed after completion of the primary periodic task.
- Tasks are executed in the order of the execution priority, from the highest to the lowest.
- In this example, the task period for the priority-5 periodic task is set to twice the primary period. Also, the task period for the priority-16 periodic task is set to four times the primary period. This means that the timing at which execution of a task starts coincides with that of the primary periodic task once every two primary periods for the priority-5 periodic task and once every four primary periods for the priority-16 periodic task.
- The system services are executed at the required time without being affected by the tasks.



Precautions for Correct Use

If you have multiple tasks that read and write to the same variables, make sure to use exclusive control of variables between the tasks. Otherwise, a task other than the one currently in execution may change the variable values. Refer to *5-7-1 Ensuring Concurrency of Variable Values between Tasks* for details.

Tasks and Operating Modes

The relationship between CPU Unit operating modes and tasks is given in the following table.

Task	Specification
Primary periodic task	• These tasks are executed in both RUN mode and PROGRAM mode. • The user program is executed only in RUN mode.
Periodic tasks	
Event tasks	Event tasks are executed only in RUN mode.

Precautions for Correct Use

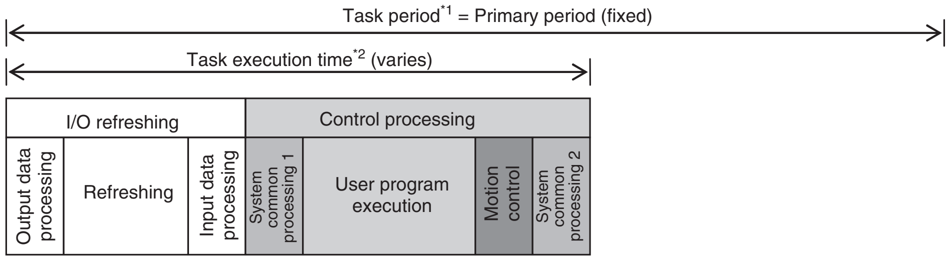
- Even if the execution condition for an event task is already met when you change the operating mode to RUN mode, the event task will not be executed. An event task is executed only when its execution condition changes from not met to met during RUN mode.
- Even in RUN mode, an event task is not executed if there is a major fault level error.

The Processing Performed in Each Task

● Primary Periodic Task

The primary periodic task has the highest execution priority. It executes processes with high speed and high precision.

In the specified period, this task performs system common processing, I/O refreshing, user program execution, and motion control.



*1: Task period — The CPU Unit executes tasks in this fixed period. This is a preset, fixed time.

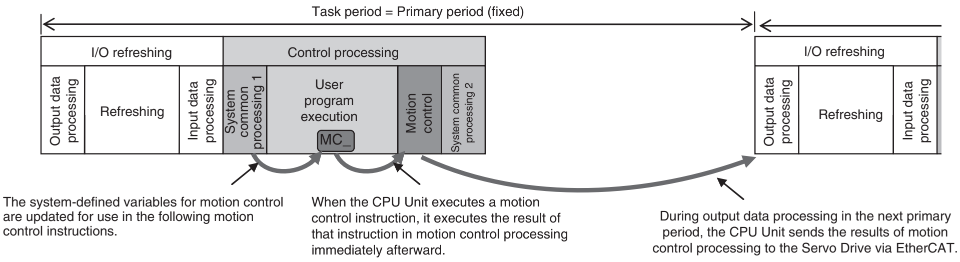
*2: Task execution time — This is the actual time it takes from the point that the execution condition is met until execution is completed.

Processing		Processing contents
I/O refreshing	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Whether the condition expression for event task execution is met or not is determined. • Input refresh data is loaded from Input Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1		• Processing for exclusive control of variables in tasks (when accessing tasks are set) • Motion input processing is performed.*1 • Data trace processing (sampling and trigger checking) is performed.
User program execution		• Programs assigned to tasks are executed in the order that they are assigned.
Motion control*2		• The motion control commands from the motion control instructions in the user program assigned to the primary periodic task and the priority-16 periodic task are executed. • Processing the motion outputs for I/O refreshing in the next primary periodic task.
System common processing 2		• Processing for exclusive control of variables in tasks (when refreshing tasks are set) • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the variable access time that is set in the Task Settings). • If there is processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed.

*1 The Axis Current Values (Position, Velocity, and Torque) and Drive Status in the system-defined variables for motion control are updated.

*2 When there are motion control instructions in user program execution in the primary periodic task, the CPU Unit executes the results from those instructions immediately afterward in motion control processing as shown below. The CPU Unit outputs the results to the Servo Drives during I/O refreshing in the next primary periodic task.

Note The processes in each cell in the above table are executed in the order of description.



When there is a motion control instruction in user program execution in the priority-16 periodic task, the CPU Unit executes the result from that instruction in the motion control processing (MC) of the next primary periodic task.

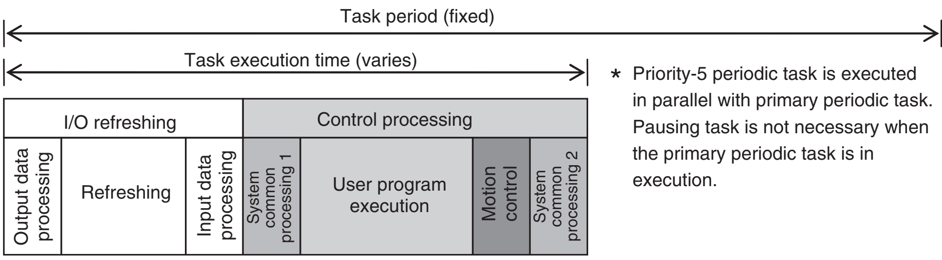
Refer to *5-10-3 System Input and Output Response Times* for details.

● Priority-5 Periodic Task

The priority-5 periodic task has the next highest execution priority after the primary periodic task. It executes processes with high speed and high precision.

In every period, this task performs system common processing, I/O refreshing, user program execution, and motion control.

The priority-5 periodic task is available only for NX-series CPU Units.



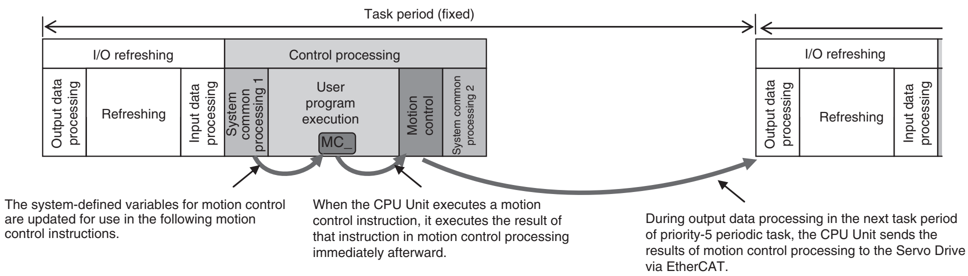
Processing		Processing contents
I/O refreshing	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Input refresh data is loaded from Input Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1		• Processing for exclusive control of variables in tasks is performed when accessing tasks are set. • Motion input processing is performed.*1 • Data tracing processing (sampling and trigger checking) is performed.
User program execution		• Programs assigned to tasks are executed in the order that they are assigned.
Motion control*2		• The motion control commands from the motion control instructions in the user program assigned to the priority-5 periodic task are executed. • Processing the motion outputs for I/O refreshing in the next priority-5 periodic task.

Processing	Processing contents
System common processing 2	• Processing for exclusive control of variables in tasks is performed when refreshing tasks are set. • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the variable access time that is set in the Task Settings). • If there is processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed.

*1 The Axis Current Values (Position, Velocity, and Torque) and Drive Status in the system-defined variables for motion control is updated.

*2 When there are motion control instructions in user program execution in the priority-5 periodic task, the CPU Unit executes the results from those instructions immediately afterward in motion control processing as shown below. The CPU Unit outputs the results to the Servo Drives during I/O refreshing in the next priority-5 periodic task.

Note The processes in each cell in the above table are executed in the order of description.



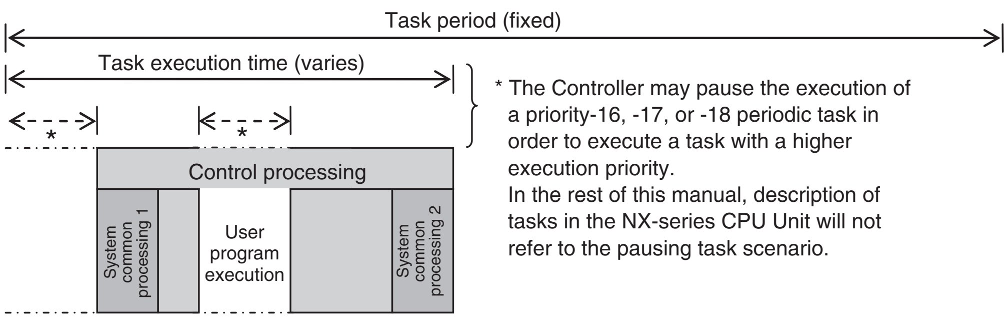
● Priority-16, Priority-17, or Priority-18 Periodic Task

A periodic task executes its programs every task period. The task period is specified as an integer multiple of the primary period. You can use 0 to 3 periodic tasks.

The priority-16 periodic task allows you to write control programs for the slaves and Units for which you set the priority-16 periodic task in the I/O Control Task Settings.

Processing for periodic tasks that do not control I/O is different from processing for periodic tasks that do control I/O.

Periodic Tasks That Do Not Control I/O

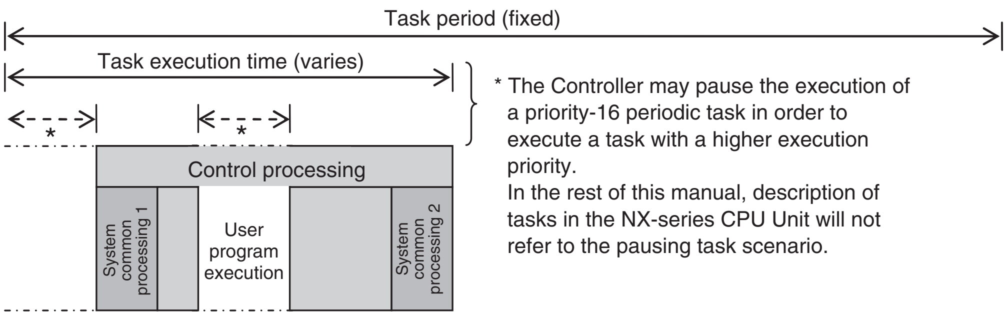


* The Controller may pause the execution of a priority-16, -17, or -18 periodic task in order to execute a task with a higher execution priority.
In the rest of this manual, description of tasks in the NX-series CPU Unit will not refer to the pausing task scenario.

Processing	Processing contents
System common processing 1	• Processing for exclusive control of variables in tasks (when accessing tasks are set) • Data trace processing (sampling and trigger checking) is performed.
User program execution	• Programs assigned to tasks are executed in the order that they are assigned.
System common processing 2	• Processing for exclusive control of variables in tasks (when refreshing tasks are set) • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the variable access time that is set in the Task Settings). • If there is processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed.

Note The processes in each cell in the above table are executed in the order of description.

Priority-16 Periodic Task That Controls I/O



Processing	Processing contents
System common processing 1	• Reflecting the input refresh data.*1 • Processing for exclusive control of variables in tasks is performed when accessing tasks are set. • Data tracing processing (sampling and trigger checking) is performed.
User program execution	• Programs assigned to tasks are executed in the order that they are assigned.
System common processing 2	• Processing for exclusive control of variables in tasks is performed when refreshing tasks are set. • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the variable access time that is set in the Task Settings). • If there is processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed. • Reflecting the execution results in the output refresh data.*2

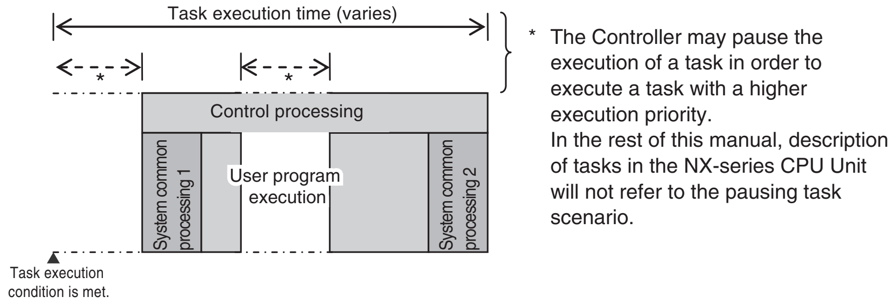
*1 This loads the input refresh data from the EtherCAT slaves for which you set the priority-16 periodic task in the I/O Control Task Settings. Input refresh data refers to the data that is input in process data communications during I/O refreshing in the primary periodic task.

*2 This reflects the execution results of the user program for the EtherCAT slaves for which you set the priority-16 periodic task in the I/O Control Task Settings in the output refresh data. The output refresh data will be output in process data communications during I/O refreshing in the primary periodic task.

Note The processes in each cell in the above table are executed in the order of description.

● Event Tasks

An event task is executed only once when the specified execution condition is met. You can use 0 to 32 event tasks. The processing details for event tasks are shown in the following figure.



Processing	Processing contents
System common processing 1	• Processing for exclusive control of variables in tasks is performed when accessing tasks are set.*1
User program execution	• Programs assigned to tasks are executed in the order that they are assigned.
System common processing 2	• Processing for exclusive control of variables in tasks is performed when refreshing tasks are set.*1

*1 Refer to *5-7-1 Ensuring Concurrency of Variable Values between Tasks* for details on exclusive control.

5-3-4 Event Task Execution Conditions for NX-series Controllers

An event task is executed only once when the specified execution condition is met. There are the following two types of execution conditions for event tasks.

Execution condition	Event task execution timing	Reason for use
Execution with the ActEventTask instruction	When ActEventTask instruction is executed	• When you need to explicitly specify which event tasks to execute in the user program • When the execution condition for the event task may change before meeting the condition expression for the variable is determined
Execution when a condition expression for a variable is met	When the specified variable value matches the specified condition expression*	When you want to simplify the user program by executing event tasks without user programming

* Refer to *Execution Timing When the Execution Condition Is a Condition Expression for a Variable* on page 5-28 for the timing of when the value of the specified variable is checked to see if the specified condition expression is met.

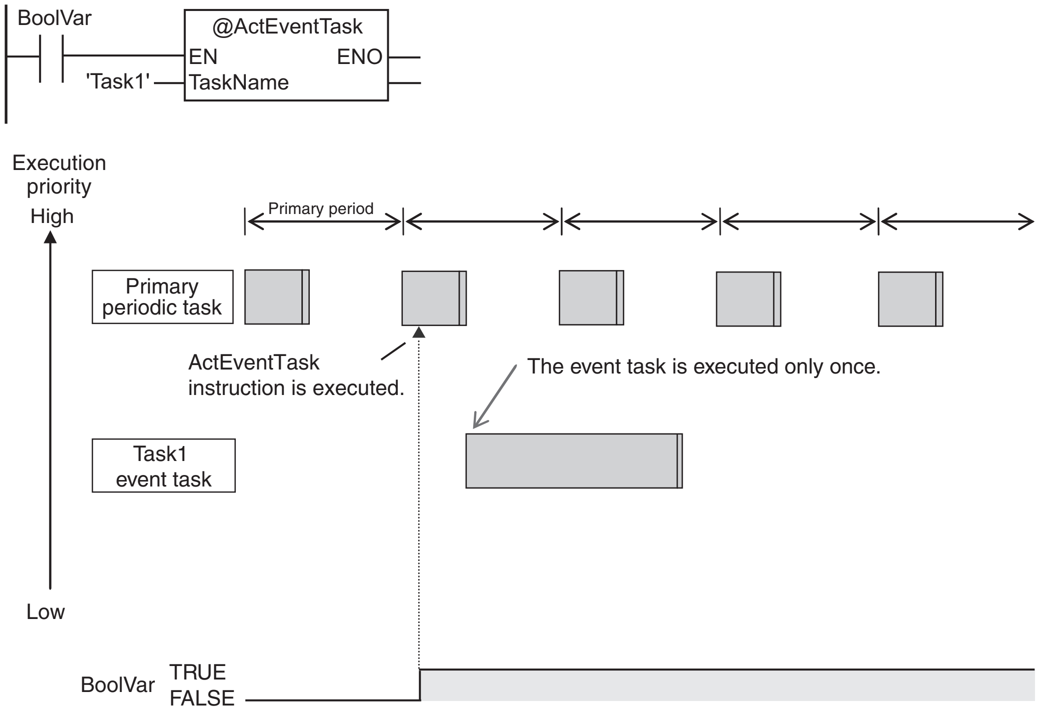
Executing Event Tasks for the ActEventTask Instruction

When the ActEventTask (Execute Event Task) instruction is executed in the user program, the specified event task is executed once. Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for the detailed specifications of the ActEventTask instruction.

Using the ActEventTask instruction to execute event tasks makes it easy to see which event tasks are executed. Also, this method is also effective when the execution condition for the event task may change before meeting the condition expression for the variable is determined.

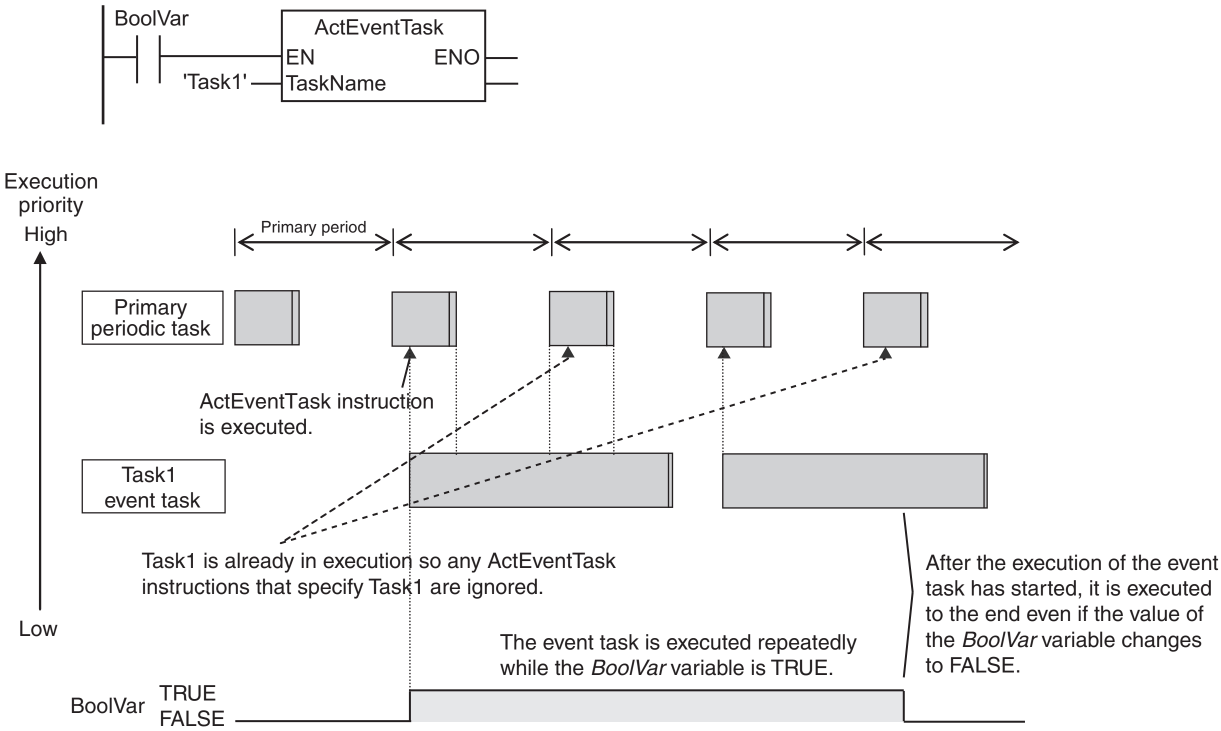
● Example of User Programming Using the ActEventTask Instruction

Example 1: Executing an Event Task Only Once When the Value of a Variable Changes
In the following example, the upward differentiation option is used for the ActEventTask instruction. This causes the Task1 event task to be executed only once when the *BoolVar* BOOL variable changes to TRUE.



Example 2: Executing an Event Task Repeatedly While the Value of a Variable Matches a Specified Value

In the following example, the upward differentiation option is not used for the ActEventTask instruction. This causes the Task1 event task to be executed as long as the *BoolVar* BOOL variable is TRUE. Any ActEventTask instructions that specify Task1 will be ignored if Task1 is already in execution. After the execution of the event task has started, it is executed to the end even if the value of *BoolVar* changes to FALSE during execution.



Executing Event Tasks When Condition Expressions for Variables Are Met

This method executes the event task once when the specified condition expression is met for the value of a variable that was specified on the Sysmac Studio. The event task is not executed repeatedly while the value of the variable matches the condition expression. It is executed only once when the value of the variable first changes so that it meets the condition expression. This method of execution does not require user programming to execute the event task.

● Variables for Which You Can Specify Condition Expressions

The following table lists the variables that you can specify for condition expressions.

Type of variables		Specification
System-defined variables		Possible.*1
Semi-user-defined variables		Possible.
User-defined variables	Global variables	Possible.
	Variables used in a program	Possible.
	Variables used in a function block	Possible.*2
	Variables used in a function	Not possible.

*1 The following variables cannot be used. EN, ENO, P_Off, P_CY, P_First_RunMode, P_First_Run and P_PRGER

*2 In-out variables cannot be used.

● Data Types of Variables for Condition Expressions

The following table lists the data types of variables that you can specify for condition expressions.

Classification of data type	Data type		Specification
Basic data types	**Boolean, bit string, integer, and real**		Possible.
	Duration, date, time of day, date and time, or text string data		Not possible.
Data type specifications	**Array specification**	**Arrays**	Not possible.
		Elements	Possible.*1
Derivative data type	**Structures**	**Structures**	Not possible.
		Members	Possible.*2
	Unions	**Unions**	Not possible.
		Members	Possible.*2
	Enumerations		Possible.

*1 The elements of the array must be Boolean variables, bit strings, integer data, or real data.

*2 The members must be Boolean, bit strings, integer data, or real data.

● Condition Expressions That You Can Specify

The condition expressions that you can specify depend on the data type of the variable that you specify for the condition expression. If the variable that you specify for a condition expression is bit string data, integer data, or real data, you must set a comparison constant to compare to the value of the variable.

Data type	Possible condition expressions
Boolean, Boolean array elements, Boolean structure members, and Boolean union members	Change to TRUE
	Change to FALSE
Bit string, real number, integer, as well as array element, structure member, or union member with one of those data types	Variable = {Comparison constant}
	Variable ≠ {Comparison constant}
	Variable > {Comparison constant}
	Variable ≥ {Comparison constant}
	Variable < {Comparison constant}
	Variable ≤ {Comparison constant}

● Valid Range of Comparison Constants

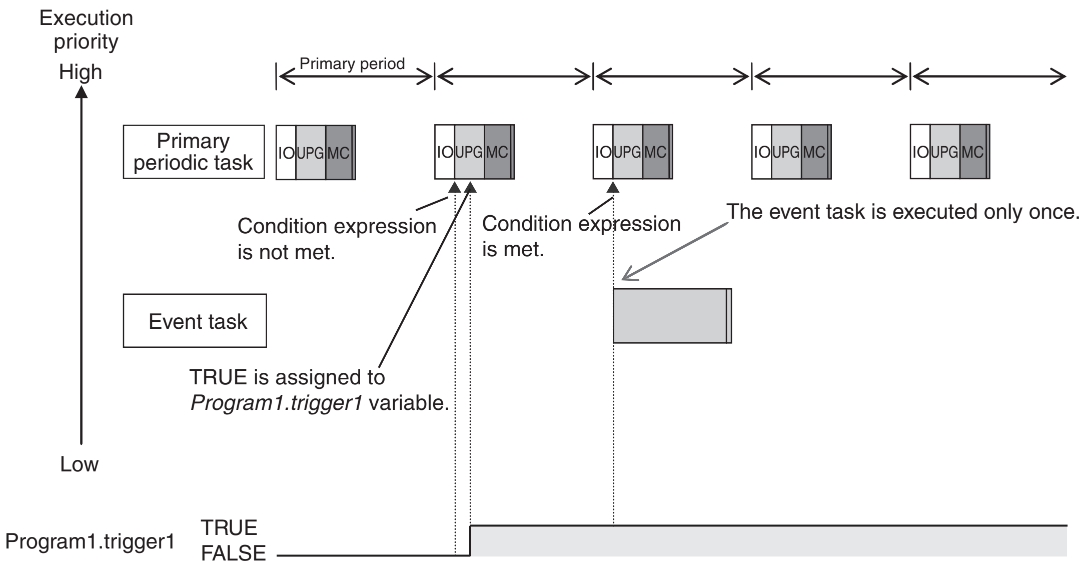
If the variable that you specify for a condition expression is bit string data, integer data, or real data, you must set a comparison constant to compare to the value of the variable. The valid range of comparison constants is the same as the valid range of the data type of the variable that you specify for the condition expression.

Refer to *Basic Data Types and Derivative Data Types* on page 6-30 for the valid range of values for each data type. For example, if the variable that you specify for the condition expression is a BYTE variable, the valid range of comparison constant values is from BYTE#16#00 to BYTE#16#FF.

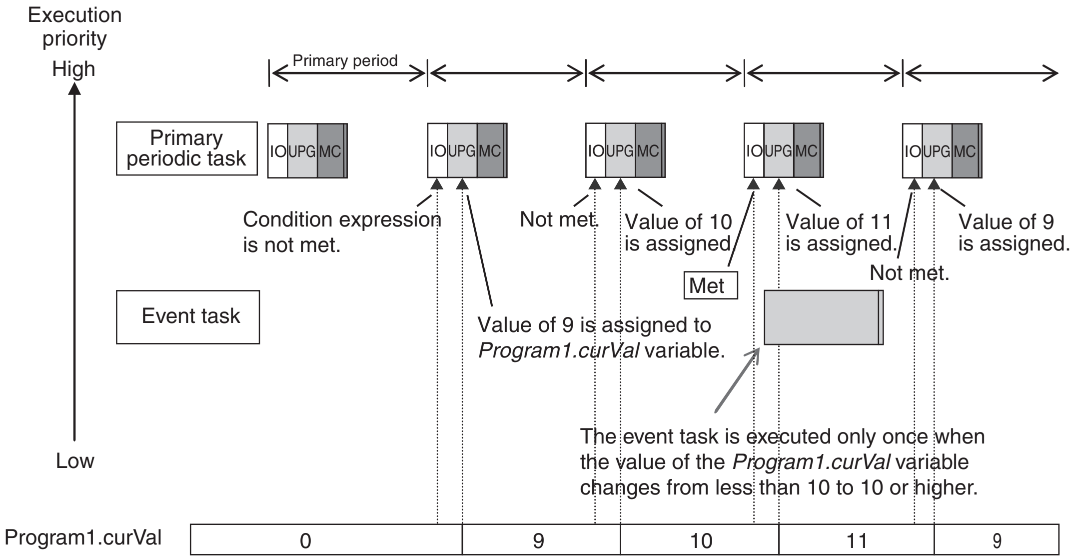
● Example of Executing Event Tasks When Condition Expressions for Variables Are Met

Example 1: Execution Condition for Event Task Set to a Change to TRUE of the *Program1.trigger1* Boolean Variable

When the value of *Program1.trigger1* changes to TRUE, the event task is executed only once.



Example 2: Execution Condition for an Event Task Set to When *Program1.curVal* (INIT variable) ≥ 10
The event task is executed only once when the value of *Program1.curVal* changes from less than 10 to 10 or higher.



Precautions for Correct Use

If the value of a specified variable changes in the primary periodic task, the CPU Unit evaluates whether the condition expression is met in the next primary periodic task. This means that the event task will be executed on completion of this evaluation against the condition expression in the next primary periodic task after the CPU Unit evaluates that the condition expression is met.

5-3-5 Event Task Execution Timing for NX-series Controllers

The execution priority of event tasks is 8 or 48. With a multi-core processor, NX-series CPU Units execute event tasks for which the execution condition is met according to the task execution priority. Depending on the user program, however, the CPU Unit may execute an event task in parallel with the primary periodic task, periodic tasks, or other event tasks with different execution priorities.

If you have multiple tasks that read and write to the same variables, make sure to use the following functions to control how an event task is executed with the primary periodic task, periodic tasks, or other event tasks with different execution priorities.

Purpose	Function to use
Accessing the same global variable from an event task and from another task	Exclusive control of variables in tasks • Settings for exclusive control of variables in tasks • Lock (Lock Tasks) instruction • Unlock (Unlock Tasks) instruction
Checking the execution status of other tasks	Task_IsActive (Determine Task Status) instruction

Refer to *5-7-1 Ensuring Concurrency of Variable Values between Tasks* for details.

The execution of an event task also depends on its execution conditions. You can also set the same execution priority for more than one event task. You must be careful when the execution conditions are met for more than one event task that has the same execution priority.

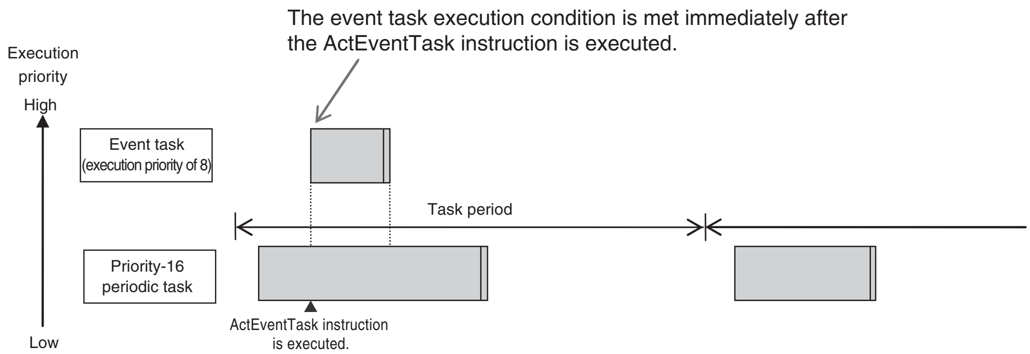
Differences in Execution Timing Based on the Execution Conditions of Event Tasks

The execution timing for event tasks depends on whether the execution condition is triggered by an ActEventTask instruction or by when a condition expression for a variable is met.

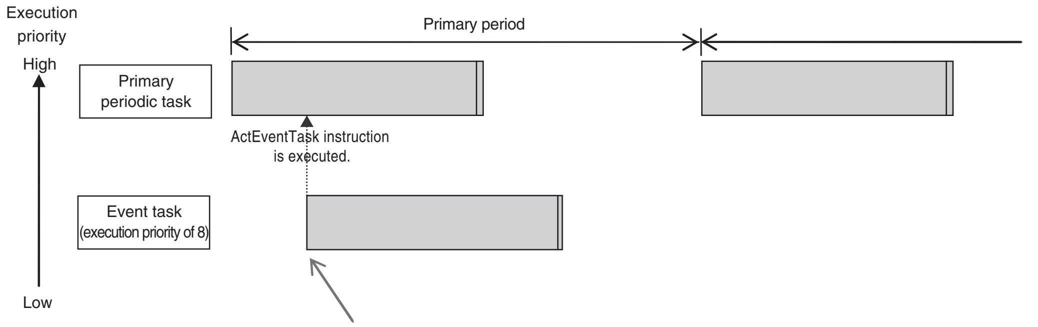
● Execution Timing When the Execution Condition Is an ActEventTask Instruction

If the execution condition for an event task is triggered by an ActEventTask instruction, the event task execution condition will be met immediately after the ActEventTask instruction is executed. The NX-series CPU Unit executes event tasks for which the execution condition is met according to the task execution priority.

Example 1: Executing an Event Task with an Execution Priority Higher Than the Task That Executes an Instruction



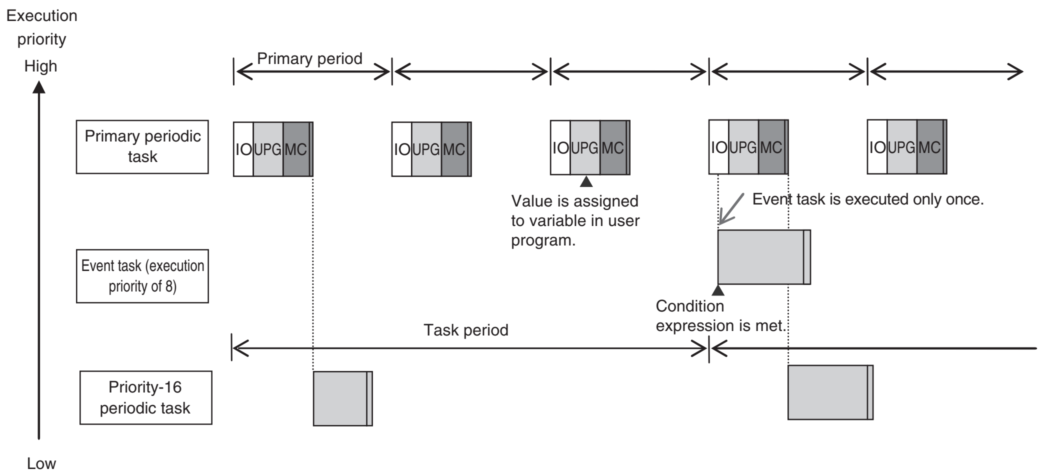
Example 2: Executing an Event Task with an Execution Priority Lower Than the Task That Executes an Instruction



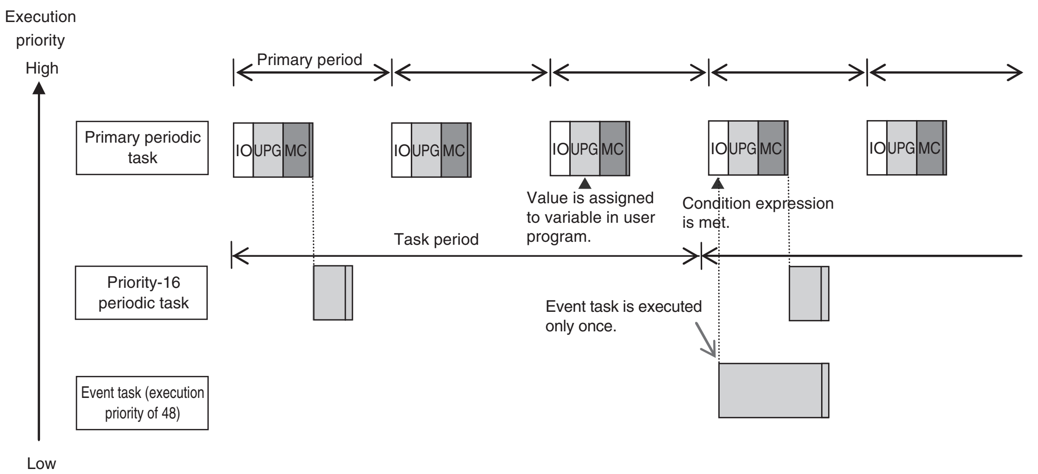
● Execution Timing When the Execution Condition Is a Condition Expression for a Variable

The condition expression is evaluated for a match inside the primary periodic task. The execution condition for an event task is met when it is evaluated to match the condition expression. The NX-series CPU Unit executes event tasks for which the execution condition is met according to the task execution priority.

Example 1: Project with a Priority-16 Periodic Task and an Event Task Execution with a Priority of 8



Example 2: Project with a Priority-16 Periodic Task and an Event Task Execution with a Priority of 48



Precautions for Correct Use

- For NX-series CPU Units, the timing at which the execution condition for an event task is met is the same regardless of whether the condition expression match is triggered by I/O refreshing in the primary periodic task, or by execution of a program that is assigned to the primary periodic task. This difference is described in the following table.

Trigger for condition expression to match	Timing at which the execution condition for an event task is met
I/O refreshing in the primary periodic task	Evaluation in the next primary periodic task
Execution of the programs in the primary periodic task	Evaluation in the next primary periodic task

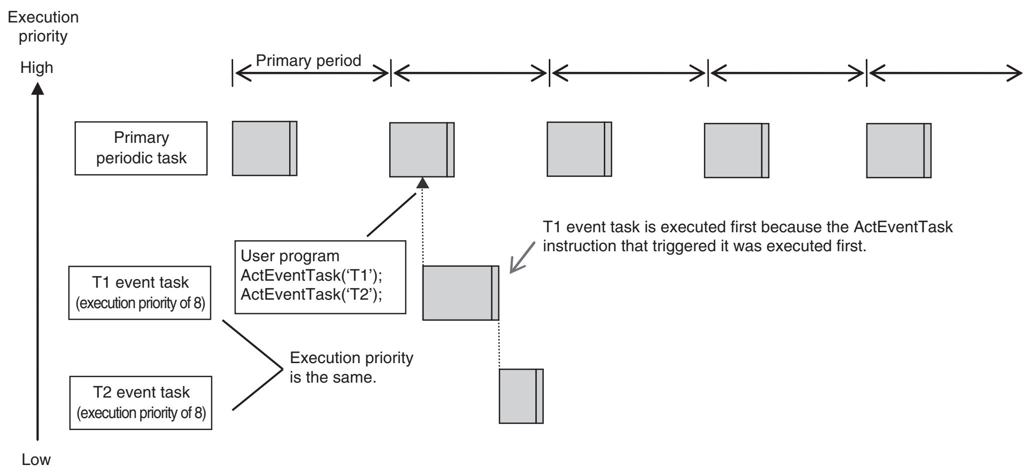
- In order for an event task to be executed, the condition expression must be met in the evaluation after the previous evaluation where the condition expression was not met. This means that even if the status of the condition expression changes from not met to met, if the condition returns to not met before the next evaluation, the event task will not be executed.

Execution Timing for Event Tasks with the Same Execution Priority

You can also set the same execution priority for more than one event task. If the execution conditions for more than one event task with the same execution priority are triggered by an ActEventTask instruction, the event tasks will be executed in the order that the instruction is executed.

Example 1: When Two ActEventTask Instructions Are Executed

In the example given below, two ActEventTask instructions are used to execute two event tasks. The T1 event task is executed before the T2 event task because the ActEventTask instruction that triggered T1 was executed first.



Example 2: When Both Condition Expressions for Variables and the ActEventTask Instruction Are Used

In this example, the execution conditions of the T1, T2, and T3 event tasks are set as given below.

- T1: Condition expression for the *VarA* variable
- T2: Condition expression for the *VarB* variable
- T3: ActEventTask instruction

The operation would proceed as described below if the condition expression for *VarA* was met during I/O refreshing, the ActEventTask instruction was executed in the user program, and the condition expression for *VarB* was met during execution of the user program all in the same primary period.

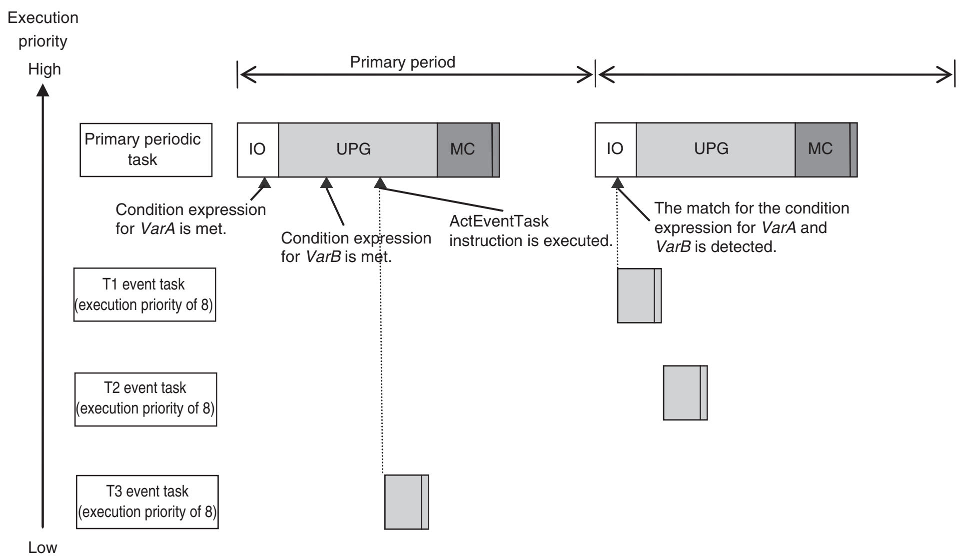
(1) The condition expression for *VarA* is met during I/O refreshing.

(2) The condition expression for *VarB* is met during execution of the user program.

(3) At this point, T1 and T2 are not executed because the condition expressions are not yet evaluated.

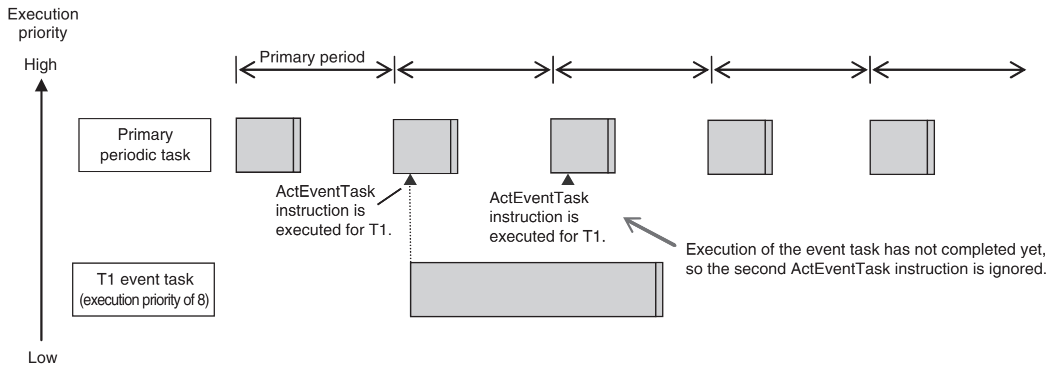
(4) The ActEventTask instruction is executed in the user program, so T3 is executed.

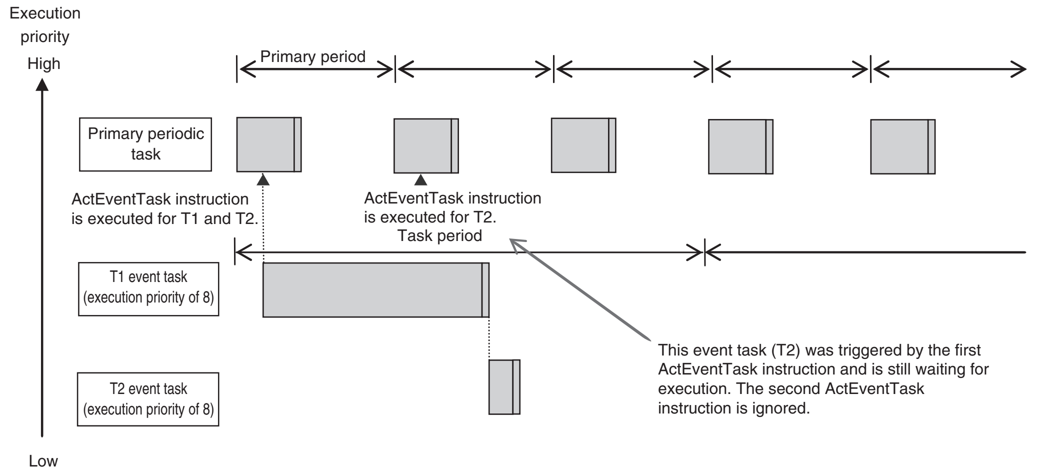
(5) When I/O refreshing is executed in the primary periodic task, the match is detected that the condition expressions for *VarA* and *VarB* are met, so T1 and T2 are executed. If the match is detected that more than one condition expression is met in the same execution period, the order of execution of event tasks is undefined. The following figure shows an example when T1 is executed first.



5-3-6 Operation When Execution Condition Is Met Again before Execution of the Event Task Is Completed

If the execution condition for an event task is met again before the execution of that event task is completed, the second match of the execution condition is ignored. “Before an event task is completed” includes the duration of execution of the event task and the time waiting for execution. After the execution of the event task has started, it is executed to the end even if the condition expression is no longer met.





5-4 Specifications and Basic Operation of Tasks for NJ-series Controllers

This section describes the specifications and basic operation of tasks for NJ-series CPU Units.

5-4-1 Specifications of Tasks for NJ-series Controllers

The specifications of tasks are given in the following table.

Item	Specification
Type of task	• Primary periodic task • Periodic task (priority 16, 17, or 18) • Event task (priority 8 or 48)
Numbers of tasks	• Primary periodic task: 1 • Periodic tasks: 0 to 3 tasks*1 • Event tasks: 0 to 32 tasks*2
Number of programs per task	128 max.
Task period of the primary periodic task	500 µs*3, 1 ms, 2 ms, or 4 ms
Task periods of periodic tasks	Set the task period of each periodic task to an integer multiple of the task period of the primary periodic task. Refer to the table of valid task periods for periodic tasks that is given below.

*1 There can be no more than one task with each of the following execution priorities: 16, 17, and 18.

*2 There can be up to 32 tasks with each of the following priorities as long as there are no more than a total of 32 tasks with these priorities: 8 and 48.

*3 With the NJ301-□□□□, you can use this setting with unit version 1.03 or later.
You cannot use this setting with the NJ101-□□□□.

● Valid Task Periods for Periodic Tasks

Task period of the primary periodic task	Task periods that you can set for periodic tasks
500 µs*1	1 ms, 2 ms, 3 ms, 4 ms, 5 ms, 8 ms, 10 ms, 15 ms, 20 ms, 25 ms, 30 ms, 40 ms, 50 ms, 60 ms, 75 ms, or 100 ms
1 ms	1 ms, 2 ms, 3 ms, 4 ms, 5 ms, 8 ms, 10 ms, 15 ms, 20 ms, 25 ms, 30 ms, 40 ms, 50 ms, 60 ms, 75 ms, or 100 ms
2 ms	2 ms, 4 ms, 8 ms, 10 ms, 20 ms, 30 ms, 40 ms, 50 ms, 60 ms, or 100 ms
4 ms	4 ms, 8 ms, 20 ms, 40 ms, 60 ms, or 100 ms

*1 With the NJ301-□□□□, you can use this setting with unit version 1.03 or later.
You cannot use this setting with the NJ101-□□□□.

5-4-2 Guidelines for Separating Tasks for NJ-series Controllers

All programs must be assigned to one of the tasks. Use the guidelines in the following table to determine which tasks to assign your programs to based on the requirements of the programs.

Task	Programs that are suitable for this task
Primary periodic task	• Programs that require I/O refreshing at an exact execution period. • Programs that require the highest execution priority. • Programs that contain motion control instructions with a high execution priority.
Priority-16 periodic task	• Programs that require I/O refreshing. • Programs with a relatively low execution priority that must be executed periodically. • Programs that contain motion control instructions with a relatively low execution priority.
Priority-17 or priority-18 periodic task	• Programs with a relatively low execution priority that must be executed periodically.
Event Task	• Programs that are executed only when specified conditions are met.

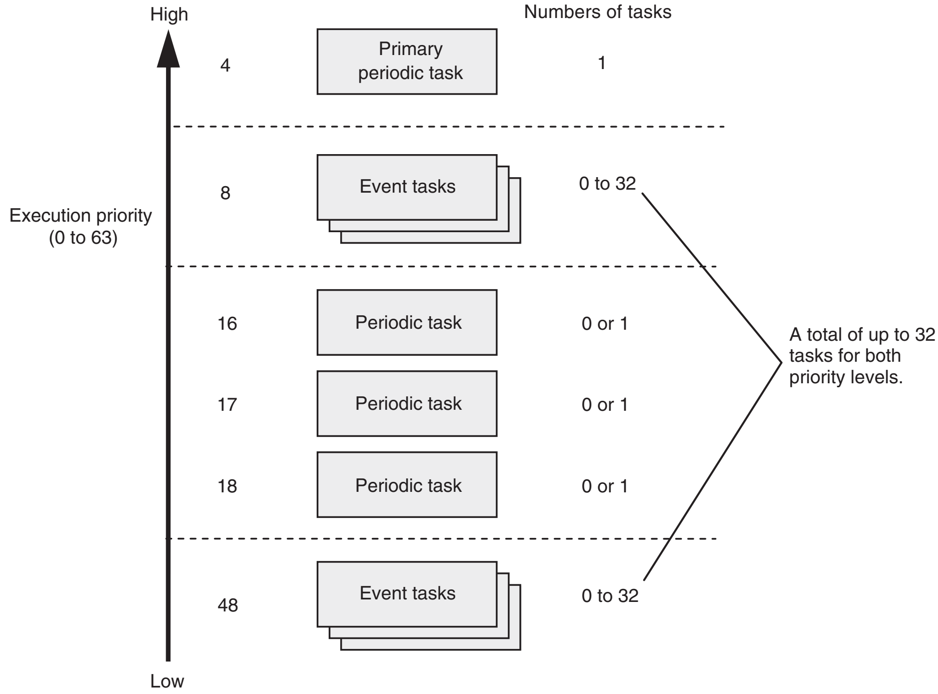
5-4-3 Basic Operation of Tasks for NJ-series Controllers

The CPU Unit cannot execute more than one task at the same time. The order in which tasks are executed depends on the execution priority that is set for each task.

Task Execution Priority

The type of the task determines its execution priority. If the execution condition is met for another task, Tb, that has a higher execution priority while task Ta execution is in progress, execution of Ta will be interrupted to allow execution of Tb. Processing for Ta will resume when processing for Tb is completed. The execution priority for each task type is given in the following table. The smaller the value of the execution priority, the higher the priority.

Task	Execution priority	Tasks with the same execution priority
Primary periodic task	4	---
Periodic task	16, 17, or 18	You cannot set the same execution priority for more than one task.
Event task	8 or 48	You can set the same execution priority for more than one event task. Refer to *5-4-5 Event Task Execution Timing for NJ-series Controllers* for the order of execution.



Task Periods for the Primary Periodic Task and Periodic Tasks

The CPU Unit repeatedly and cyclically executes the primary periodic task and periodic tasks. The task periods for periodic tasks must be assigned as integer multiples of the task period of the primary periodic task (called the primary period). Therefore, execution of both tasks will be start at the same time every few cycles.

For example, if the primary period is set to 1 ms and the task period of the priority-16 periodic task is set to 4 ms, the execution timing of the primary periodic task and the priority-16 periodic task is synchronized after each four executions of the primary periodic task.

Additional Information

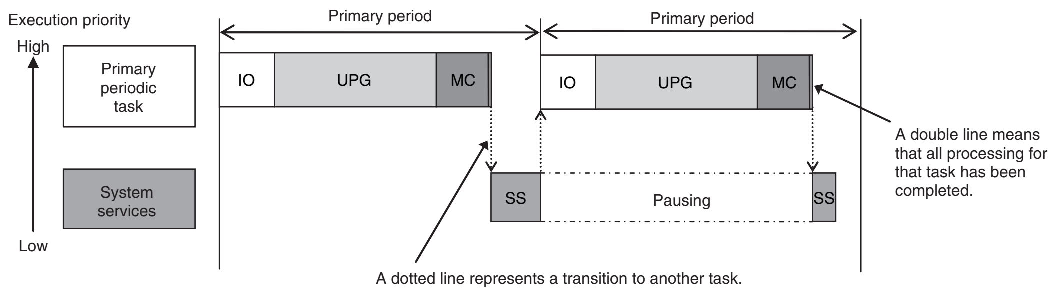
An event task is not executed periodically. Instead, it is executed only once when the specified execution condition is met. Therefore, execution of an event task depends on when its execution condition is met and on its execution priority.

Examples of Execution Order for Tasks

This section gives a few examples of the execution order for the primary periodic task and periodic tasks. Refer to *5-4-5 Event Task Execution Timing for NJ-series Controllers* for the order of execution of event tasks.

● Projects with Only the Primary Periodic Task

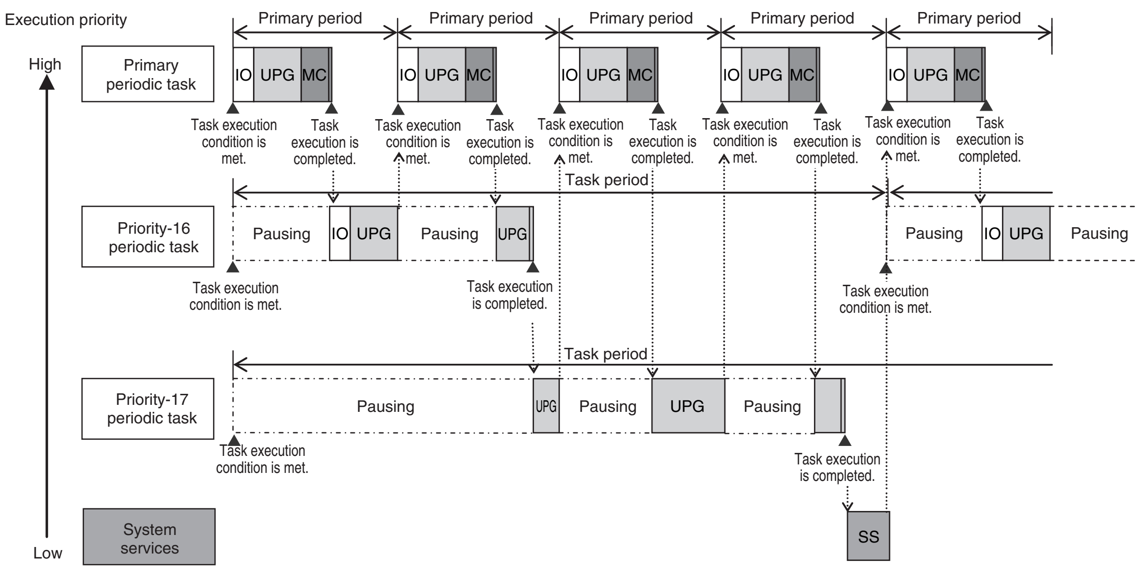
The primary periodic task is executed every primary period. The system service shown in this figure refers to non-task related processing, such as communications processing, that is performed by the CPU Unit. System services are executed in the unused time between execution of the tasks. Refer to *Processing Performed in System Services* on page 5-59 for details on the system services.



IO: Indicates I/O refreshing (data outputs and then data inputs).
UPG: User program execution.
MC: Indicates motion control.
SS: Indicates system services.

- **Project with the Primary Periodic Task, Priority-16 Periodic Task, and Priority-17 Periodic Task**

  - The primary periodic task has the highest execution priority, so it is always executed in the primary period.
  - The priority-16 periodic task has a lower execution priority than the primary periodic task, so it is executed when the primary periodic task is not being executed.
  - The priority-17 periodic task has an even lower execution priority, so it is executed when the above two tasks are not under execution.
  - In this example, the task period for the priority-16 periodic task is set to four times the primary period. This means that once every four primary periods, execution of the primary periodic task and the priority-16 periodic task will start at the same time.
  - System services are executed in the unused time between execution of the tasks.

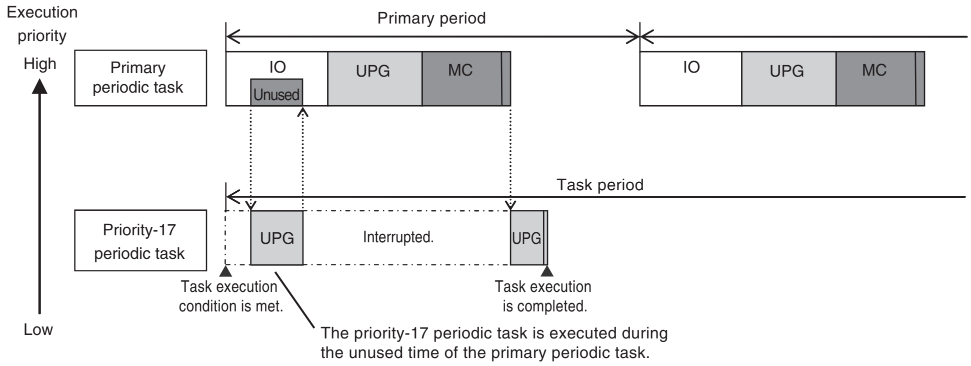


Precautions for Correct Use

If you have multiple tasks that read and write to the same variables, make sure to use exclusive control of variables between the tasks. Otherwise, a task other than the one currently in execution may change the variable values. Refer to *5-7-1 Ensuring Concurrency of Variable Values between Tasks* for details.

Executing Other Tasks during the Unused Time of a Task

A task with a higher execution priority is never interrupted to start execution of a task with a lower execution priority. However, for NJ-series CPU Units, if unused time occurs during a task with higher priority, that time may be used to start execution of a task with lower priority. An example of this is the time spent waiting for a data transfer to be completed in an I/O refresh process. As soon as processing resumes for the task with higher priority, the task with lower priority will be interrupted. This processing order is illustrated in the following figure.



Precautions for Correct Use

Even if unused time occurs in the primary periodic task, the priority-16 periodic task is always executed after the primary periodic task is completed. However, for NJ-series CPU Units, this restriction does not apply to the priority-17 or priority-18 periodic tasks. This restriction also does not apply to I/O refreshing in the priority-16 periodic task. I/O refreshing for the priority-16 periodic task may be executed during the unused time of the primary periodic task.

Tasks and Operating Modes

The relationship between CPU Unit operating modes and tasks is given in the following table.

Task	Specification
Primary periodic task	• These tasks are executed in both RUN mode and PROGRAM mode. • The user program is executed only in RUN mode.
Periodic tasks	
Event tasks	Event tasks are executed only in RUN mode.

Precautions for Correct Use

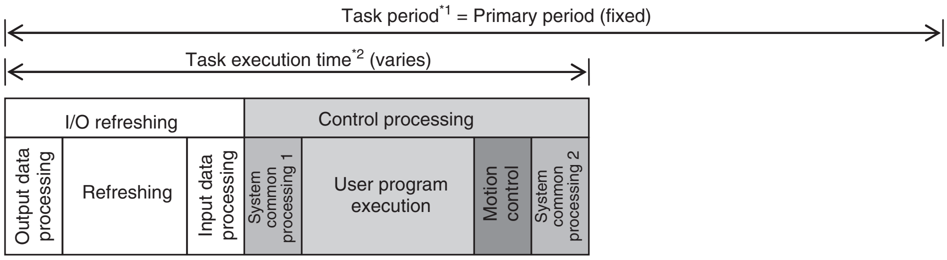
- Even if the execution condition for an event task is already met when you change the operating mode to RUN mode, the event task will not be executed. An event task is executed only when its execution condition changes from not met to met during RUN mode.
- Even in RUN mode, an event task is not executed if there is a major fault level error.

The Processing Performed in Each Task

● Primary Periodic Task

The primary periodic task has the highest execution priority. It executes processes with high speed and high precision.

In every period, this task performs system common processing, I/O refreshing, user program execution, and motion control. The primary periodic task performs motion control processing (MC).



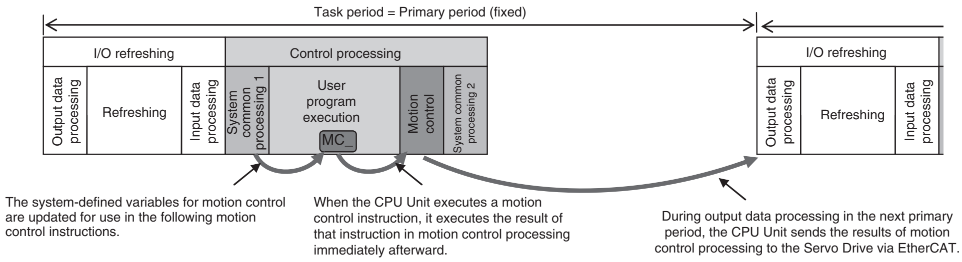
*1: Task period	The CPU Unit executes tasks in this fixed period. This is a preset, fixed time.
*2: Task execution time	This is the actual time it takes from the point that the execution condition is met until execution is completed.

Processing		Processing contents
I/O refreshing	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Input refresh data is loaded from Input Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1		• Processing for exclusive control of variables in tasks (when accessing tasks are set) • Motion input processing is performed.*1 • Data trace processing (sampling and trigger checking) is performed. • Whether the condition expression for event task execution is met or not is determined.
User program execution		• Programs assigned to tasks are executed in the order that they are assigned.
Motion control*2		• The motion control commands from the motion control instructions in the user program execution are executed. • Processing the motion outputs for I/O refreshing in the next primary periodic task.
System common processing 2		• Processing for exclusive control of variables in tasks (when refreshing tasks are set) • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the variable access time that is set in the Task Settings). • If there is processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed.

*1 The Axis Current Values (Position, Velocity, and Torque) and Drive Status in the system-defined variables for motion control are updated.

*2 When there are motion control instructions in user program execution in the primary periodic task, the CPU Unit executes the results from those instructions immediately afterward in motion control processing. The CPU Unit outputs the results to the Servo Drives during I/O refreshing in the next primary periodic task.

Note The processes in each cell in the above table are executed in the order of description.



When there is a motion control instruction in user program execution in the periodic task, the CPU Unit executes the result from that instruction in the motion control processing (MC) of the next primary periodic task.

Refer to *5-10-3 System Input and Output Response Times* for details.

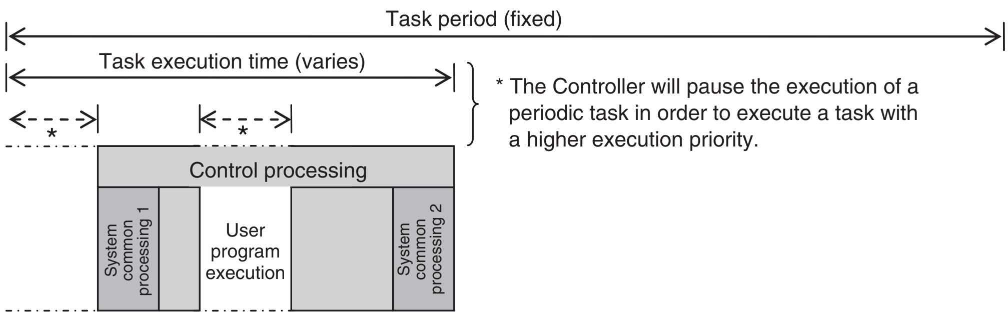
● Periodic Tasks

A periodic task executes its programs every task period. The task period is specified as an integer multiple of the primary period. You can use 0 to 3 periodic tasks.

The priority-16 periodic task can also refresh I/O.

Processing for periodic tasks that do not control I/O is different from processing for periodic tasks that do control I/O.

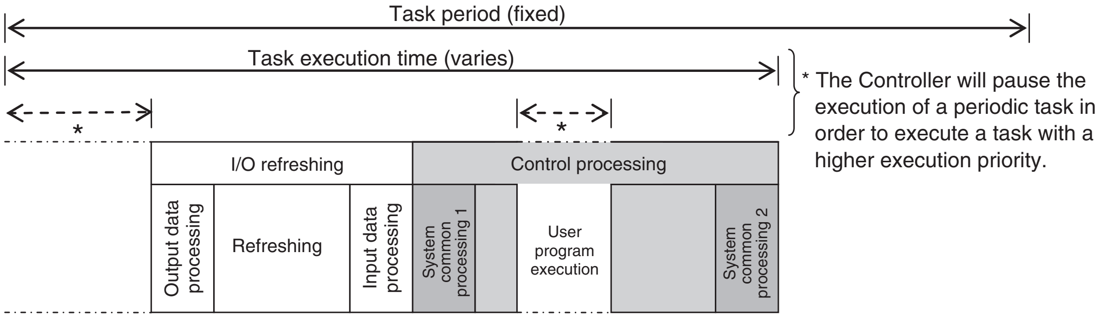
Periodic Tasks That Do Not Control I/O



Processing	Processing contents
System common processing 1	• Processing for exclusive control of variables in tasks (when accessing tasks are set) • Data trace processing (sampling and trigger checking) is performed.
User program execution	• Programs assigned to tasks are executed in the order that they are assigned.
System common processing 2	• Processing for exclusive control of variables in tasks (when refreshing tasks are set) • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the variable access time that is set in the Task Settings). • If there is processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed.

Note The processes in each cell in the above table are executed in the order of description.

Periodic Tasks That Control I/O



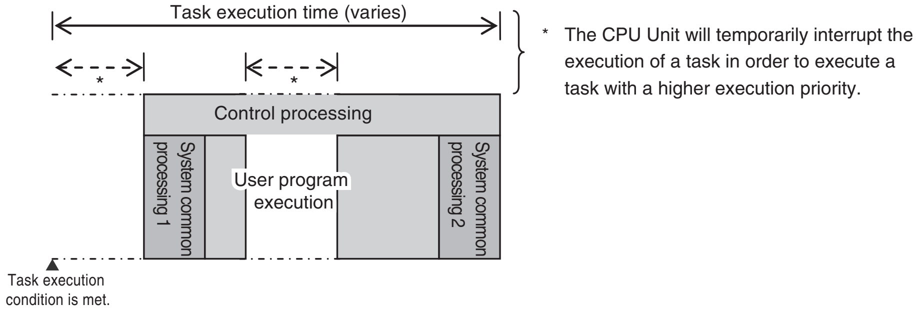
Processing		Processing contents
I/O refreshing*1	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Input refresh data is loaded from Input Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1		• Processing for exclusive control of variables in tasks (when accessing tasks are set) • Data trace processing (sampling and trigger checking) is performed.
User program execution		• Programs assigned to tasks are executed in the order that they are assigned.
System common processing 2		• Processing for exclusive control of variables in tasks (when refreshing tasks are set) • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the variable access time that is set in the Task Settings). • If there is processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed.

*1 I/O refreshing of CJ-series Units is executed.

Note The processes in each cell in the above table are executed in the order of description.

● Event Tasks

An event task is executed only once when the specified execution condition is met. You can use 0 to 32 event tasks. The processing details for event tasks are shown in the following figure.



Processing	Processing contents
System common processing 1	• Processing for exclusive control of variables in tasks (when accessing tasks are set)*1
User program execution	• Programs assigned to tasks are executed in the order that they are assigned.
System common processing 2	• Processing for exclusive control of variables in tasks (when refreshing tasks are set)*1

*1 Refer to *5-7-1 Ensuring Concurrency of Variable Values between Tasks* for details on exclusive control.

5-4-4 Event Task Execution Conditions for NJ-series Controllers

An event task is executed only once when the specified execution condition is met. There are the following two types of execution conditions for event tasks.

Execution condition	Event task execution timing	Reason for use
Execution with the ActEvent-Task instruction	When ActEventTask instruction is executed	• When you need to explicitly specify which event tasks to execute in the user program • When the execution condition for the event task may change before meeting the condition expression for the variable is determined
Execution when a condition expression for a variable is met	When the specified variable value matches the specified condition expression*	When you want to simplify the user program by executing event tasks without user programming

* Refer to *Execution Timing When the Execution Condition Is a Condition Expression for a Variable* on page 5-28 for the timing of when the value of the specified variable is checked to see if the specified condition expression is met.

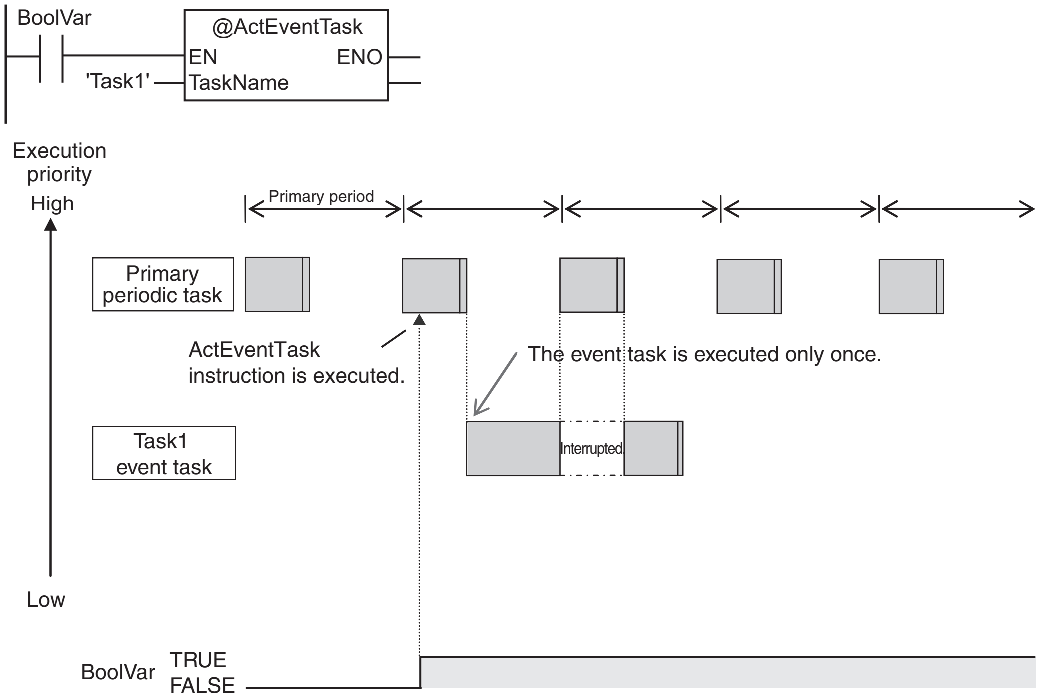
Executing Event Tasks for the ActEventTask Instruction

When the ActEventTask (Execute Event Task) instruction is executed in the user program, the specified event task is executed once. Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for the detailed specifications of the ActEventTask instruction.

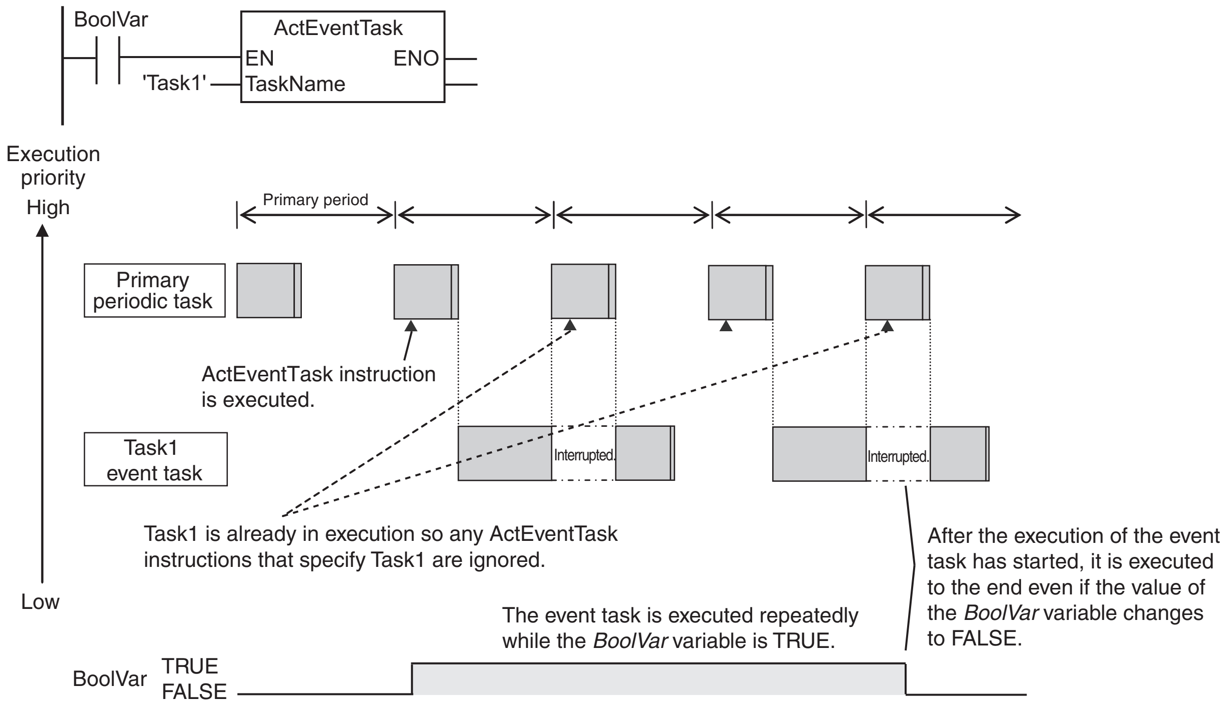
Using the ActEventTask instruction to execute event tasks makes it easy to see which event tasks are executed. Also, this method is also effective when the execution condition for the event task may change before meeting the condition expression for the variable is determined.

● Example of User Programming Using the ActEventTask Instruction

Example 1: Executing an Event Task Only Once When the Value of a Variable Changes
In the following example, the upward differentiation option is used for the ActEventTask instruction. This causes the Task1 event task to be executed only once when the *BoolVar* BOOL variable changes to TRUE.



Example 2: Executing an Event Task Repeatedly While the Value of a Variable Matches a Specified Value
In the following example, the upward differentiation option is not used for the ActEventTask instruction. This causes the Task1 event task to be executed as long as the *BoolVar* BOOL variable is TRUE. Any ActEventTask instructions that specify Task1 will be ignored if Task1 is already in execution. After the execution of the event task has started, it is executed to the end even if the value of *BoolVar* changes to FALSE during execution.



Executing Event Tasks When Condition Expressions for Variables Are Met

This method executes the event task once when the specified condition expression is met for the value of a variable that was specified on the Sysmac Studio. The event task is not executed repeatedly while the value of the variable matches the condition expression. It is executed only once when the value of the variable first changes so that it meets the condition expression. This method of execution does not require user programming to execute the event task.

● Variables for Which You Can Specify Condition Expressions

The following table lists the variables that you can specify for condition expressions.

Type of variables		Specification
System-defined variables		Possible.*1
Semi-user-defined variables		Possible.
User-defined variables	Global variables	Possible.
	Variables used in a program	Possible.
	Variables used in a function block	Possible.*2
	Variables used in a function	Not possible.

*1 The following variables cannot be used. EN, ENO, P_Off, P_CY, P_First_RunMode, P_First_Run and P_PRGER

*2 In-out variables cannot be used.

● Data Types of Variables for Condition Expressions

The following table lists the data types of variables that you can specify for condition expressions.

Classification of data type	Data type		Specification
Basic data types	**Boolean, bit string, integer, and real**		Possible.
	Duration, date, time of day, date and time, or text string data		Not possible.
Data type specifications	**Array specification**	**Arrays**	Not possible.
		Elements	Possible.*1
Derivative data type	**Structures**	**Structures**	Not possible.
		Members	Possible.*2
	Unions	**Unions**	Not possible.
		Members	Possible.*2
	Enumerations		Possible.

*1 The elements of the array must be Boolean variables, bit strings, integer data, or real data.

*2 The members must be Boolean, bit strings, integer data, or real data.

Condition Expressions That You Can Specify

The condition expressions that you can specify depend on the data type of the variable that you specify for the condition expression. If the variable that you specify for a condition expression is bit string data, integer data, or real data, you must set a comparison constant to compare to the value of the variable.

Data type	Possible condition expressions
Boolean, Boolean array elements, Boolean structure members, and Boolean union members	Change to TRUE
	Change to FALSE
Bit string, real number, integer, as well as array element, structure member, or union member with one of those data types	Variable = {Comparison constant}
	Variable ≠ {Comparison constant}
	Variable > {Comparison constant}
	Variable ≥ {Comparison constant}
	Variable < {Comparison constant}
	Variable ≤ {Comparison constant}

Valid Range of Comparison Constants

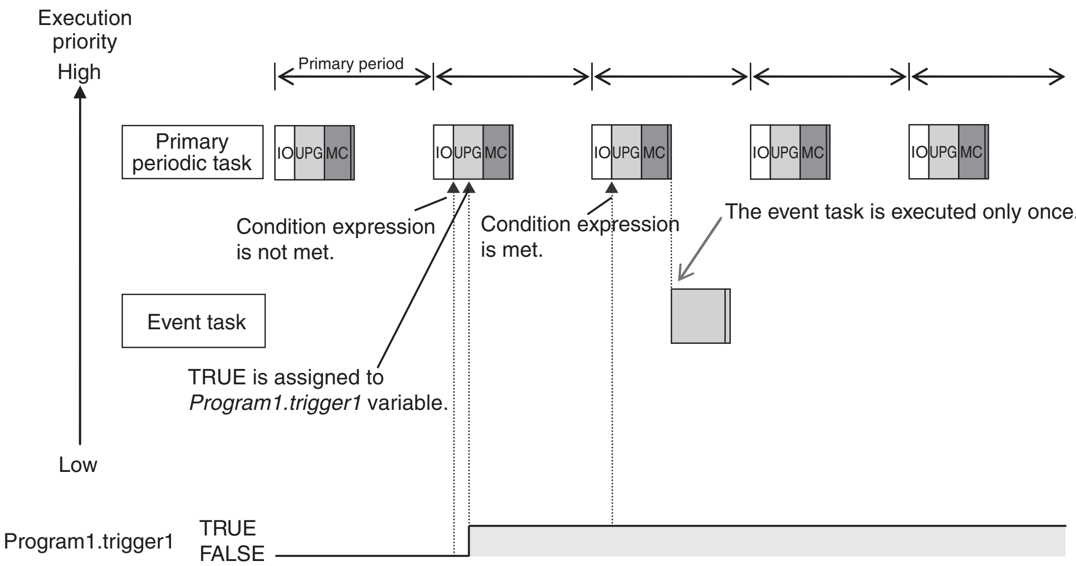
If the variable that you specify for a condition expression is bit string data, integer data, or real data, you must set a comparison constant to compare to the value of the variable. The valid range of comparison constants is the same as the valid range of the data type of the variable that you specify for the condition expression.

Refer to *Basic Data Types and Derivative Data Types* on page 6-30 for the valid range of values for each data type. For example, if the variable that you specify for the condition expression is a BYTE variable, the valid range of comparison constant values is from BYTE#16#00 to BYTE#16#FF.

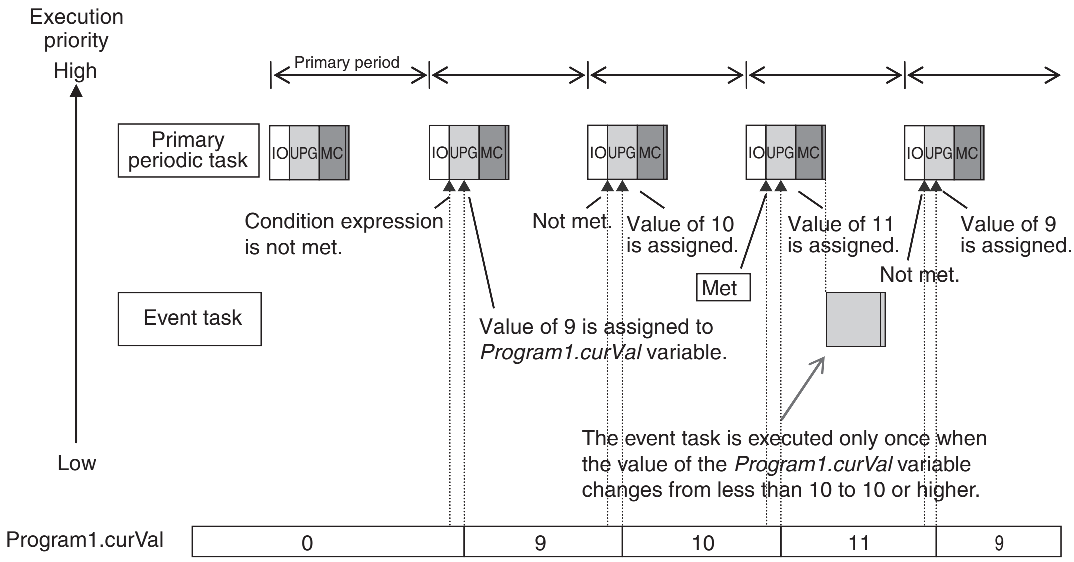
Example of Executing Event Tasks When Condition Expressions for Variables Are Met

Example 1: Execution Condition for Event Task Set to a Change to TRUE of the *Program1.trigger1* Boolean Variable

When the value of *Program1.trigger1* changes to TRUE, the event task is executed only once.



Example 2: Execution Condition for an Event Task Set to When *Program1.curVal* (INIT variable) ≥ 10
The event task is executed only once when the value of *Program1.curVal* changes from less than 10 to 10 or higher.



Precautions for Correct Use

The CPU Unit evaluates whether the condition expression is met before the programs that are assigned to the primary periodic task are executed. Even if the specified value of the variable matches the condition expression during execution of the program in the primary periodic task, the condition is not evaluated until just before the next execution of the primary periodic task. This means that the event task will be executed after the end of the execution of the primary periodic task that follows the execution of the primary periodic task in which the value of the variable meets the condition expression.

5-4-5 Event Task Execution Timing for NJ-series Controllers

The execution priority of event tasks is 8 or 48. If the execution conditions for an event task are met while another task is in execution, the task with the higher execution priority is given priority. The task with the lower execution priority is interrupted. This is the same as with the primary periodic task and periodic tasks. The execution of an event task also depends on its execution conditions.

You can also set the same execution priority for more than one event task. You must be careful when the execution conditions are met for more than one event task that has the same execution priority.

Differences in Execution Timing Based on the Execution Conditions of Event Tasks

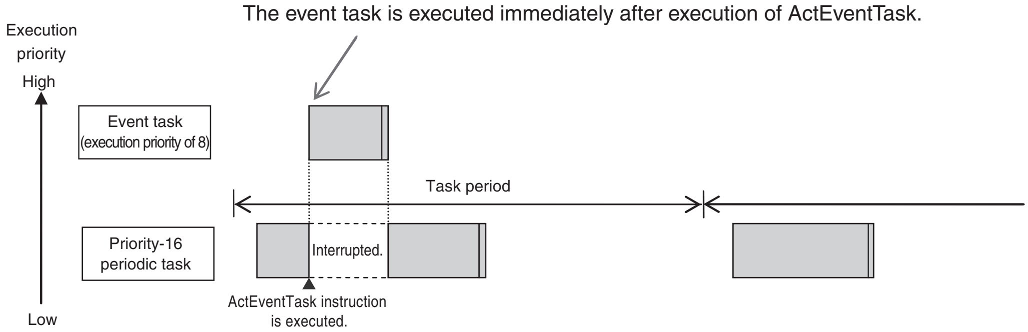
The execution of event tasks depends on whether the execution condition is triggered by an ActEventTask instruction or by when an condition expression for a variable is met.

● Execution Timing When the Execution Condition Is an ActEventTask Instruction

If the execution condition for an event task is triggered by an ActEventTask instruction, the event task will be executed immediately after the ActEventTask instruction is executed.

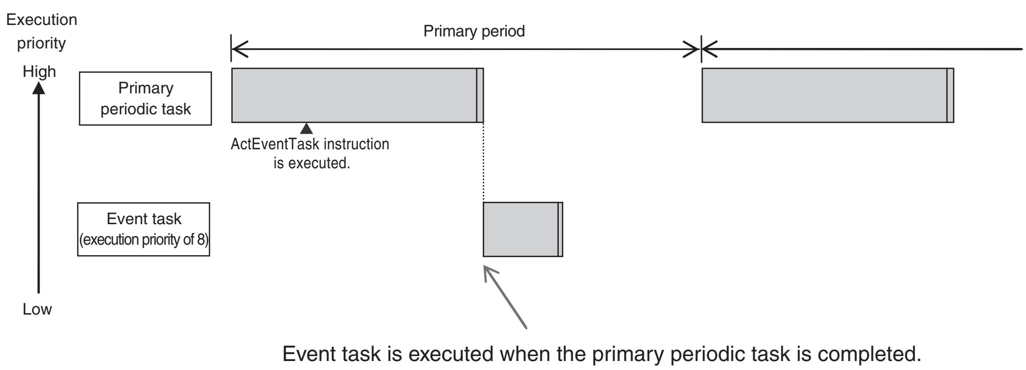
Example 1: ActEventTask Instruction Executed in Priority-16 Periodic Task and an Event Task with an Execution Priority of 8

The execution priority of the event task (execution priority of 8) is higher than the execution priority of the priority-16 periodic task. Execution of the priority-16 periodic task is therefore interrupted and the event task is executed.



Example 2: Executing an ActEventTask Instruction within the Primary Periodic Task

The event task has a lower execution priority than the primary periodic task, so the event task is executed after execution of the primary periodic task is completed.

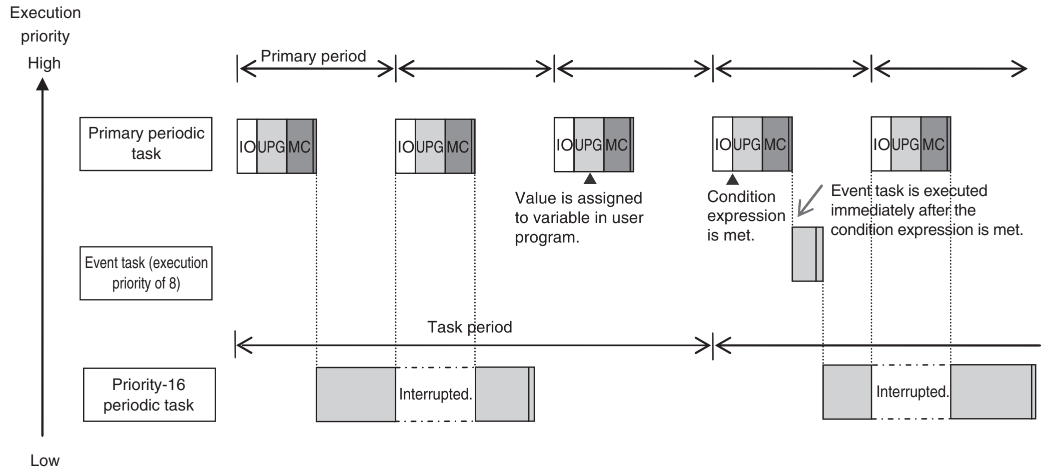


● Execution Timing When the Execution Condition Is a Condition Expression for a Variable

The condition expression is evaluated for a match inside the primary periodic task. This means that the event task will be executed immediately after the first execution of the primary periodic task after the specified value of the variable meets the condition expression. However, if there are tasks with a higher execution priority than the event task, those tasks will be executed first.

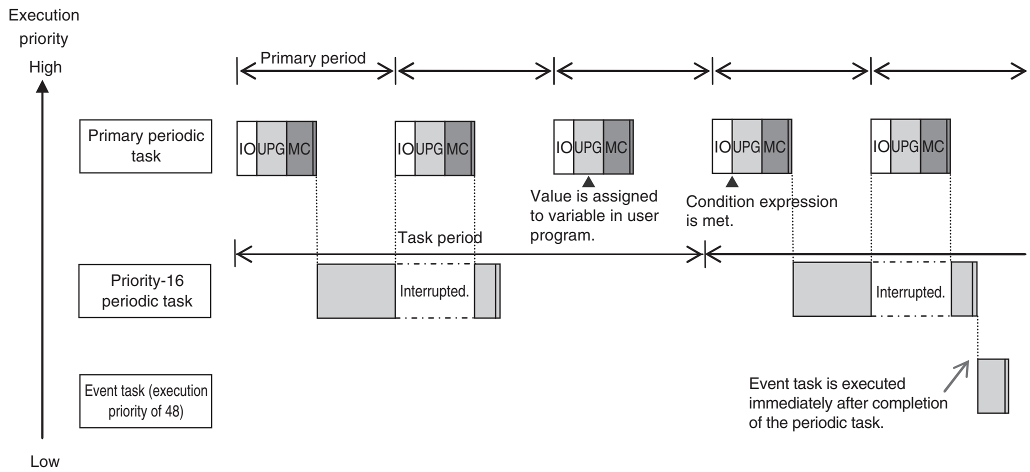
Example 1: Project with a Priority-16 Periodic Task and an Event Task Execution with a Priority of 8

The event task is executed immediately after the first execution of the primary periodic task after the condition expression is met. The execution priority of the event task (execution priority of 8) is higher than the execution priority of the priority-16 periodic task. The priority-16 periodic task is therefore executed after the event task is executed.



Example 2: Project with a Priority-16 Periodic Task and an Event Task Execution with a Priority of 48

The execution priority of the event task is lower than the execution priority of the priority-16 periodic task. The event task is therefore executed after the priority-16 periodic task is executed.



Precautions for Correct Use

- The execution timing of an event task depends on how the condition expression is met. The match can be triggered by I/O refreshing in the primary periodic task, or by execution of a program that is assigned to the primary periodic task. This difference is described in the following table. This difference occurs because the condition expression is evaluated for a match by system common processing 1 inside the primary periodic task. Processing in the primary periodic task takes place in this order: I/O refreshing, system common processing 1, and user program execution.

Trigger for condition expression to match	Event task execution timing
I/O refreshing in the primary periodic task	After completion of the primary periodic task
Execution of the programs in the primary periodic task	After completion of the next execution of the primary periodic task

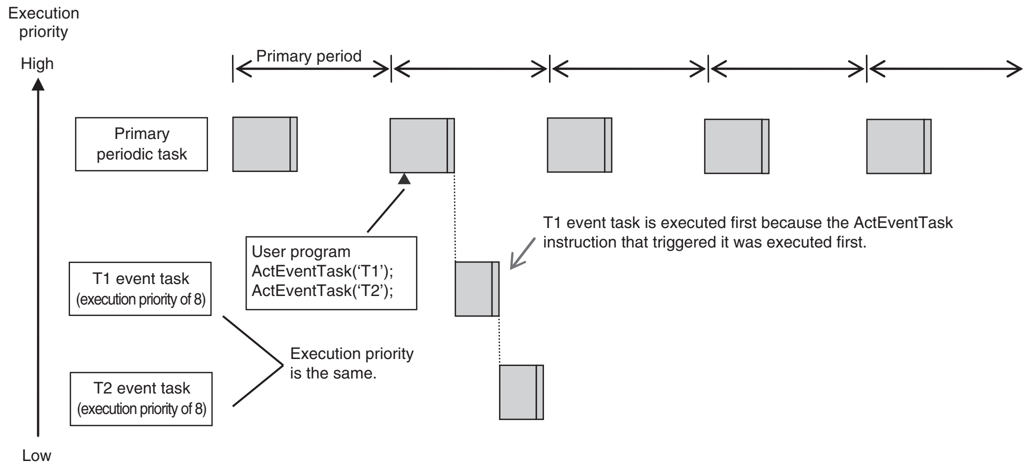
- In order for an event task to be executed, the condition expression must be met in the evaluation after the previous evaluation where the condition expression was not met. This means that even if the status of the condition expression changes from not met to met, if the condition returns to not met before the next evaluation, the event task will not be executed.

Execution Timing for Event Tasks with the Same Execution Priority

You can also set the same execution priority for more than one event task. If the execution conditions are met for more than one event task that has the same execution priority, the event tasks will be executed in the order that their execution conditions are met.

Example 1: When Two ActEventTask instructions Are Executed

In the example given below, two ActEventTask instructions are used to execute two event tasks. The T1 event task is executed before the T2 event task because the ActEventTask instruction that triggered T1 was executed first.



Example 2: When Both Condition Expressions for Variables and the ActEventTask Instruction Are Used

In this example, the execution conditions of the T1, T2, and T3 event tasks are set as given below.

- T1: Condition expression for the *VarA* variable
- T2: Condition expression for the *VarB* variable
- T3: ActEventTask instruction

The operation would proceed as described below if the condition expression for *VarA* was met during I/O refreshing, the ActEventTask instruction was executed in the user program, and the condition expression for *VarB* was met during execution of the user program all in the same primary period.

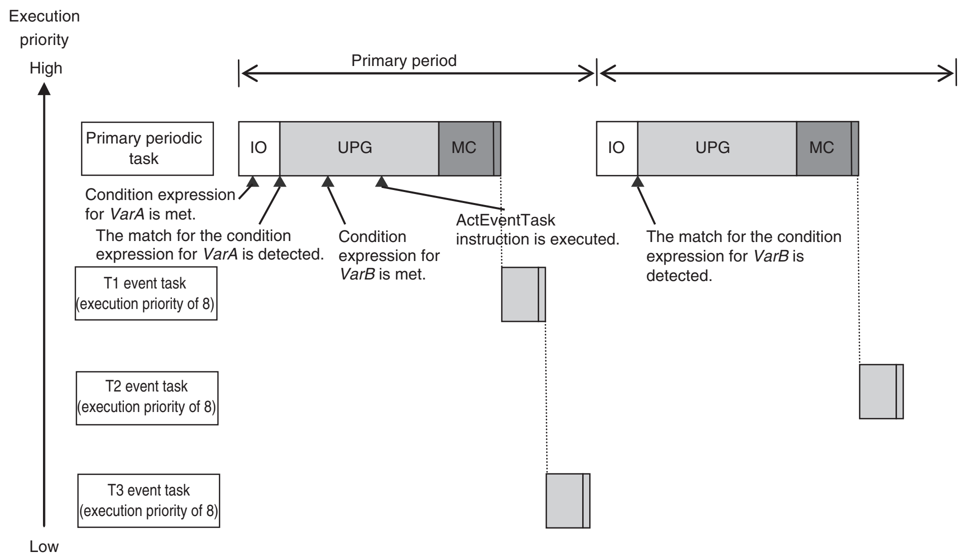
(1) The condition expression for *VarA* is met during I/O refreshing.

(2) In system common processing 1, the match is detected for the condition expression for *VarA* and T1 is executed. The condition expression for *VarB* is not yet met, so T2 is not executed.

(3) The condition expression for *VarB* is met during execution of the user program.

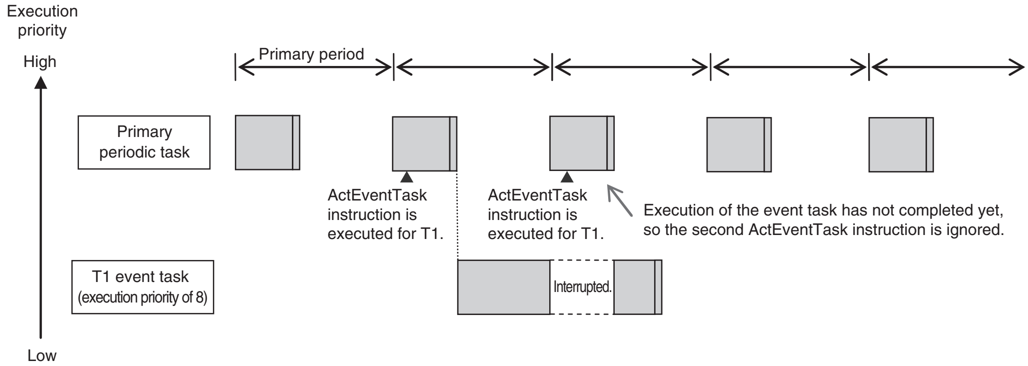
(4) The ActEventTask instruction is executed in the user program, so T3 is executed.

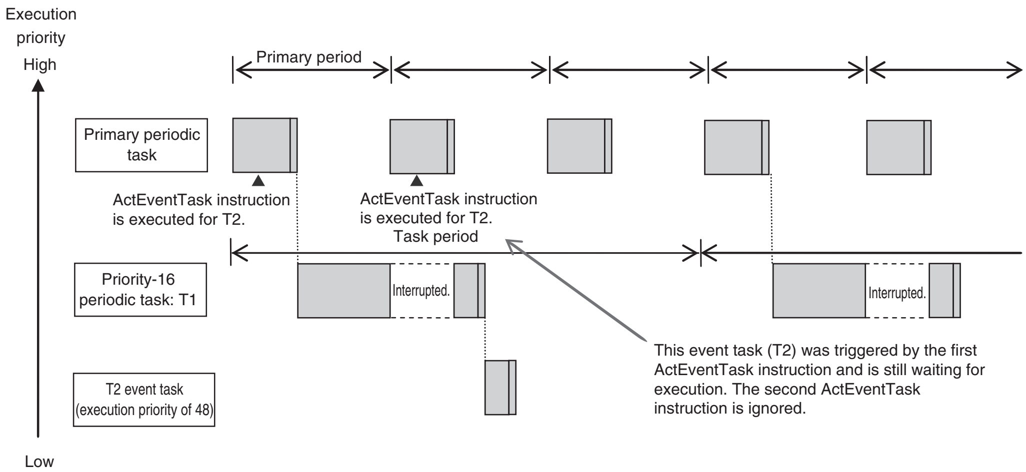
(5) In system common processing 1 in the next primary period, the match is detected for the condition expression for *VarB* and T2 is executed.



5-4-6 Operation When Execution Condition Is Met Again before Execution of the Event Task Is Completed

If the execution condition for an event task is met again before the execution of that event task is completed, the second match of the execution condition is ignored. “Before an event task is completed” includes the duration of execution of the event task and the time waiting for execution. After the execution of the event task has started, it is executed to the end even if the condition expression is no longer met.





5-5 Tag Data Link Service and System Services

The CPU Unit performs processing other than the primary periodic task, periodic tasks, and event tasks. This processing includes the tag data link service and the system services. The processing that is performed, the execution priority, and the execution timing for these services are given in the following table.

Item	Tag data link service	System services
Execution priority	The execution priority is between the execution priorities of the priority-16 periodic task and the priority-17 periodic task.	The execution priority is lower than that of any of the tasks.
Processing	Data is exchanged by using tags with other controllers and devices on an EtherNet/IP network. You can use the built-in EtherNet/IP port in the CPU Unit or a CJ-series CJ1W-EIP21 EtherNet/IP Unit to connect to an HMI. *1	System services include services with low execution priority, such as the USB port service and SD Memory Card service.
Execution timing	The execution interval and the time that is required for each execution depend on the model of the CPU Unit and on the tag data link settings.*2	For NX-series CPU Units, the system services are executed at the required time without being affected by the tasks and tag data link service. For NJ-series CPU Units, the system services are executed in the unused time between execution of the tasks and tag data link service.*3

*1 For details on the processing that is performed for tag data links, refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506) or the *CJ-series EtherNet/IP Unit Operation Manual for NJ-series CPU Unit* (Cat. No. W495). You can use CJ-series Units only with NJ-series CPU Units.

*2 Refer to *5-5-2 Processing Performed in and Execution Timing of the Tag Data Link Service* for details on the execution interval and the time that is required for execution of tag data links.

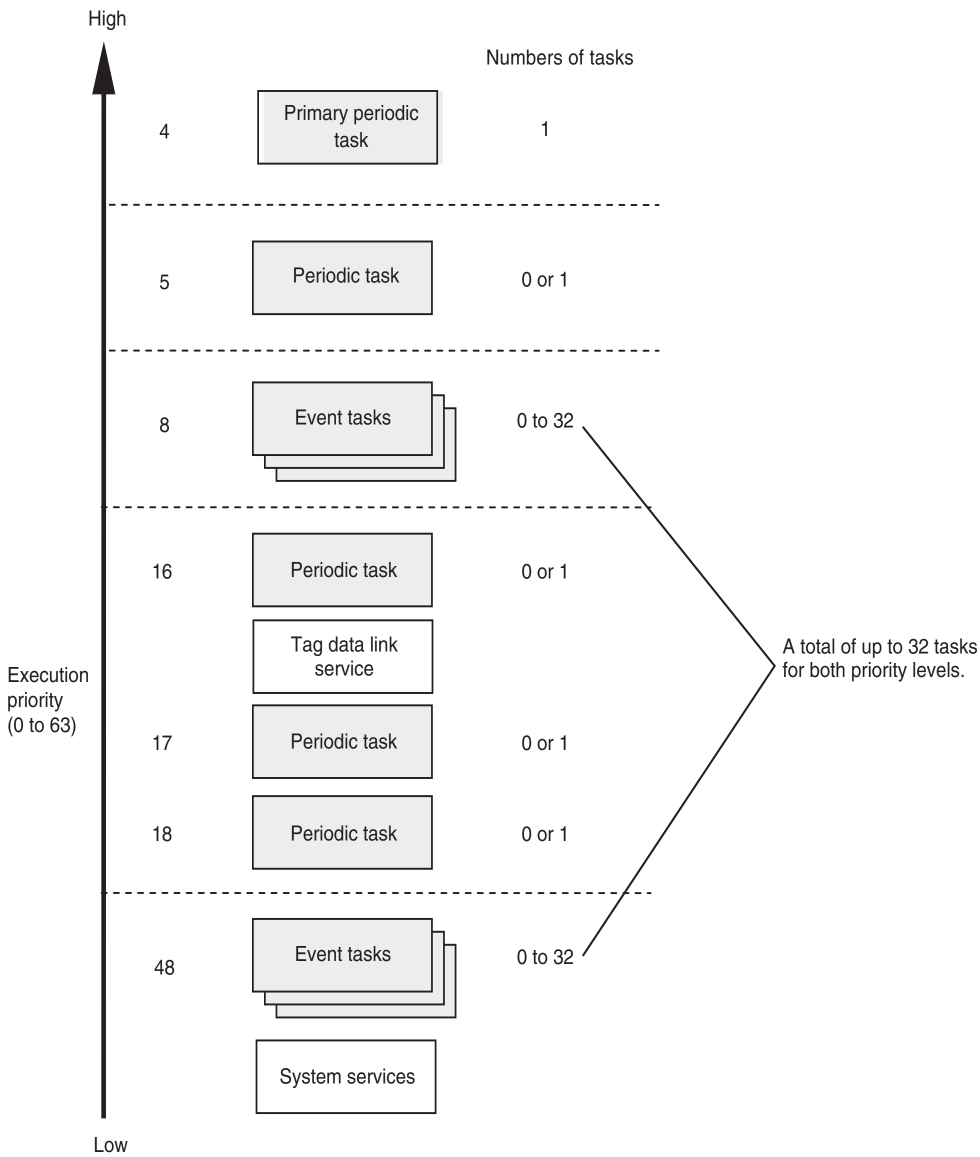
*3 If sufficient time cannot be obtained to execute the system services with an NJ-series CPU Unit, the processing of tasks with an execution priority of 17 or higher will be interrupted to allocate sufficient time. You can set the time for execution of the system services in the System Service Monitoring Settings on the Sysmac Studio.

5-5-1 Execution Priorities and Execution Orders of the Tag Data Link Service and System Services

This section provides examples of the execution priorities and execution orders of the tag data link service and system services.

Execution Priorities of the Tag Data Link Service and System Services

The execution priorities of the tag data link service and system services are shown below. The execution priority of the tag data link service is between the execution priorities of the priority-16 periodic task and the priority-17 periodic task. The execution priority of the system services is lower than the execution priority of any of the tasks.



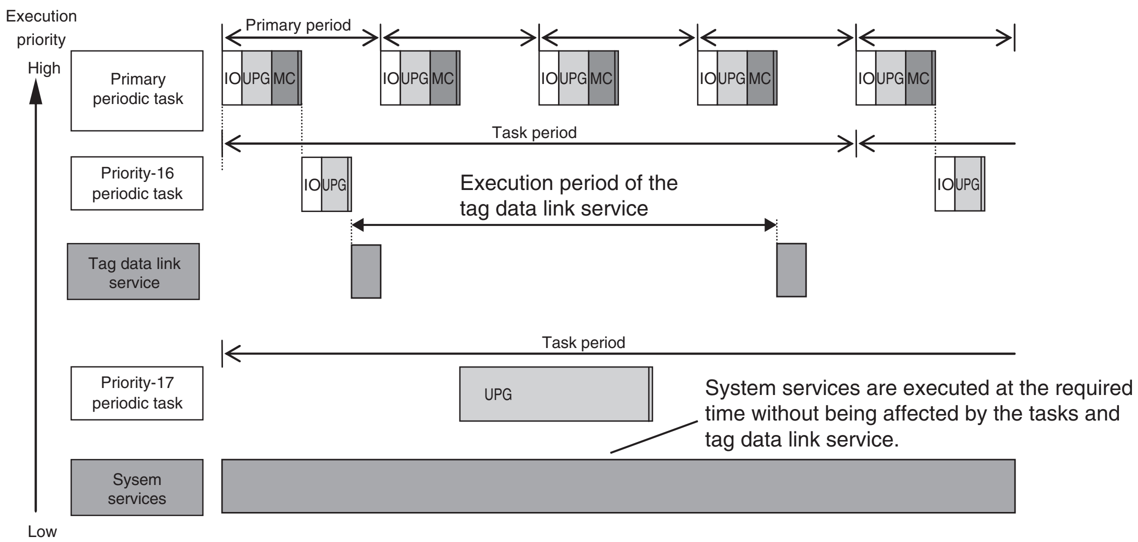
Note You can use the priority-5 periodic task only with NX-series CPU Units.

Examples of the Order of Execution for the Tag Data Link Service and System Services

As an example, the order of execution for the primary periodic task, priority-16 periodic task, priority-17 periodic task, tag data link service, and system services is shown below.

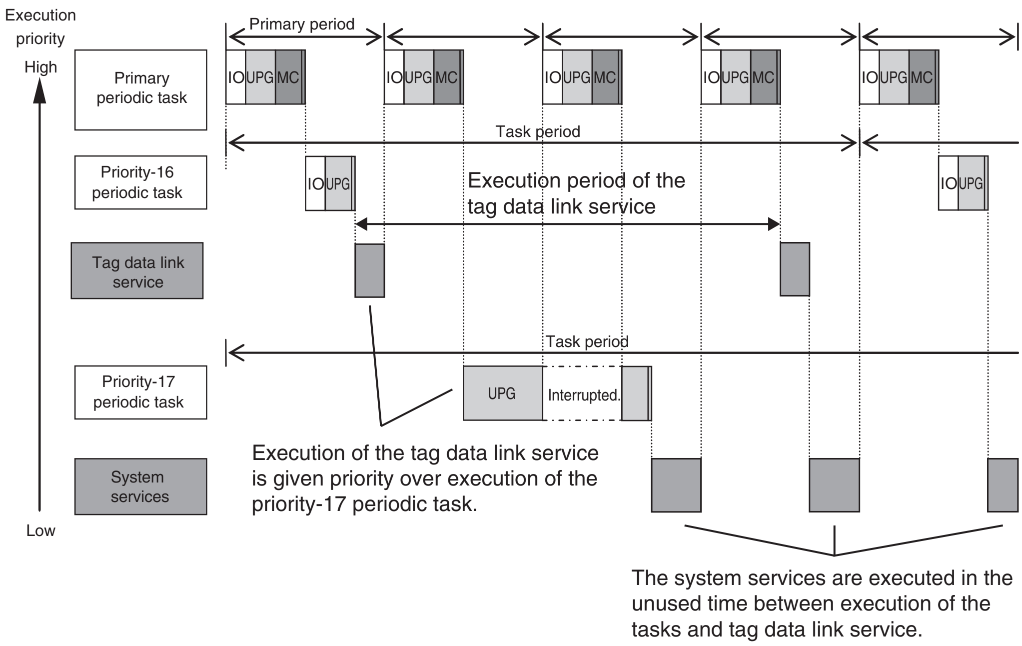
● NX-series CPU Units

- Execution of the tag data link service is given priority over execution of the priority-17 periodic task. However, execution of the primary periodic task and priority-16 periodic task is given even higher priority.
- The system services are executed at the required time without being affected by the tasks and tag data link service.



NJ-series CPU Units

- Execution of the tag data link service is given priority over execution of the priority-17 periodic task. However, execution of the primary periodic task and priority-16 periodic task is given even higher priority.
- The system services are executed in the unused time between execution of all of the tasks and tag data link service.



The execution interval and the execution time for one execution depend on the model of the CPU Unit and on the tag data link settings. Refer to *5-5-2 Processing Performed in and Execution Timing of the Tag Data Link Service* for details.

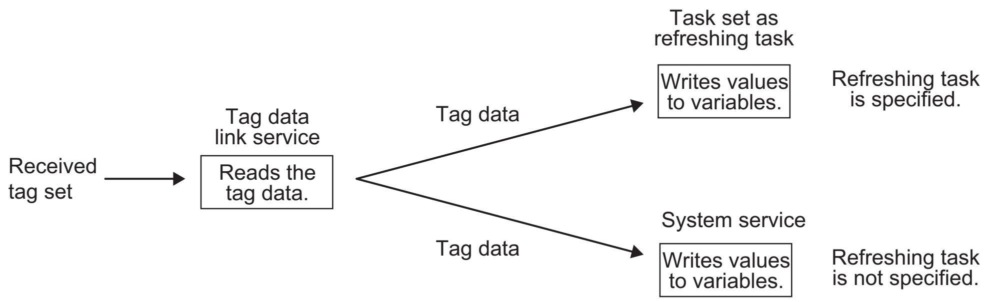
5-5-2 Processing Performed in and Execution Timing of the Tag Data Link Service

Processing Performed in Tag Data Link Service

The processing for tag data links is separated in multiple processes. This processing is performed in the tag data link service, the system services, and the tasks. The following example shows the processing that is performed for the tag data links when the built-in EtherNet/IP port on the CPU Unit is used.

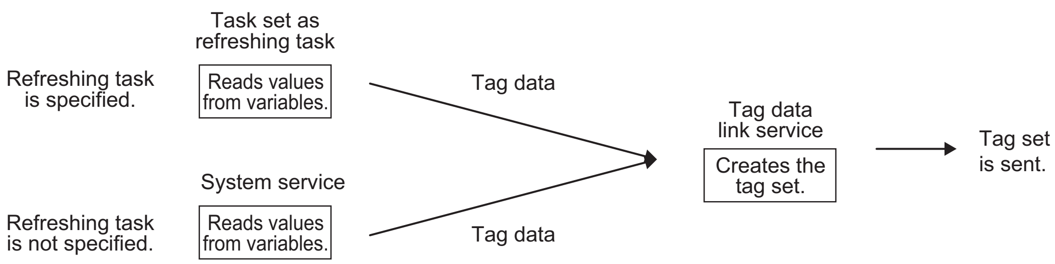
Flow of Tag Data Reception Processing

- The tag data link service reads the tag data from the received tag sets.
- If a refreshing task is set, the task writes the values of the tag data to the variables that are assigned to the tags.
- If a refreshing task is not set, a system service writes the values of the tag data to the variables that are assigned to the tags.



Flow of Tag Data Transmission Processing

- If a refreshing task is set, the task reads the values of the tag data from the variables that are assigned to the tags.
- If a refreshing task is not set, a system service reads the values of the tag data from the variables that are assigned to the tags.
- Then, in the tag data link service, the tag set is created from the tag data and the tag set is sent.



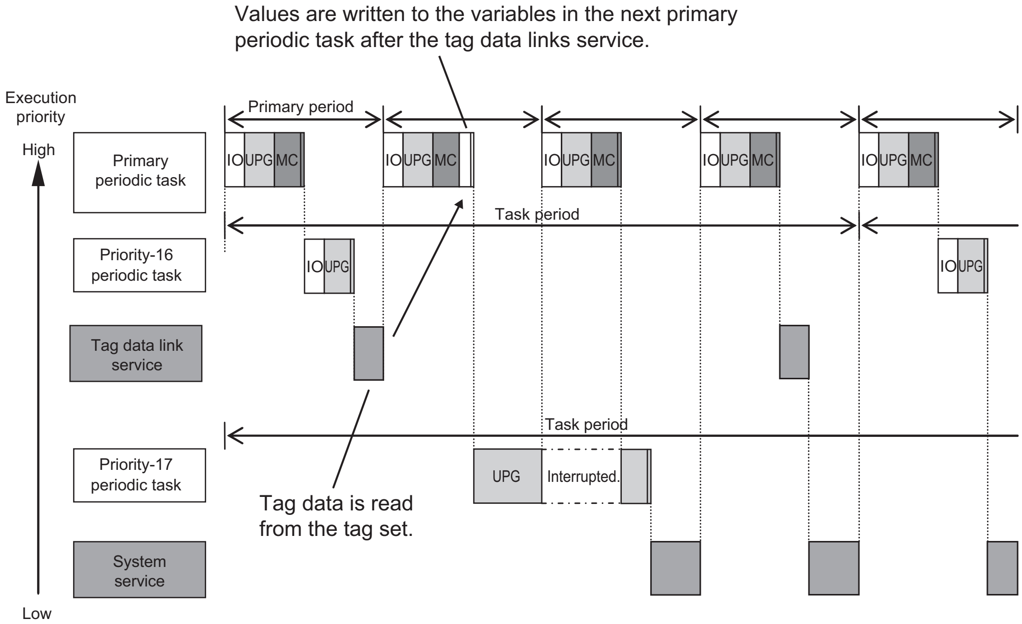
Additional Information

Differences in the Timing of Processing Tag Data Links Depending on Whether a Refreshing Task Is Set

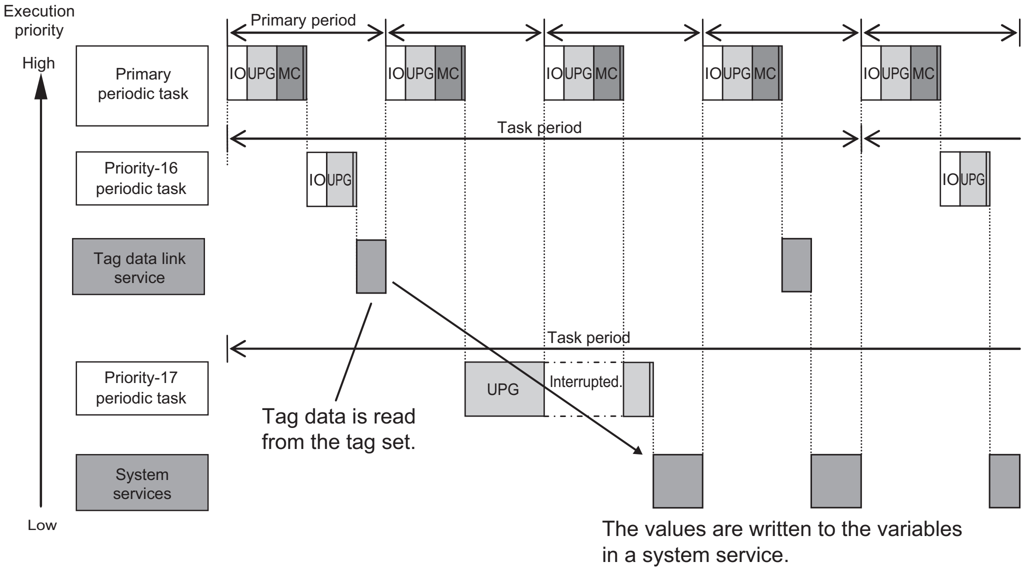
The process to write values to and read values from variables is different depending on whether a refreshing task is set. If a refreshing task is set, the values are read and written in that task. If a refreshing task is not set, the values are read and written in a system service. This means there is a difference in the execution timing of processing tag data links.

The difference in the execution timing when data is received for tag data links with an NJ-series CPU Unit is shown below.

- When a Refreshing Task Is Specified
  In this example, the primary periodic task is set as the refreshing task. Values are written to the variables in the next primary periodic task after the tag data links service.

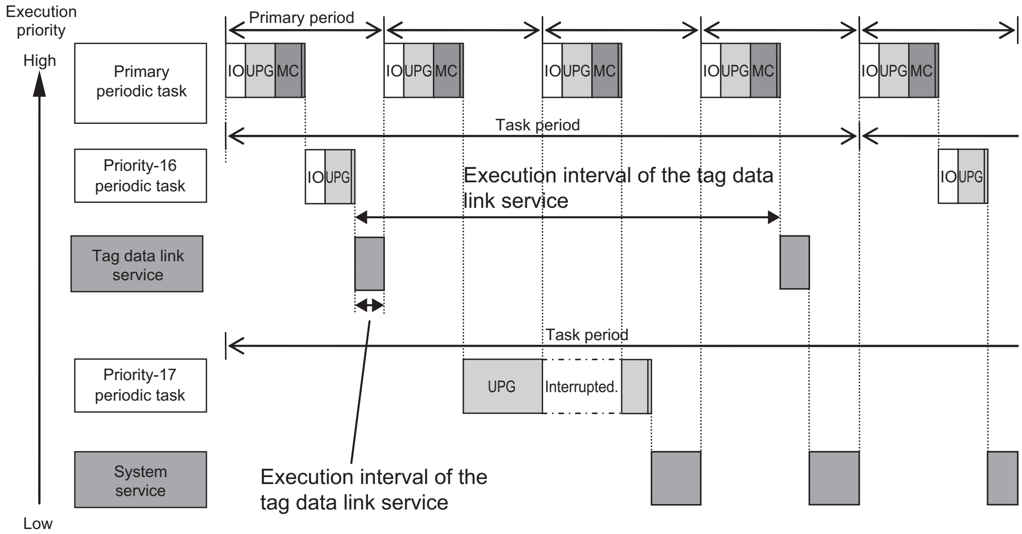


- When a Refreshing Task Is Not Specified
  The values are written to the variables in a system service.



Execution Timing of the Tag Data Link Service

The execution interval and the execution time depend on the model of the CPU Unit and on the tag data link settings. Guidelines are provided below. For an NJ-series CPU Unit, however, if a task with a higher execution priority than the tag data link service is executed, the execution interval and the execution time will be longer than the following values.



● Using the Built-in EtherNet/IP Port on the CPU Unit

CPU Unit	Tag data link settings: Number of tags	Tag data link settings: Number of connections	Tag data link settings: RPI [ms]	Execution interval (ms)	Execution time (ms)
NJ501-□□□□	8	1	1	1	0.02 to 0.04
		4	2	1	0.02 to 0.10
		32	50	1	0.02 to 0.10
NJ301-□□□□	8	1	1	1	0.03 to 0.06
		4	2	1	0.03 to 0.15
		32	50	1	0.03 to 0.15
NJ101-□□□□	8	1	1	1	0.04 to 0.09
		4	2	1	0.04 to 0.28
		32	50	1	0.04 to 0.28

Additional Information

For an NX-series CPU Unit, the priority-17 and priority-18 periodic tasks and the system service are not affected by execution of the tag data link service.

● Using the CJ1W-EIP21 EtherNet/IP Unit

CPU Unit	Tag data link settings				Execution interval (ms)	Execution time (ms)
	Number of tags	Number of connections	RPI [ms]	Send/receive		
NJ501-□□□□	8	1	1	Send	3	0.10
				Receive	3	0.10
		4	2	Send	4	0.22
				Receive	4	0.13
		32	50	Send	12	0.68
				Receive	6 to 35	0.23 to 0.66
NJ301-□□□□	8	1	1	Send	3	0.16
				Receive	3	0.16
		4	2	Send	3	0.21
				Receive	4	0.33
		32	50	Send	15	1.05
				Receive	8 to 35	0.29 to 1.85
NJ101-□□□□	8	1	1	Send	3	0.21
				Receive	3	0.21
		4	2	Send	6	0.46
				Receive	13	0.80
		32	50	Send	12 to 50	2.97 to 4.09
				Receive	25	2.36

Version Information

Execution of processing for tag data links depends on the unit version of the CPU Unit and the Sysmac Studio version as given below.

Unit version of CPU Unit	Sysmac Studio version	
	1.03 or lower	1.04 or higher
1.03 or later	Built-in EtherNet/IP port service in system services	Tag data link service*
1.02 or earlier	Built-in EtherNet/IP port service in system services	

* If a unit version of 1.02 or earlier is set on the Sysmac Studio, the tag data links are processed in the built-in EtherNet/IP port service in the system services.

5-5-3 Processing Performed in and Execution Timing of the System Services

Processing Performed in System Services

System services include the following processing.

System service	Description
USB port service	• Processing of service requests from the Sysmac Studio or host computers.
Built-in EtherNet/IP port service	• Processing of message service requests, such as CIP commands, from the Sysmac Studio, an HMI, host computers, or other Controllers • Execution of communications instructions for CIP and socket communications
Built-in EtherCAT port service	• Execution of EtherCAT message communications

System service	Description
Service for CJ-series Special Units*1	• Event servicing for CJ-series Special Units • Execution of communications instructions (CIP)
SD Memory Card service	• Access from FTP client • SD Memory Card operations from the Sysmac Studio • Execution of SD Memory Card instructions
Self-diagnosis	• Hardware error detection

*1 The CPU Unit exchanges data between CJ-series Special Units and the memory words that are allocated to them during I/O refreshing. You can use CJ-series Special Units only with NJ-series CPU Units.

Execution Timing of the System Services

For NX-series CPU Units, the system services are executed at the required time without being affected by the tasks and tag data link service. It is designed to always secure sufficient time for system service execution.

For NJ-series CPU Units, the system services are executed in the unused time between execution of the tasks and tag data link service.

If sufficient time cannot be obtained to execute the system services, the processing of tasks with an execution priority of 17 or higher will be interrupted to allocate sufficient time.

You can set the time for execution of the system services in the System Service Monitoring Settings on the Sysmac Studio.

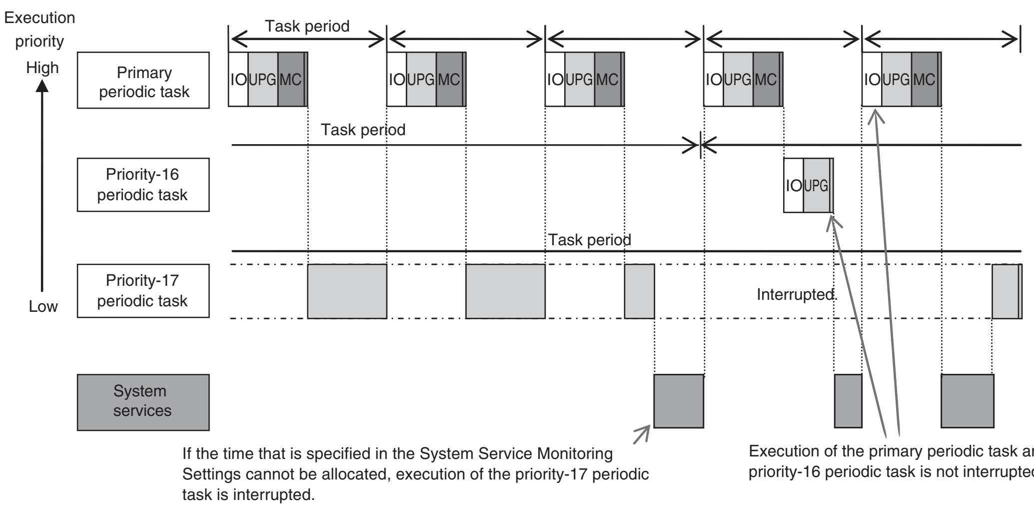
There is no priority in the processing of system services. All of the processing is executed in parallel with time slicing.

● Execution Timing When Sufficient Execution Time for the System Services Cannot Be Obtained

With an NJ-series CPU Unit, if there is not enough unused time between execution of the tasks and tag data link service to execute the system services, the execution of tasks with an execution priority of 17 or higher will be interrupted to allocate sufficient time.

If the system service execution time cannot be obtained even if execution of tasks with an execution priority of 17 or higher is temporarily interrupted, an Insufficient System Service Time Error occurs and user program execution stops.

You can set the time for execution of the system services in the System Service Monitoring Settings on the Sysmac Studio.



System Service Monitoring Settings

The System Service Monitoring Settings are used to monitor whether sufficient system service execution time can be secured with NJ-series CPU Units. If the execution of all tasks cannot be completed within the system service execution interval, execution is monitored to see if at least the ratio of execution that is set for the System Service Execution Time Ratio is completed. You can set the System Service Monitoring Settings in the Basic Settings Display of the Operation Settings Tab Page on the Sysmac Studio.

Access point	Setting group	Setting [unit]	Description	Set values	Default	Update timing	Changes in RUN mode
Operation Settings, Operation Settings Tab, Basic Settings	System Service Monitoring Settings	System Service Execution Interval [ms]	Sets the interval of system service execution.	10 ms to 1 s	10 ms	When transferred to CPU Unit	Not allowed.
		System Service Execution Time Ratio [%]	Sets the ratio for monitoring system service execution.	5% to 50%	10%	When transferred to CPU Unit	Not allowed.

Additional Information

NX-series CPU Units are designed to always secure sufficient time for system service execution, so the System Service Monitoring Settings are not provided. Also an Insufficient System Service Time Error will not occur.

Precautions for Correct Use

- For NJ-series CPU Units, set the System Service Monitoring Settings to the minimum values that are required to meet the response performance of the system services so that sufficient time can be allocated to the system services and task execution. The System Service Monitoring Settings are used to monitor whether the specified system service execution time can be obtained. System services will not necessarily be executed for the specified time.
- For NJ-series CPU Units, design the tasks so that sufficient time can be allocated to execution of the system services. Refer to *5-10 Task Design Methods and I/O Response Times* for the setting procedures for tasks.
- To increase the system service execution time with an NJ-series CPU Unit, increase the task period or take other steps to increase the unused time between task execution.
- With an NJ-series CPU Unit, if the time that is specified in the System Service Monitoring Settings cannot be allocated to the system service execution time even if execution of tasks with an execution priority of 17 or higher is interrupted, an Insufficient System Service Time Error occurs and user program execution stops.
- With an NJ-series CPU Unit, if sufficient system service execution time cannot be allocated and execution of tasks with an execution priority of 17 or higher is interrupted, a Task Period Exceeded error will occur for the tasks that are interrupted. Design the tasks so that the execution of tasks with an execution priority of 17 or higher is completed within the task periods even if the execution time of the system services satisfies the System Service Monitoring Settings.

5-6 Assignment and Settings Related to Tasks

This section describes the assignment and settings related to tasks.

5-6-1 Assigning I/O Refreshing to Tasks

I/O refreshing of the EtherCAT slaves and the CJ-series Units is assigned to the tasks.

Tasks to which assignment is possible and unit of assignment are different depending on the target for I/O refreshing. Unit of assignment refers to a target or a group of targets for I/O refreshing assigned to one task. For example, when the unit of assignment is Slave Terminal, you can assign I/O refreshing to only one task even if more than one NX Unit is connected to a Communications Coupler Unit.

If you want to perform input and output operations in tasks to which I/O refreshing is not assigned, refer to *Input and Output Operations in Tasks to Which I/O Refreshing Is Not Assigned* on page 5-64.

The following table shows the relationship among the target for I/O refreshing, the assignable task, and the unit of assignment.

I/O refreshing target	Assignable task	Unit of assignment
Communications Coupler Unit with an NX-series Safety Control Unit on the Slave Terminal	Primary periodic task or priority-5 and priority-16 periodic tasks*1*2	By Slave Terminal
Communications Coupler Unit with an NX Unit assigned to an axis on the Slave Terminal	Primary periodic task or priority-5 periodic task	
Communications Coupler Unit without an NX Unit assigned to an axis on the Slave Terminal	Primary periodic task or priority-5 and priority-16 periodic tasks*2	
EtherCAT slaves to which axes are assigned	Primary periodic task or priority-5 periodic task*2	By slave
Other EtherCAT slaves	Primary periodic task or priority-5 and priority-16 periodic tasks*2	
CJ-series Basic I/O Units*3	Primary periodic task and priority-16 periodic task	By Unit
CJ-series Special I/O Units*3		
CJ-series CPU Bus Units*3		

*1 If multiple Slave Terminals with mounted NX-series Safety Control Units exist, you cannot assign Slave Terminals to a combination of the periodic tasks other than priority-5 periodic task and priority-5 periodic task.

*2 You can use the priority-5 periodic task only with NX-series CPU Units.

*3 You can use various CJ-series Units only with NJ-series CPU Units.

Precautions for Safe Use

If two different function modules are used together, such as when you use CJ-series Basic Units and EtherCAT slaves, take suitable measures in the user program and external controls to ensure that safety is maintained in the controlled system if one of the function modules stops.

The relevant outputs will behave according to the slave or Unit specifications if a partial fault level error occurs in one of the function modules.

Refer to *12-1-3 Non-fatal Errors in the CPU Unit* for details on partial fault level Controller errors.

Additional Information

When using both EtherCAT slaves and CJ-series Units, you can shorten the task execution time of the primary periodic task by assigning the I/O refreshing for EtherCAT slaves to a different task than the I/O refreshing for CJ-series Units. Assign I/O refreshing for EtherCAT slaves to the primary periodic task and assign I/O refreshing for CJ-series Units to the priority-16 periodic task.

Sysmac Studio Setting Procedure

For the slaves and Units that are not assigned to axes, set the tasks in which to perform I/O refreshing in **I/O Control Task Settings** under **Configurations and Setup - Task Settings** of the Sysmac Studio.

Refer to *I/O Control Task Settings* on page 4-9 for details.

For the slaves and Units that are assigned to axes, specify the motion controls to use in **Motion Control Setup** under **Configurations and Setup** of the Sysmac Studio. The tasks to perform I/O refreshing are set.

Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for details.

Timing of I/O Refreshing

The table below shows when I/O is refreshed for EtherCAT slaves and CJ-series Units.

I/O refreshing target	I/O control task	Period of I/O refreshing
EtherCAT slaves	Primary periodic task	Task period of the primary periodic task
	Priority-5 periodic task*1	Task period of the priority-5 periodic task
	Priority-16 periodic task	Task period of the primary periodic task*2
CJ-series Units*3	Primary periodic task	Task period of the primary periodic task
	Priority-16 periodic task	Task period of the priority-16 periodic task

*1 You can use the priority-5 periodic task only with NX-series CPU Units.
*2 EtherCAT communications is executed during I/O refreshing in the primary periodic task. If the priority-16 periodic task is used to control EtherCAT slaves, the data refresh period during I/O refreshing will be the task period of the priority-16 periodic task.
*3 You can use CJ-series Units only with NJ-series CPU Units.

Priorities in Process Data Communications

EtherCAT process data communications is executed during I/O refreshing. With an NX-series CPU Unit, you can perform process data communications in the primary periodic task and the priority-5 periodic task.

When process data communications is executed in the primary periodic task or the priority-5 periodic task, process data communications in the primary periodic task is prioritized. Therefore, the I/O refresh time of the priority-5 task is longer even if the process data communications cycle and process data size of two tasks are the same.

Accessing I/O from the User Program

You use device variables to access I/O ports from the user program. Access the device variables from a program in the task that is set in the I/O Control Task Settings.

Input and Output Operations in Tasks to Which I/O Refreshing Is Not Assigned

If you attempt to output data directly from a task to which I/O refreshing is not assigned, an error will occur when you check the program on the Sysmac Studio. For example, if the processes for NX Units are assigned to different tasks on the Slave Terminal, data can be output only from the task to which I/O refreshing is assigned. In this case, you need to create the program to pass the data from the task to which I/O refreshing is not assigned to the task to which I/O refreshing is assigned and output data from the task to which I/O refreshing is assigned.

The following sample programming shows how to pass the data from the task to which I/O refreshing is not assigned to the task to which I/O refreshing is assigned and output data from the task to which I/O refreshing is assigned.

In this sample programming, the external data input processing is performed in a task to which I/O refreshing is not assigned. Usually, this kind of processing causes a warning to occur when you check the program on the Sysmac Studio. However, the execution of processing is possible.

Unit Configuration

A Slave Terminal is used. The following table shows the Unit configuration of the Slave Terminal.

Model number	Product name
NX-ECC201	EtherCAT Coupler Unit
NX-ID3317	DC Input Unit
NX-OD3256	Transistor Output Unit

I/O Map

The following I/O map is used. The table below shows the bits that are used in the sample programming.

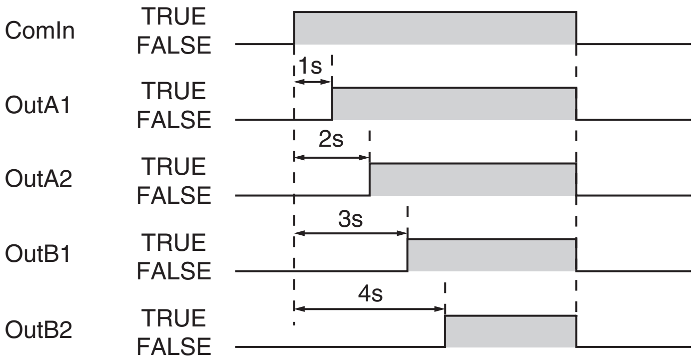
Position	Port	Description	R/W	Data type	Variable	Variable comment	Variable type
Unit1	NX-ID3317						
	Input Bit 00	Input bit 00	R	BOOL	ComIn	Common input value	Global variable
Unit2	NX-OD3256						
	Output Bit 00	Output bit 00	W	BOOL	OutA1	Output value A1	Global variable
	Output Bit 01	Output bit 01	W	BOOL	OutA2	Output value A2	Global variable
	Output Bit 02	Output bit 02	W	BOOL	OutB1	Output value B1	Global variable
	Output Bit 03	Output bit 03	W	BOOL	OutB2	Output value B2	Global variable

● I/O Specifications

The I/O specifications are as follows:

- *OutA1* changes from FALSE to TRUE one second after the value of *ComIn* changes from FALSE to TRUE. In the same way, *OutA2* changes to TRUE two seconds, *OutB1* three seconds and *OutB2* four seconds after the value of *ComIn* changes to TRUE.
- When the value of *ComIn* changes from TRUE to FALSE, the values of *OutA1*, *OutA2*, *OutB1* and *OutB2* change from TRUE to FALSE.

The following figure shows the timing chart.



● Task Processing

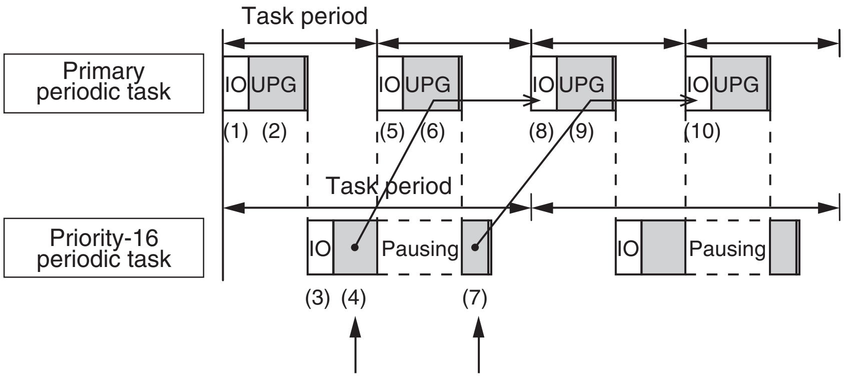
The primary periodic task and priority-16 periodic task are used. The I/O refreshing is assigned to the primary periodic task.

OutA1 and *OutA2* are controlled in the primary periodic task. A series of processing, i.e., input of *ComIn*, calculations and outputs of *OutA1* and *OutA2*, is performed in the primary periodic task.

For controls of *OutB1* and *OutB2*, the processing from input of *ComIn* to calculations is performed in the periodic task. The calculation results of the periodic task are assigned to the temporary global variables *tmp_OutB1* and *tmp_OutB2*. In the primary periodic task, the values of *tmp_OutB1* and *tmp_OutB2* are assigned to *OutB1* and *OutB2*. Then, *OutB1* and *OutB2* are output.

The Controller may pause the periodic task in order to execute the primary periodic task. The assignment of calculation results to *tmp_OutB1* and *tmp_OutB2* may be performed before or after the pause of the periodic task. And the timing of assignment determines the timing of output of calculation results in the primary periodic task.

The following figure shows an example of processing flow. The task period of the periodic task is set to twice as long as that of the primary periodic task.



If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), they are output in step (8).

If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (7), they are output in step (10).

(1) *ComIn* is input during I/O refreshing in the primary periodic task.

**(2) The calculation is performed during the user program execution in the primary periodic task and the values are assigned to *OutA1* and *OutA2*.
Also, the values of *tmp_OutB1* and *tmp_OutB2* are assigned to *OutB1* and *OutB2*. However, the values of *tmp_OutB1* and *tmp_OutB2* are the initial values because the values are not assigned to *tmp_OutB1* and *tmp_OutB2* in the periodic task. Therefore, the values of *OutB1* and *OutB2* are also the initial values.**

(3) *ComIn* is input during I/O refreshing in the periodic task.

(4) The calculation is performed during the user program execution in the periodic task. If the primary periodic task is executed while the periodic task execution is in progress, the periodic task is paused. Depending on the timing of processing in the periodic task, the assignment of the values to *tmp_OutB1* and *tmp_OutB2* may be performed before or after the pause of the periodic task.

**(5) *OutA1*, *OutA2*, *OutB1* and *OutB2* are output during I/O refreshing in the primary periodic task.
Also, *ComIn* is input again.**

**(6) The calculation is performed during the user program execution in the primary periodic task and the values are assigned to *OutA1* and *OutA2*.
Also, the values of *tmp_OutB1* and *tmp_OutB2* are assigned to *OutB1* and *OutB2*. If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), the calculation results of the periodic task are reflected in *OutB1* and *OutB2*. If the values are not assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), the values of *tmp_OutB1* and *tmp_OutB2* are the initial values. Therefore, the values of *OutB1* and *OutB2* are also the initial values.**

**(7) The calculation is performed during the user program execution in the periodic task that follows step (4).
If the values are not assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), they are assigned here.**

**(8) *OutA1*, *OutA2*, *OutB1* and *OutB2* are output during I/O refreshing in the primary periodic task.
If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), the calculation results of the periodic task are output as *OutB1* and *OutB2*.
If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (7), the initial values of *OutB1* and *OutB2* are output.**

(9) The calculation is performed during the user program execution in the primary periodic task and the values are assigned to *OutA1* and *OutA2*.
Also, the values of *tmp_OutB1* and *tmp_OutB2* are assigned to *OutB1* and *OutB2*. If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (7), the calculation results of the periodic task are reflected in *OutB1* and *OutB2*.

(10) *OutA1*, *OutA2*, *OutB1* and *OutB2* are output during I/O refreshing in the primary periodic task.
If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (7), the calculation results of the periodic task are output as *OutB1* and *OutB2*.

You can use the Lock and Unlock instructions to perform the task exclusive controls to prevent the values of *tmp_OutB1* and *tmp_OutB2* from being overwritten by the periodic task before they are accessed by the primary periodic task. Refer to *5-7-1 Ensuring Concurrency of Variable Values between Tasks* for details on task exclusive controls.

● Global Variable Table

The global variables are shown below.

Global variable table

Name	Data type	Initial value	AT specification	Comment
ComIn	BOOL	FALSE	ECAT://node#[1,1]/Input Bit 00	Common input value
OutA1	BOOL	FALSE	ECAT://node#[1,2]/Output Bit 00	Output value A1
OutA2	BOOL	FALSE	ECAT://node#[1,2]/Output Bit 01	Output value A2
OutB1	BOOL	FALSE	ECAT://node#[1,2]/Output Bit 02	Output value B1
OutB2	BOOL	FALSE	ECAT://node#[1,2]/Output Bit 03	Output value B2
tmp_OutB1	BOOL	FALSE		Temporary variable for B1
tmp_OutB2	BOOL	FALSE		Temporary variable for B2

● Ladder Diagram for Primary Periodic Task

The ladder diagram for the primary periodic task is shown below.

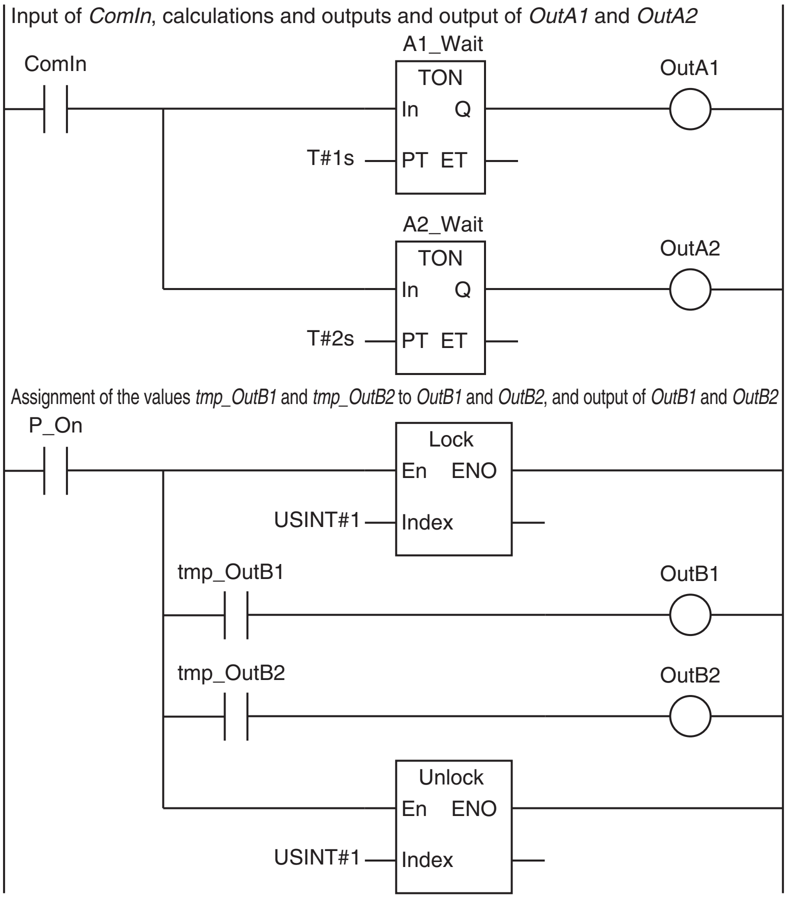
Internal variable table

Name	Data type
A1_Wait	TON
A2_Wait	TON

External variable table

Name	Data type	Comment
ComIn	BOOL	Common input value
OutA1	BOOL	Output value A1
OutA2	BOOL	Output value A2
OutB1	BOOL	Output value B1
OutB2	BOOL	Output value B2
tmp_OutB1	BOOL	Temporary variable for B1
tmp_OutB2	BOOL	Temporary variable for B2

Algorithm



- **Ladder Diagram for Periodic Task**

The ladder diagram for the periodic task is shown below.

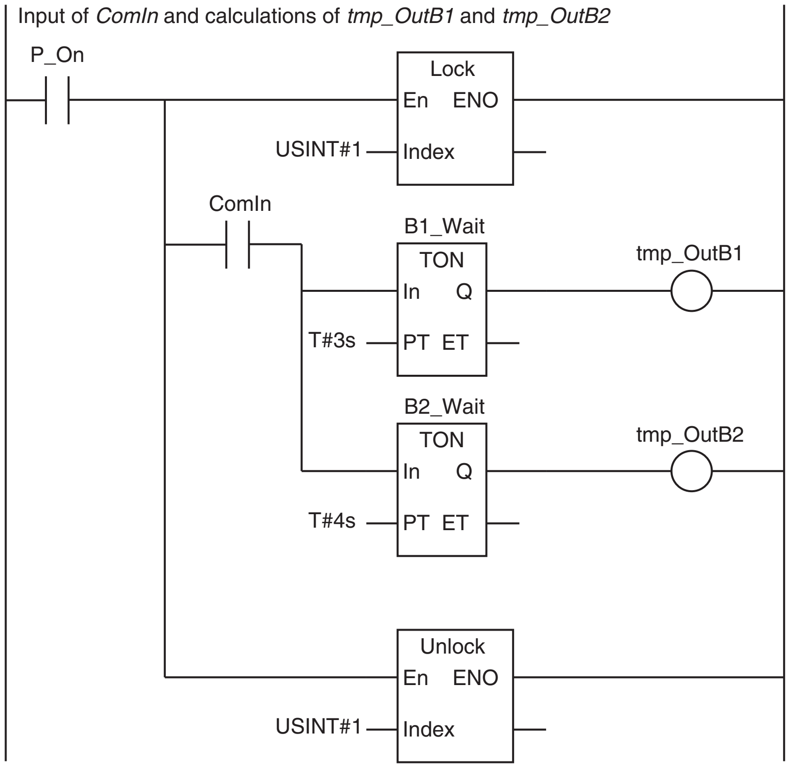
Internal variable table

Name	Data type
B1_Wait	TON
B2_Wait	TON

External variable table

Name	Data type	Comment
ComIn	BOOL	Common input value
tmp_OutB1	BOOL	Temporary variable for B1
tmp_OutB2	BOOL	Temporary variable for B2

Algorithm for periodic task



5-6-2 Assigning Tasks to Programs

You assign the programs to execute to tasks. (You can assign up to 128 programs to one task.) Also, you set the operation of the programs at the start of operation.

Order of Program Execution

The order of execution of the programs in a task is set with the Sysmac Studio.

Initial Status for Programs at the Start of Operation

Set the operation of the programs at the start of operation. The *Initial Status* at the start of operation is used to set whether to execute the program when the task to which the program is assigned is executed for the first time after the operating mode of the CPU Unit is changed from PROGRAM mode to RUN mode. You have a setting option between *Run* or *Stop*.
If the *Initial Status* is *Stop*, when enabling the execution of the specified program with the PrgStart instruction, it is executed from the next time the timing for executing the program occurs. If the *Initial Status* is *Run*, when disabling the execution of the specified program with the PrgStop instruction, it is disabled from the next time the timing for executing the program occurs.

● Sysmac Studio Setting Procedure

Assign programs to tasks, set the order of program execution within the task, and set the *Initial Status* for each program in **Program Assignment Settings** under **Configurations and Setup** – **Task Settings** of the Sysmac Studio.

Refer to *Program Assignment Settings* on page 4-9 for details.

Version Information

The CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the *Initial Status* setting for programs at the start of operation, the PrgStart instruction and the PrgStop instruction.

POUs That You Can Assign to Tasks

From 0 to 128 programs can be assigned to one task.
You can assign only program POUs. You cannot assign function block instances or functions directly to tasks. You cannot assign the same program to more than one task.

5-6-3 Parameters for Primary Periodic Task and Periodic Tasks

The parameters for primary periodic task and periodic tasks are given below.

● Parameters for Primary Periodic Tasks

Parameter		Setting range	Default	Update timing	Changes in RUN mode
Task Type		Specify the primary periodic task.	---	When transferred to CPU Unit	Not allowed.
	Execution priority	Always 4.	---		
Task Name		Text string	Primary-Task		
Period/Execution Conditions	Task period*1	• NX-series CPU Units 125 µs, 250 µs to 8 ms (in 250-µs increments) • NJ-series CPU Units 500 µs*2, 1 ms, 2 ms, or 4 ms	1 ms*3		
Task Period Exceeded Detection		Specify whether to detect an error if the task execution time exceeds the specified task period. • Detect (a minor fault level Controller error is generated). • Do not detect (an observation is recorded in event log). Refer to *Task Period Exceeded* on page 5-83 for details.	Detect.		
Task Timeout Detection Time		Set the time to detect timeouts if task execution does not end, e.g., if there is an infinite loop. Set a multiple of the task period. 1 to 5 Refer to *Task Execution Timeout* on page 5-84 for details.	5		
Variable Access Time [%]		Set the percentage of the task period to assign to variable access. 1% to 50% Refer to *Settings for Variable Access Time* on page 5-81 for details.	3%		

*1 The process data communications cycle (process data communications cycle 1) in the EtherCAT settings will be the same as this period.

*2 With the NJ301-□□□□, you can use this setting with unit version 1.03 or later.
You cannot use this setting with the NJ101-□□□□.

*3 For an NJ101-□□□□ CPU Unit, the default of the primary periodic task is 2 ms.

● Parameters for Priority-5 Periodic Task

Parameter		Setting range	Default	Update timing	Changes in RUN mode
Task Type		Specify the priority-5 periodic task.*1	---	When transferred to CPU Unit	Not allowed.
	Execution priority	Automatically set to 5.	---		
Task Name		Text string	PeriodicTask0		
Period/ Execution Conditions	Task period*2	• NX-series CPU Units 125 µs, 250 µs to 100 ms (in 250-µs increments)	2 ms		
Task Period Exceeded Detection		The same as for the primary periodic task.	The same as for the primary periodic task.		
Task Timeout Detection Time					
Variable Access Time [%]					

*1 You can use the priority-5 periodic task only with NX-series CPU Units.
*2 The process data communications cycle 2 in the EtherCAT settings will be the same as this period.

● Parameters for Priority-16, Priority-17, and Priority-18 Periodic Tasks

Parameter		Setting range	Default	Update timing	Changes in RUN mode
Task Type		You can set any of the following. Priority-16 periodic task Priority-17 periodic task Priority-18 periodic task	---	When transferred to CPU Unit	Not allowed.
	Execution priority	Automatically set to 16, 17, or 18.	---		
Task Name		Text string	PeriodicTask0		
Period/ Execution Conditions	Task period	• NX-series CPU Units Refer to *5-3-1 Specifications of Tasks for NX-series Controllers*. • NJ-series CPU Units Refer to *5-4-1 Specifications of Tasks for NJ-series Controllers*.	10 ms		
Task Period Exceeded Detection		The same as for the primary periodic task.	The same as for the primary periodic task.		
Task Timeout Detection Time					
Variable Access Time [%]					

Event Task Parameters

Parameter	Setting range	Default	Update timing	Changes in RUN mode
Task Type	You can set any of the following. Priority-8 event task Priority-48 event task	---	When transferred to CPU Unit	Not allowed.
Task Name	Text string	EventTask0		
Execution Condition	Select either *Execution by instructions* or *When a variable expression is satisfied*.	Execution with an instruction		
Task Execution Timeout Time	Set the time to detect a timeout if task execution does not end, e.g., if there is an infinite loop. The setting unit is milliseconds. • Execution priority of 8: 1 to 500 ms • Execution priority of 48: 1 ms to 10 s	• Execution priority of 8: 200 ms • Execution priority of 48: 1 s		

Sysmac Studio Setting Procedure

Add and set the tasks in the **Task Settings** under **Configurations and Setup** on the Sysmac Studio.

Refer to *Task Settings* on page 4-7 for details.

5-7 Ensuring Concurrency of Variable Values

This section describes how to ensure concurrency of variable values between tasks and provides an overview of variable access from outside the Controller.

5-7-1 Ensuring Concurrency of Variable Values between Tasks

If more than one task reads or writes the same global variable, you can use either of the following two methods to ensure the concurrency of the value of the global variable between the tasks.

Method 1: Write the global variable from only one task and read the variable from the other tasks. Use the settings for exclusive control of variables in tasks.

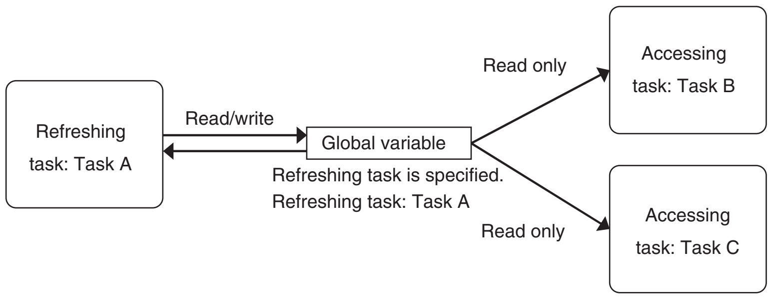
Method 2: With this method, you can write the global variable from more than one task. Use the task exclusive control instructions.

Method 1: Settings for Exclusive Control of Variables in Tasks

● Introduction

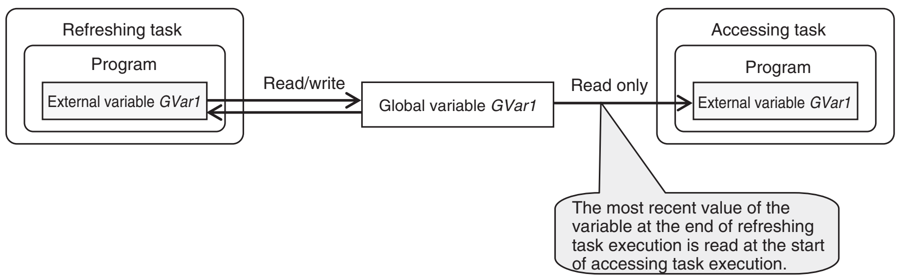
You can specify the task that refreshes a global variable and the tasks that access the global variable. This ensures the concurrency of the value of the global variable from the point of view of the tasks that access the variable.

A single task is set to read and write the value of a specified global variable. That task is called the refreshing task. Tasks that only read the value of the global variable are also specified. These tasks are called accessing tasks. This ensures the concurrency of the value of the global variable.



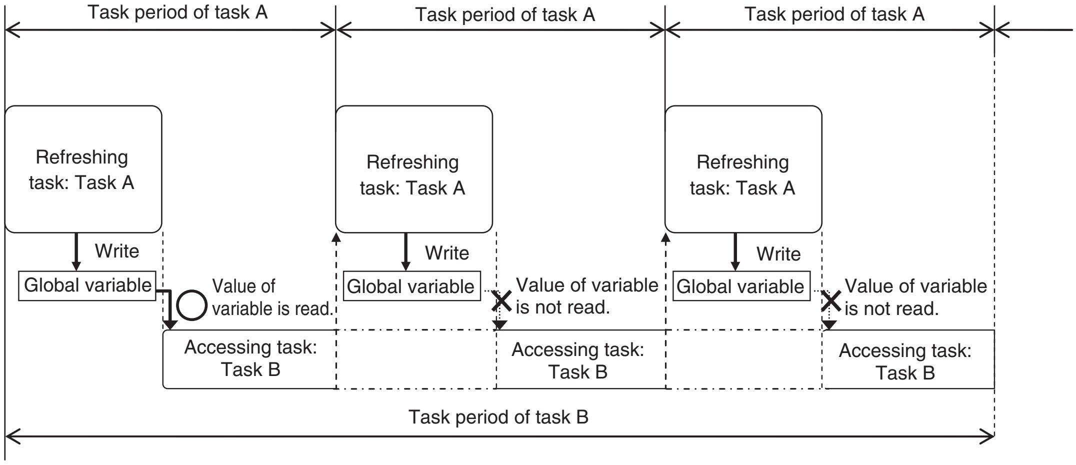
● Application Example

The refreshing task specification is used to ensure the concurrency of the value of a global variable within a periodic task when the variable is written in the primary periodic task.



● System

If a refreshing task is set for a global variable, the accessing task, at the start of accessing task execution, always reads the most recent value of the variable that was written at the completion of refreshing task execution.

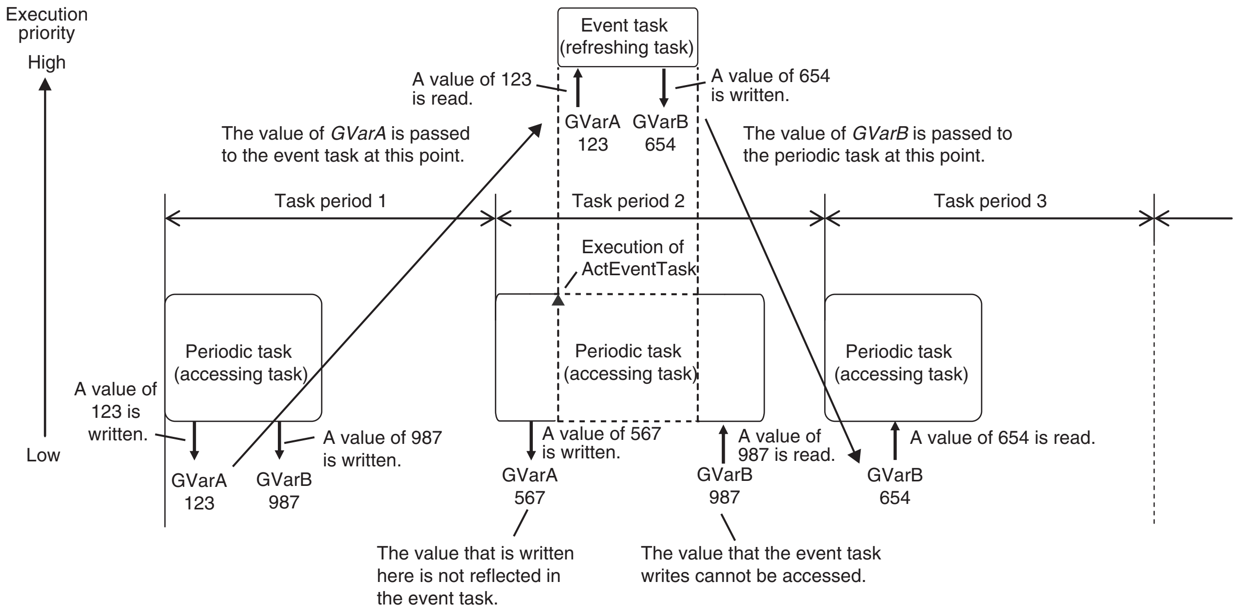


This will allow you to maintain the concurrency of the values of global variables within the tasks without performing any special programming.

If an instruction that writes the value to a global variable is used in the accessing task, an error will occur when you check the program on the Sysmac Studio.

Precautions for Correct Use

If you are using the ActEventTask instruction between two tasks, you must keep in mind when the global variables are accessed, and when they are refreshed. For example, in the following diagram, the value of the *GVarA* global variable that is accessed from the event task is the value that was current at the end of task period 1. Therefore, even if the periodic task in task period 2 writes the value of *GVarA*, that value will not be reflected in the event task. The value that the event task writes to the *GVarB* global variable is not passed to the periodic task until the start of task period 3. Even if the periodic task in task period 2 accesses the value of *GVarB*, the value that the event task writes will not be accessed.



Because of this, do not use exclusive control of variables in tasks to pass the values of global variable if you are using the ActEventTask instruction to execute event tasks. To ensure the concurrency of global variables when using the ActEventTask instruction, you should use the Task_IsActive (Determine Task Status) instruction. The Task_IsActive instruction determines whether the specified task is in execution or waiting to be executed. Use this instruction to prevent other tasks from accessing variables that the event task writes to while it is in execution. Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for details on the Task_IsActive instruction.

Restrictions

- Only one refreshing task can be set for each global variable. If it is necessary to write a global variable from more than one task, use the task exclusive control instructions described below to ensure concurrency.
- If you specify a refreshing task for a structure or union variable, you must specify only one refreshing task for the entire structure or union variable. You cannot specify a different refreshing task for different structure or union members.
- If you specify a refreshing task for an array variable, you must specify only one refreshing task for the entire array variable. You cannot specify a different refreshing task for different array elements.

Precautions for Correct Use

Do not write the value of a variable for which concurrency is required from any task that is not the refreshing task, e.g., do not write the value from the accessing task. If you read or write the value of a variable for which a refreshing task is set from any task that is not a refreshing or accessing task, the concurrency of the global variable may be lost. If you write such a program, a warning is given when the program is checked.

Additional Information

You can use a data trace to sample an external variable for a global variable for which settings for exclusive control of variables in tasks are used. This allows you to sample the values of the global variable in the refreshing and accessing tasks in a data trace. Refer to *8-5-4 Data Tracing* for information on data tracing.

● Sysmac Studio Setting Procedure

Set the global variables for which to specify refreshing tasks, and set the accessing tasks in the Settings for Exclusive Control of Variables in Tasks on the Task Settings Tab Page on the Sysmac Studio.

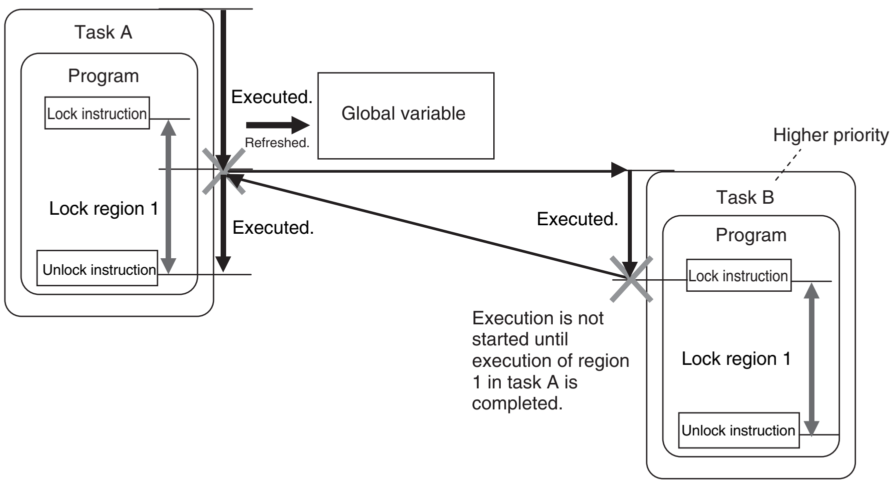
For details, refer to *Settings for Exclusive Control of Variables in Tasks* on page 4-10.

Method 2: Task Exclusive Control Instructions

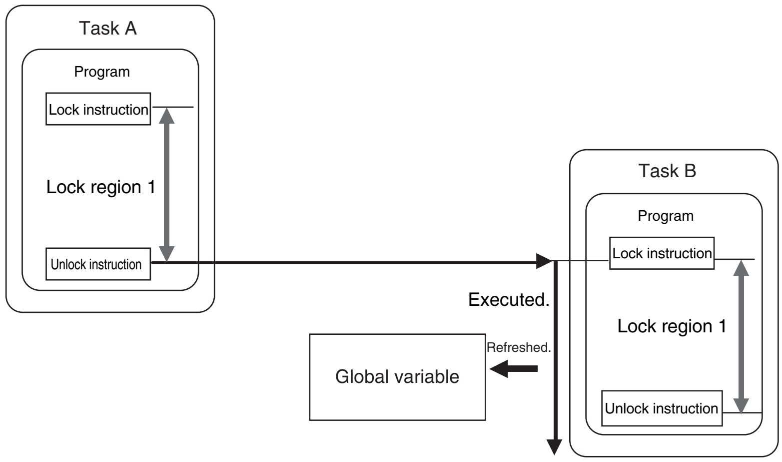
Use the task exclusive control instructions (i.e., the Lock and Unlock instructions) when it is necessary to write the value of a global variable from more than one task while maintaining concurrency in the value of the variable. The task exclusive control instructions create a lock region from one Lock instruction to the next Unlock instruction. If a lock region in one task is being executed, the lock regions with the same lock number in other tasks are not executed. If you place the instructions that write to the global variable in lock regions, the concurrency of the value is maintained even if you write the value of the variable from more than one task.
Refer to information on the Lock and Unlock instructions in the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for details.

Example:

In this example, task A and task B both have lock region 1. The priority of task B is higher than the priority of task A. If the execution condition for task B is met during execution of lock region 1 in task A, execution of task A is paused during lock region 1 and task B is executed. However, in this case, lock region 1 in task A is not completed, so task B is paused before it processes lock region 1. When task B is paused, execution of lock region 1 in task A is started again.



When execution of lock region 1 in task A is completed, task A is paused again and the remainder of lock region 1 in task B is executed. The concurrency of the value of the global variable is maintained by implementing exclusive control of the write processing of the global variable between the tasks.



Precautions for Correct Use

- Do not make the locked regions any longer than necessary. If the lock regions are too long, the task execution period may be exceeded.
- Always use the Lock and Unlock instructions in a pair in the same section of the same POU.

Example of Accessing the Same Global Variable Between Tasks

This section describes how to request processing to another task with multiple global variables.

The following sample programming uses one global variable *gReq* as an exclusive flag for processing to perform exclusive control in the user program. Even in this case, exclusive control of variables in tasks is required for the access logic to the *gPar1*, *gPar2*, and *gReq* global variables that are used between tasks.

A sample programming that uses the task exclusive control instructions (i.e., the Lock and Unlock instructions) to perform exclusive control is shown.

● Global Variables

Name	Data type	Comment
gReq	BOOL	Request flag
gPar1	ULINT	Parameter 1
gPar2	DATA_AND_TIME	Parameter 2

● Task That Makes Processing Requests (MainTask)

• Internal Variables

Name	Data type	Comment
ReqTrg	BOOL	Request trigger
cCnt	ULINT	100-ms counter value
cTime	DATA_AND_TIME	Current time

• ST Program

```
cCnt:=Get100msCnt();                      (* Get the 100-ms counter value. *)
cTime:=GetTime();                         (* Get the current time. *)

Lock(1);                                  (* Start an exclusive lock between tasks. *)
IF ReqTrg=TRUE AND gReq=FALSE THEN        (* Access the exclusive flag. *)
  gPar1:=cCnt;                            (* Set the parameter to process in SubTask. *)
  gPar2:=cTime;                           (* Set the parameter to process in SubTask. *)
  gReq:=TRUE;
  ReqTrg:=FALSE;
END_IF;
Unlock(1);                                (* Stop an exclusive lock between tasks. *)
```

● Task That Receives Processing Requests (SubTask)

• Internal Variables

Name	Data type	Comment
ReqBusy	BOOL	---
UserDefFB_ins	UserDefFB	User-defined function block instance that executes processing

• ST Program

```
Lock(1);                                  (* Start an exclusive lock between tasks. *)
IF gReq=TRUE AND ReqBusy=FALSE THEN       (* Access the exclusive flag. *)
  ReqBusy:=TRUE;
  UserDefFB_ins.PutData:=gPar1;           (* Read the parameter from MainTask. *)
  UserDefFB_ins.PutDate:=gPar2;           (* Read the parameter from MainTask. *)
  gReq:=FALSE;                            (* Reset the exclusive flag. *)

  UserDefFB_ins.Execute:=TRUE;
END_IF;
Unlock(1);                                (* Stop an exclusive lock between tasks. *)

UserDefFB_ins();

IF UserDefFB_ins.Done:=TRUE THEN
  UserDefFB_ins.Execute:=FALSE;
  ReqBusy:=FALSE;
END_IF;
```

5-7-2 Variable Access from Outside the Controller

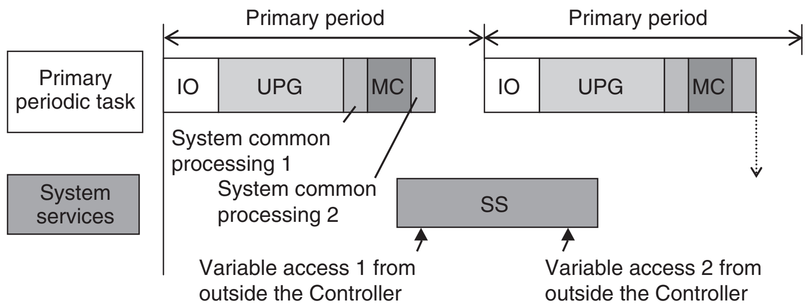
A variable access from outside the Controller is executed during the system service. The system service has a lower execution priority than tasks. This means, if multiple variables are accessed from outside the Controller, refreshing all variable values may not be completed in a task period. If refreshed variables and not-refreshed variables are mixed in the user program, the Controller may perform unintended operation.

To avoid this, make the variable access from outside the Controller be executed during the system common processing 2 of the task. By making this, multiple variable values can be securely refreshed in the same task period.

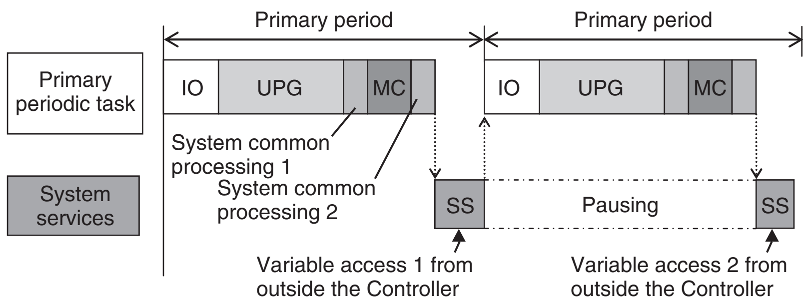
● Accessing Variables from Outside the Controller during the System Service

Whether you use an NX-series CPU Unit or an NJ-series CPU Unit, access to multiple variables may not be completed in the same task period.

● NX-series CPU Unit

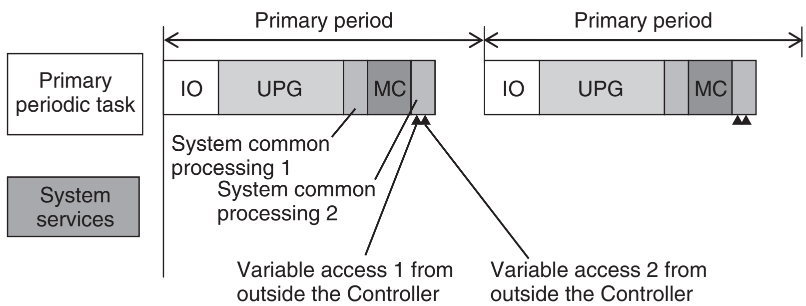


● NJ-series CPU Unit



● Accessing Variables from Outside the Controller during the System Common Processing 2

Whether you use an NX-series CPU Unit or an NJ-series CPU Unit, access to multiple variables is executed in the same period.



This section particularly describes how to execute the variable access from outside the Controller during the system common processing 2 of the task.

Methods to Access Variables From Outside the Controller

There are the following four methods to access variables from outside the Controller.

- Sysmac Studio
- NA/NS-series PT
- EtherNet/IP tag data links
- CIP communications instruction from the host computer

If the Sysmac Studio is used to access variables, it can only refresh the variable values during the system common processing 2 of the task. Values are accessed during the system services.

Tasks that Execute Variable Access during the System Common Processing 2

The tasks that execute variable access from outside the Controller during the system common processing 2 are predetermined as follows according to the variable types.

Variable	Tasks that refresh values
Global variables specified in the settings for exclusive control of variables in tasks	The refreshing task specified in **Settings for Exclusive Control of Variable in Tasks** under **Configurations and Setup** – **Task Settings** on the Sysmac Studio.
Device variables for EtherCAT slaves	Tasks specified in **I/O Control Task Settings** under **Configurations and Setup** – **Task Settings** on the Sysmac Studio.
Device variables for CJ-series Basic I/O Units	
Device variables for CJ-series Special Units	Primary periodic task
Variables with AT specifications in memory used for CJ-series Units	

Note You can use CJ-series Units and memory for CJ-series Units only with NJ-series CPU Units.

Settings for Executing Variable Access during the System Common Processing 2

To access variables from outside the Controller during the system common processing 2, it is necessary to make the following two settings on the Sysmac Studio.

- Settings for exclusive control of variables in tasks (when the target variables are the global variables)
- Settings for Variable Access Time

Settings for Exclusive Control of Variables in Tasks

If global variables are accessed from outside the Controller during the system common processing 2 of the task, it is necessary to make setting for exclusive control of variables in tasks. The Exclusive Control of Variables in Tasks refers to the function that specifies the task that can refresh the target global variable. This function prevents the target variable from being updated by other tasks or by other methods to access variables from outside the Controller.

For the details on the exclusive control of variables in tasks, refer to *Settings for Exclusive Control of Variables in Tasks* on page 4-10.

Precautions for Correct Use

When you use EtherNet/IP tag data links, always specify the same task as the refreshing task for all tags (variables that have Network Publish attribute) in the same tag set. Otherwise, multiple tags in a tag set may be refreshed in separate task periods.

Settings for Variable Access Time

When variable access from outside the Controller is executed during the system common processing 2 of the task, the task execution time may be longer. The user must set the upper limit of the processing time for accessing variables on the Sysmac Studio. The Variable Access Time refers to the upper limit of the processing time for accessing variables.

● Calculating Variable Access Time

Use the following equation for calculating the variable access time.

Variable access time [μs] = total size of variables [bytes] * a + number of variables * b + number of accesses * c + d

The values of the constants a to d in above equation vary depending on the type of the CPU Unit.

CPU Unit model	Constant value [μs]			
	a	b	c	d
NX701-□□□□	0.0005	0.033	2.67	7.22
NJ501-□□□□	0.001	0.49*1	1.41	6.68
NJ301-□□□□	0.0015*2	0.56*3	2.15	7.52
NJ101-□□□□	0.0015	1.07	3.83	10.29

*1 The constant value is 0.58 for a CPU Unit with unit version 1.02 or earlier.

*2 The constant value is 0.0009 for a CPU Unit with unit version 1.02 or earlier.

*3 The constant value is 1.03 for a CPU Unit with unit version 1.02 or earlier.

● Setting Variable Access Time

Set the variable access time in the **Task Settings** Display under **Configurations and Setup** – **Task Settings** on the Sysmac Studio. The setting must be made for each task by entering the ratio to the task period. The default value is 3%. For the details on the settings, refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504).

● Example of Variable Access Time Setting

The following is an example of variable access time setting.

In this example, it is assumed that there are the following three variable accesses from outside the Controller to the task that operates in the NJ501-□□□□ CPU Units.

Access No.	Source of variable access	Total size of variables to access [bytes]	Number of variables to access	Number of accesses
1	EtherNet/IP tag data link	600	8	1
2	EtherNet/IP tag data link	200	4	1
3	CIP communications instruction	1,000	1	1

Using the equation, the variable access time for Access No.1 is calculated as follows.

Variable access time for Access No.1 = 600 * 0.001 + 8 * 0.49 + 1 * 1.41 + 6.68

= 12.61 [µs]

In the same way, you can calculate the access time for the other accesses and get the following values.

Access No.	Variable Access Time [µs]
1	12.61
2	10.25
3	9.58

If only one of these accesses occurs in one task period, you set the Variable Access Time to the one for Access No.1, which requires the longest access time.

The variable access time for Access No.1 is 12.61 µs. Therefore, when the task period is 500 µs, the Variable Access Time is set to 12.61/500 ≈ 3%.

If every access occurs once in one task period, the Variable Access Time is calculated with the equation as follows.

Variable access time = (600 + 200 + 1000) * 0.001 + (8 + 4 + 1) * 0.49 + (1 + 1 + 1) * 1.41 + 6.68

= 19.08 [µs]

When the task period is 500 µs, the Variable Access Time is set to 19.08/500 ≈ 4%.

● Processing in the case that actual variable access time became longer than set value

If actual variable access time became longer than the set value, the following processing is performed depending on the number of variable accesses in one task period.

Set a sufficiently long access time so that multiple variable accesses can be completed within a task period.

Number of variable accesses in one task period	Processing
Multiple times	Variable accesses are executed for the number of times that can be completed within the set variable access time. The rest of accesses that could not be done will be executed in the next task period. This means, the multiple variable accesses cannot be completed within the same task period.
Once	Variable accesses continue even after the set variable access time is exceeded. This means, the task execution time gets longer.

5-8 Errors Related to Tasks

This section describes the following errors.

- Task Period Exceeded
- Motion Control Period Exceeded
- Task Execution Timeout
- I/O Refreshing Timeout Error
- Insufficient System Service Time Error

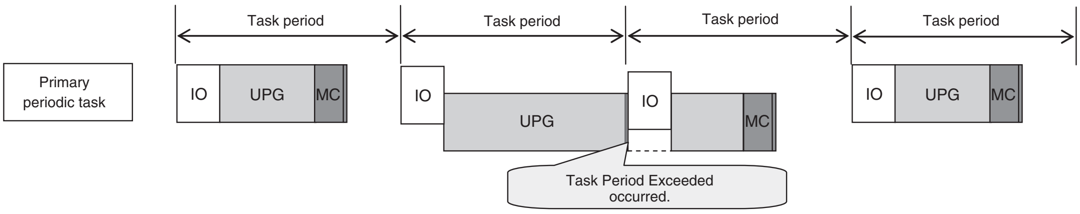
Task Period Exceeded

A Task Period Exceeded error occurs if the task execution time exceeds the specified task period.

This is a minor fault level Controller error. Operation continues even when this error occurs.

It can occur for the primary periodic task and periodic tasks.

You can also disable the Task Period Exceeded errors with a setting. Use the Task Period Exceeded Detection setting in the Task Settings of the Sysmac Studio. The default setting is to detect the error.



Error name	Error level	Correction
Task Period Exceeded	Minor fault	Review the task settings and programs and download the project again. Reset the error from the Sysmac Studio.

Even if the Task Period Exceeded Detection setting is disabled, information will be output to the following system-defined variables if task processing is not completed within the period: Task Period Exceeded Flag (_*TaskName_Exceeded*), Task Period Exceeded Count (_*TaskName_ExceedCount*), Controller Error Status (_*ErrSta*), and the event log.

I/O is refreshed as follows according to what the I/O is for if task processing is not completed within the task period.

I/O is for	I/O refresh operation if task processing is not completed within the task period
EtherCAT slave*1	Outputs: The values from the previous period are output. Inputs: Inputs are refreshed, but the input data is not updated in the executed user program.
CJ-series Unit	I/O is not refreshed until execution of the task is completed.

*1 This includes NX Units on EtherCAT Slave Terminals.

Precautions for Correct Use

If the Task Period Exceeded error occurs, shorten the programs to fit in the task period or increase the setting of the task period.

Motion Control Period Exceeded

A Motion Control Period Exceeded error occurs if the motion control processing (MC) is not completed within the primary period (i.e., the motion control period) twice or more in a row. A partial fault level Controller error will occur in the Motion Control Function Module. A Task Period Exceeded error will occur at the same time.

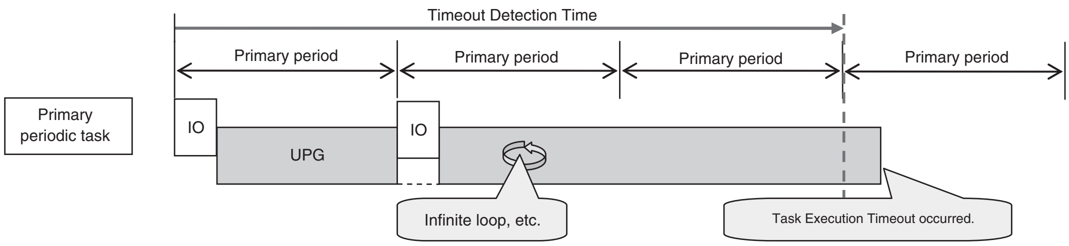
Error name	Error level	Correction
Motion Control Period Exceeded	Partial fault	Reduce the amount of processing in the programs or increase the control period within the range that does not adversely affect operation.

Task Execution Timeout

A Task Execution Timeout error occurs if task processing is not completed within the specified Task Execution Timeout Time.

This is a major fault level Controller error. Execution of the user program stops when the error occurs.

This error also occurs when normal task operation is not possible due to errors in program logic, such as infinite loops.

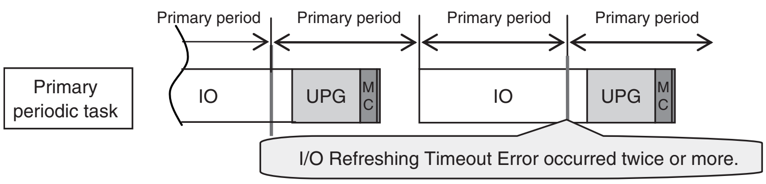


Error name	Error level	Correction
Task Execution Timeout	Major fault	Review the task settings and download the user program again. The power supply must be cycled or the CPU Unit reset.

I/O Refreshing Timeout Error

An I/O Refreshing Timeout Error occurs when I/O refreshing for the primary periodic task or a priority-16 periodic task is not completed within the period twice or more in a row.

This is a major fault level Controller error. Execution of the user program stops when the error occurs.



Error name	Error level	Correction
I/O Refreshing Timeout Error	Major fault	Review the task settings and download the project again. The power supply must be cycled or the CPU Unit reset.

Insufficient System Service Time Error

With an NJ-series CPU Unit, an Insufficient System Service Time Error occurs if the system service execution time that is specified in the System Service Monitoring Settings cannot be obtained.
This is a major fault level Controller error. Execution of the user program stops when the error occurs.

Error name	Error level	Correction
Insufficient System Service Time Error	Major fault	Review the task settings and the system service monitoring settings and download the project again. The power supply must be cycled or the CPU Unit reset.

Additional Information

NX-series CPU Units are designed to always secure sufficient time for system service execution, so the System Service Monitoring Settings are not provided. Also an Insufficient System Service Time Error will not occur.

5-9 Monitoring Task Execution Status and Task Execution Times

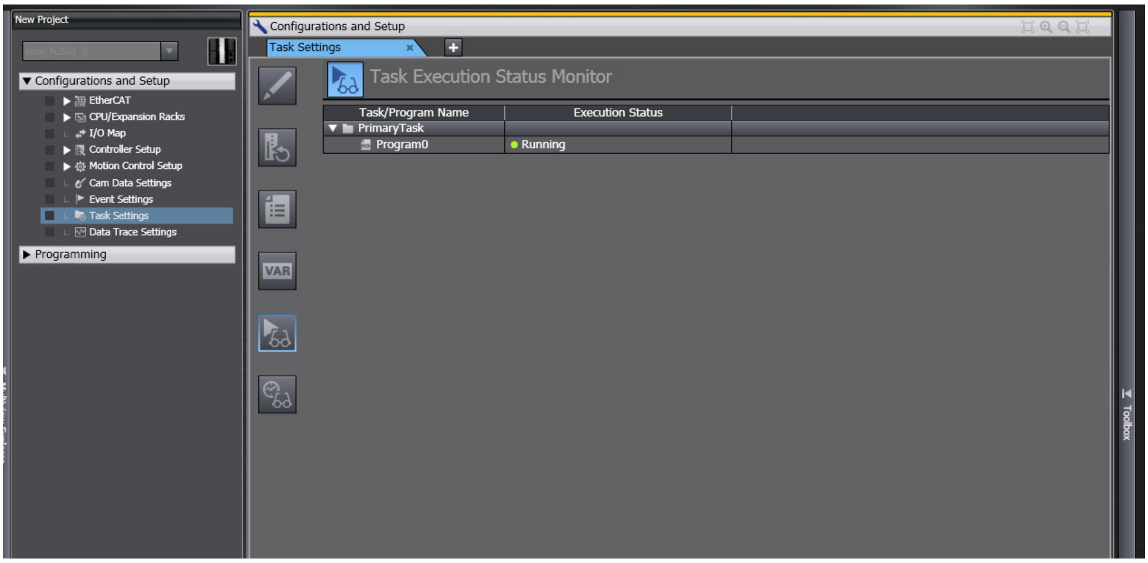
You can use online operations from the Sysmac Studio to monitor the task execution status and task execution times.

Monitoring Task Execution Status

You can monitor the execution status of the programs in all of the tasks (started/stopped) from the Sysmac Studio.

● Sysmac Studio Operation

Place the Sysmac Studio online with the CPU Unit and select **Configurations and Setup** – **Task Settings**. Click the **Task Execution Status Monitor** Button to display the following window.



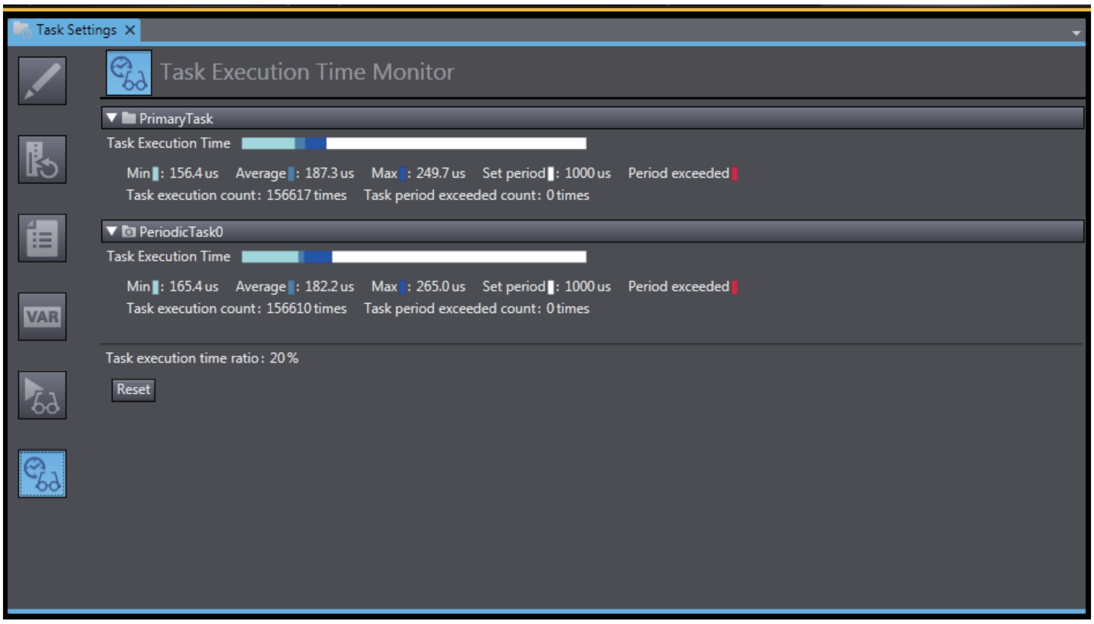
Task Execution Time Monitor

You can monitor the execution time of each task from the Sysmac Studio.

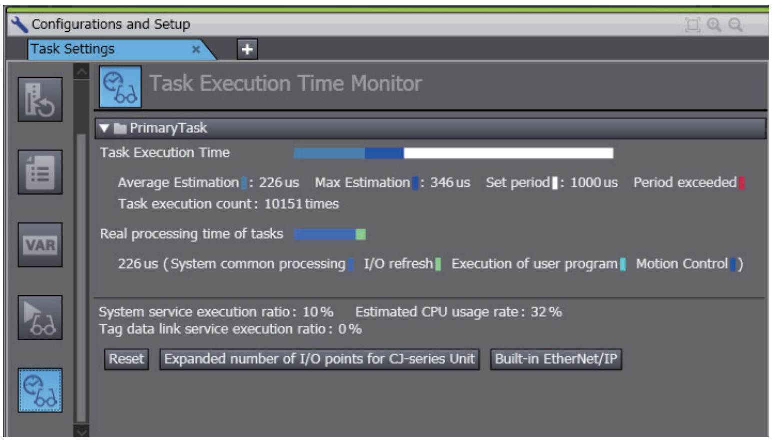
● Values You Can Monitor from the Sysmac Studio

The display depends on whether you connect to the physical Controller or to the Simulator.

Connected to the Controller



Connected to the Simulator

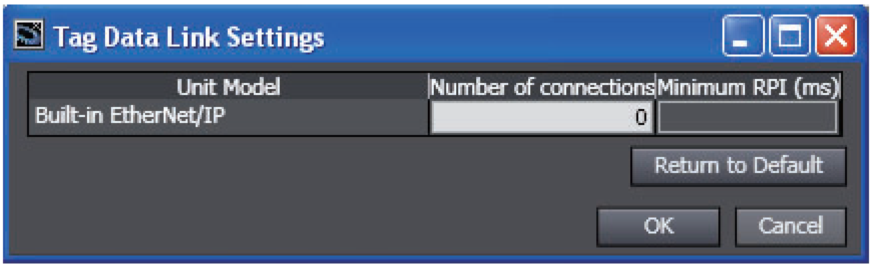


Additional Information

To check or estimate the system service execution time, the task execution time ratio, system service execution ratio, estimated CPU usage rate, tag data service execution ratio, expanded number of I/O points for CJ-series Unit, and built-in EtherNet/IP are displayed for NJ-series CPU Units.

The above information are not displayed for NX-series CPU Units because the system services are executed at the required time without being affected by the tasks and tag data link service.

Built-in EtherNet/IP Dialog Box for Simulator Connection



The parameters are listed in the following table.

Port or Unit that is used	Parameter	Description	Set values	Default
CPU Unit Built-in EtherNet/IP Port	Number of connections	This is the number of tag data link connections.	0 to 32	0
	Minimum RPI	This is the lowest packet interval (RPI) that is set for all of the tag data link connections.	1 ms to 10 s in 1-ms increments	---
CJ-series EtherNet/IP Unit (CJ1W-EIP21)[*1]	Number of tags	This is the number of tags in the tag data links.	0 to 256	0

*1 Entries are made on the Expanded number of I/O points for CJ-series Unit Dialog Box.

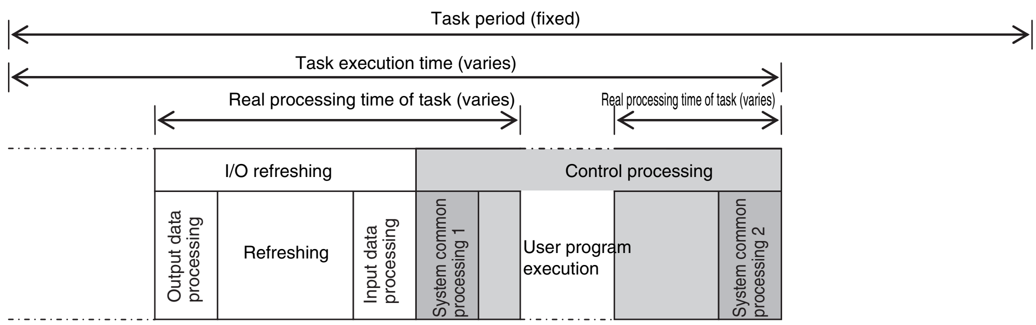
You can monitor the following items.

Monitor item		Description	Connected to the Controller	Connected to the Simulator
Task execution time[*1]	Min.	The minimum value of the task execution time.	Displayed.	Not displayed.
	Average	The maximum value of the task execution time.		Displayed.
	Max.	The maximum value of the task execution time.		
Set period		The specified task period.[*2]		
Period exceeded		If the task execution time exceeds the task period (i.e., if the Task Period Exceeded Flag system-defined variable is TRUE), the amount by which the time was exceeded is displayed in the bar.[*2]		
Task execution count		Displays the number of executions of the task. The value of the Task Execution Count system-defined variable is displayed.		
Real processing time of tasks[*3]		The time ratios are displayed with bars for the system common processing, I/O refreshing, user program execution, and motion control processing. (Specific time values are not displayed.)	Not displayed.	Displayed.
	Average estimation	The estimated average value of the real processing time of task is displayed.		
	Max estimation	The estimated maximum value of the real processing time of task is displayed.		
Task execution time ratio[*4]		The percentage of the total task execution time per unit time.	Displayed.	Not displayed.

*1 This is the actual time required from the point that task execution was started until it was completed. This interval includes both the time to execute other tasks and the time for system services that were executed from when task execution was started until it was completed.
Only the primary periodic task is displayed when connected to the Simulator in an NX-series CPU Unit.

*2 This item is not displayed for event tasks.

*3 This interval is the time required to execute only the task itself. It is the same as the task execution time for the primary periodic task. For periodic tasks, this is the task execution time minus the time to execute other tasks and the time for system services that were executed between the point that the execution condition is met until execution is completed.



*4 This item was added for version 1.12 of the Sysmac Studio. It is not displayed for an NX-series CPU Unit.

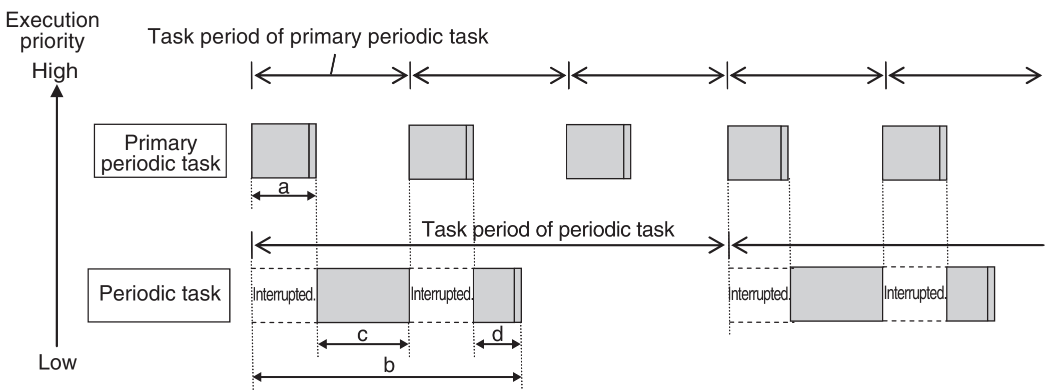
Precautions for Correct Use

The above values when connected to the Simulator of the Sysmac Studio may contain more error in comparison to the times when connected to the physical Controller. Use them as guidelines. Always confirm operation while connected to the physical Controller to study the designs and before starting actual system operation.

Meaning of the Task Execution Time and the Real Processing Time of the Task

The Task Execution Time and Real Processing Time of Tasks that are displayed in the Monitor View for the execution time of tasks are described in the following table. The Real Processing Time of Tasks shows only the time the Simulator was connected.

Displayed time	Meaning
Task execution time	This is the time from when the task period starts until task processing ends. However, this includes the time when the periodic task is interrupted for execution of tasks with higher execution priorities.
Real processing time of task	This is the time from when the task period starts until task processing ends. However, this does not include the time when the periodic task is interrupted for execution of tasks with higher execution priorities.



a: Task execution time of primary periodic task
b: Task execution time of periodic task
c+d: Real processing time of periodic tasks (only when the Simulator is connected).

5-10 Task Design Methods and I/O Response Times

This section provides guidelines for designing tasks, information on estimating task execution times, information on confirming system service monitoring settings, an example of task designing, and information on I/O response times.

The primary periodic task and periodic tasks of an NJ/NX-series Controller operate according to the specified task periods. If the actual execution time exceeds the task period, an error occurs.

This section uses an example that consists of one primary periodic task to describe estimation and appraisal methods.

Precautions for Safe Use

The task execution times in the physical Controller depends on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors. Before starting actual operation, you must test performance under all foreseeable conditions on the actual system and make sure that the task periods are not exceeded and that suitable communications performance is achieved.

5-10-1 Checking the Task Execution Time

Always design your system so that the average and maximum task execution times that are estimated with the methods that are described in this section sufficiently fit within the specified task periods.

● Desktop Calculations

First, refer to *A-2 Calculating Guidelines for the Real Processing Times of Tasks for the NX-series System* and *A-3 Calculating Guidelines for the Real Processing Times of Tasks for the NJ-series System* to make a rough estimate of the average task execution time on paper. You cannot estimate the maximum value on paper.

● Estimating with the Simulator on the Sysmac Studio

Use the Task Execution Time Monitor of the Simulator on the Sysmac Studio to estimate the average and maximum task execution times. Use the following procedure to check operation on the Simulator.

1. Create the Unit and slave configurations, create the global variables and device variables, and create the axes (to create the Axis Variables).
2. Create the programs to check.
3. Set up the tasks and build the project.
4. Start the Simulator in Execution Time Estimation Mode.
5. With an NJ-series CPU Unit, set the *Expanded number of I/O points* in the CJ-series Unit parameters in the Task Execution Time Monitor to create user-defined variables for specified CJ-series Special Units. Also set the sizes of the expansion areas (e.g., fixed I/O allocation areas for the DeviceNet Unit) for AT specifications (i.e., the number of output words and the number of input words). These sizes are used to calculate the I/O refresh time for the specific Special Units.
6. Estimate the task execution times in the Task Execution Time Monitor.

You can check the following values in the Task Execution Time Monitor when you start the Simulator in Execution Time Estimation Mode.

- The average and maximum values of the task execution time
- The average and maximum values of the real processing times of the tasks
- Bar graph that shows the system common processing time, I/O refreshing time, user program execution time, and motion control time
- CPU usage

Note Only the primary periodic task is displayed for an NX-series CPU Unit.

Additional Information

You can check the following values when connected to the Simulator of the Sysmac Studio. You cannot check these values when connected to the physical Controller.

- CPU usage: Displays how much of the task period is used by the total of the maximum estimated task processing time, the tag data link service execution time ratio, and the system service processing time of an NJ-series CPU Unit (as specified in the system service monitoring settings). If CPU usage exceeds 100%, it means that there is not sufficient time for task processing and the system service monitoring settings.
  This information is not displayed for an NX-series CPU Unit.
- Real processing time of tasks: This is the time that was required for the task from when task execution is started until it is completed. The time to execute other tasks that were executed from when task execution was started until it was completed is not included.

● Calculating Times on the Physical Controller

You can check the following values in the Task Execution Time Monitor when you are connected to the physical Controller.

- The minimum, average, and maximum values of the task execution time
- Set period
- Number of times a task is executed (Task Execution Count)
- Number of times the task period was exceeded (Task Period Exceeded Count)
- Task execution time ratio

The maximum values that are displayed on the Sysmac Studio are the results of operation on the physical Controller. As described previously, the maximum value of the task execution time varies depending on the internal status of the physical Controller. As a result, the maximum values obtained here may be exceeded in actual operation. Use the obtained values or the larger values in the following calculating results as guidelines of maximum values.

NX-series CPU Units

(Average value of task execution time + (Average value of task execution time − Minimum value of task execution time)) × 1.2 + 25 µs

NJ-series CPU Units

- Task period of 500 µs: Average value of task execution time + (Average value of task execution time − Minimum value of task execution time) + 100 µs
- Task period of 1, 2, or 4 ms: Average value of task execution time + (Average value of task execution time − Minimum value of task execution time) + 120 µs

For NJ-series CPU Units, you can also check whether sufficient system service execution time is obtained.
Insufficient system service execution time may decrease the online operations of the Sysmac Studio or the communications response performance with external devices such as an HMI. For NJ-series CPU Units, system services are executed during the unused time between execution of the tasks.
Use the following guideline for the task execution time ratio.

CPU Unit model	Guideline for task execution time ratio
NJ501-□□□□	90% or less
NJ301-□□□□	80% or less
NJ101-□□□□	70% or less

Version Information

- The task execution time ratio is displayed when an NJ-series CPU Unit is used with Sysmac Studio version 1.12 or higher.
- Use an NJ101-□□□□ CPU Unit with Sysmac Studio version 1.13 or higher. You cannot use an NJ101-□□□□ CPU Unit with Sysmac Studio version 1.12 or lower.

Precautions for Correct Use

NX-series CPU Units are designed to always secure sufficient time for system service execution. Therefore, the task execution time ratio is not displayed.

Additional Information

The average values of the task execution times that are displayed for task execution time monitoring are the averages for 10 task execution times.

5-10-2 Examples of Task Design

This section provides the design procedure for a project that consists of only the primary periodic task. In any actual application or for specific conditions, you need to consider any elements for which the design procedure must be changed. This example is therefore for reference only.

1. Find the I/O response times that are required for the system from the equipment specifications.
2. From the system I/O response times, determine the task period for the primary periodic task.
3. See if the task execution time fits into the task period that you determined.

   Then, work on paper or use the Task Execution Time Monitor of the Sysmac Studio to estimate the average and maximum values of the task execution time.
4. For NJ-series CPU Units, see if the system service times are within the monitor settings.

   If you use the Sysmac Studio, check the CPU usage.
5. Use the physical Controller to see if the task execution time fits into the task period.

   Place the Sysmac Studio online with the physical Controller and use Task Execution Time Monitor to check the task execution times.

- **If only the primary periodic task is too large to fit within the specified task period, consider separating it into periodic tasks as follows.**

  - For NX-series CPU Units, among processes required for device control, assign those which require the high-speed control to the primary periodic task and other processes to the priority-5 periodic task.
  - To reduce the task execution time, use the Enable Program (PrgStart) and Disable Program (PrgStop) instructions to execute the program assigned to the primary periodic task only when necessary.
  - If the primary periodic task will still not fit within the specified task period after these measures, among the processes of primary periodic task, assign those which do not require the high-speed or high-accuracy control to the priority-16 periodic task.
    However, if variables such as the Axis Variables are accessed between the primary periodic task and the priority-16 periodic task, the task execution time for the primary periodic task may be longer. For details, refer to the *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507).
  - If the primary periodic task will still not fit within the specified task period even after you take all of these measures, change the task period for the primary periodic task.

- **If a task is separated, the periodic task will vary greatly with the unused time for primary periodic task execution.**

  For a periodic task, use twice the average and maximum values calculated for the task execution time to set the task period and then fine-tune the setting from there.

5-10-3 System Input and Output Response Times

The times that are required for the system to produce an output after it receives an input are described in this section.

The I/O response times depend on various conditions.

The input response times and output response times between external devices and the slaves and Units must be added to the system I/O response times.

Sequence Control with Basic I/O Units

I/O refreshing between Basic I/O Units and external devices is performed in the task to which I/O refreshing is assigned. The I/O response times that include EtherCAT communications times are given below.

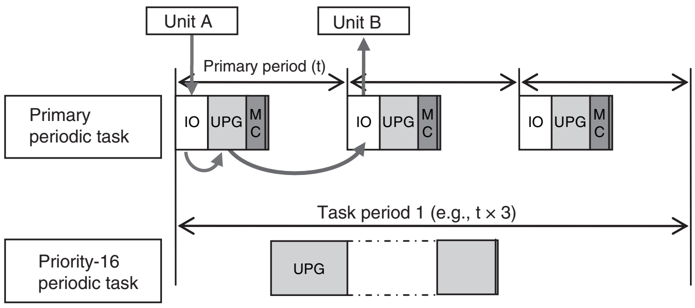
The I/O response times that include EtherCAT communications times are given below.

● Performing Control with the Programs in the Primary Periodic Task

The Controller makes a response in the following I/O response time.

Minimum I/O response time = Primary period

Example: Controlling Unit A and Unit B with the Primary Periodic Task



Note: The above diagram shows only one input and one output.

However, the I/O response time may be as follows depending on the timing of the input from the Unit.

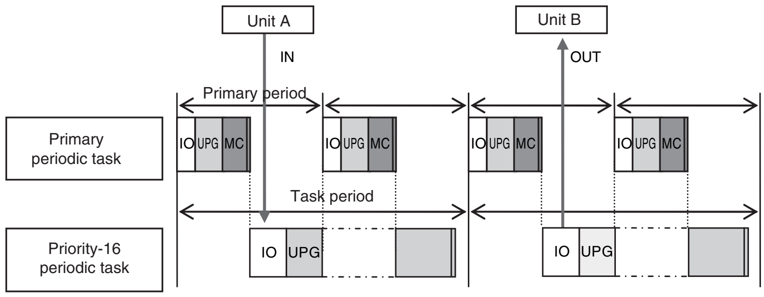
Maximum I/O response time = Primary task period × 2

● Performing Control with the Programs in the Priority-16 Periodic Task

The Controller makes a response in the following I/O response time.

Minimum I/O response time = Priority-16 periodic task period

Example: Controlling Unit A and Unit B with the Priority-16 Periodic Task



Note: The above diagram shows only one input and one output.

However, the I/O response time may be as follows depending on the timing of the input from the Unit.

Maximum I/O response time = Priority-16 periodic task period × 2

Sequence Control with EtherCAT Slaves

For EtherCAT slaves, EtherCAT communications with external devices is performed for I/O refreshing in the primary periodic task.

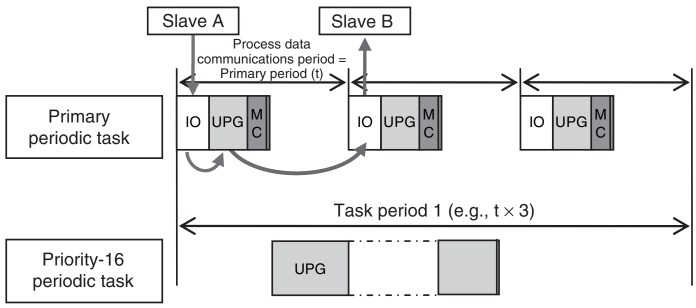
The I/O response times that include EtherCAT communications times are given below.

● Performing Control with the Programs in the Primary Periodic Task

The Controller makes a response in the following I/O response time.

Minimum I/O response time = Primary period (= process data communications cycle)

Example: Controlling EtherCAT Input Slave A and EtherCAT Output Slave B with the Primary Periodic Task



Note: The above diagram shows only one input and one output.

However, the I/O response time may be as follows depending on the timing of the input from the slave.

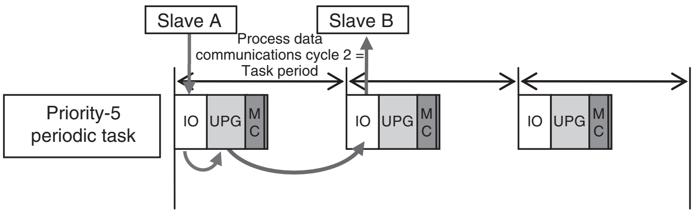
Maximum I/O response time = Primary period (= process data communications cycle) × 2

● Performing Control with the Programs in the Priority-5 Periodic Task

When you perform control with the user program in the priority-5 periodic task, the NX-series CPU Unit makes a response in the following I/O response time.

Maximum I/O response time = Priority-5 periodic task period (= process data communications cycle 2)

Example: Controlling EtherCAT Input Slave A and EtherCAT Output Slave B with the Priority-5 Periodic Task



Note: The above diagram shows only one input and one output.

However, the I/O response time may be as follows depending on the timing of the input from the slave.

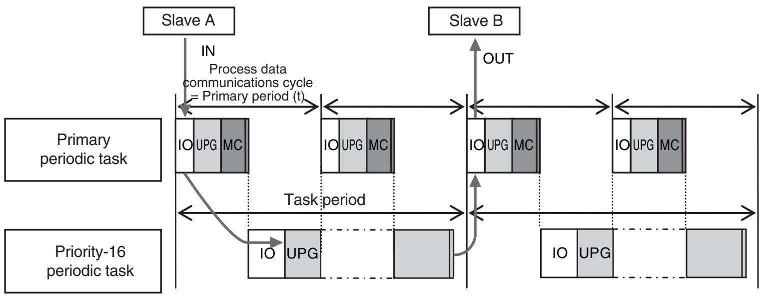
Maximum I/O response time = Priority-5 periodic task period (= process data communications cycle 2) × 2

● Performing Control with the Programs in the Priority-16 Periodic Task

The Controller makes a response in the following I/O response time.

I/O response time = Priority-16 periodic task period

Example: Controlling EtherCAT Input Slave A and EtherCAT Output Slave B with the Priority-16 Periodic Task



Note: The above diagram shows only one input and one output.

However, the I/O response time may be as follows depending on the timing of the input from the slave.

Maximum I/O response time = Priority-16 periodic task period × 2

Performing Motion Control with Motion Control Instructions

Motion control instructions access the Servo Drive and encoder input slaves to which axes are assigned.

For NJ-series CPU Units, motion control instructions can be used in the primary periodic task and in a priority-16 periodic task.

In either case, the motion control instructions are processed in the motion control processing (MC) section of the primary periodic task.

For NX-series CPU Units, motion control instructions can be used in the primary periodic task, in the priority-16 periodic task, and in the priority-5 periodic task.

The motion control instructions included in the primary periodic task and in a priority-16 periodic task are executed in the motion control processing (MC) of the primary periodic task.

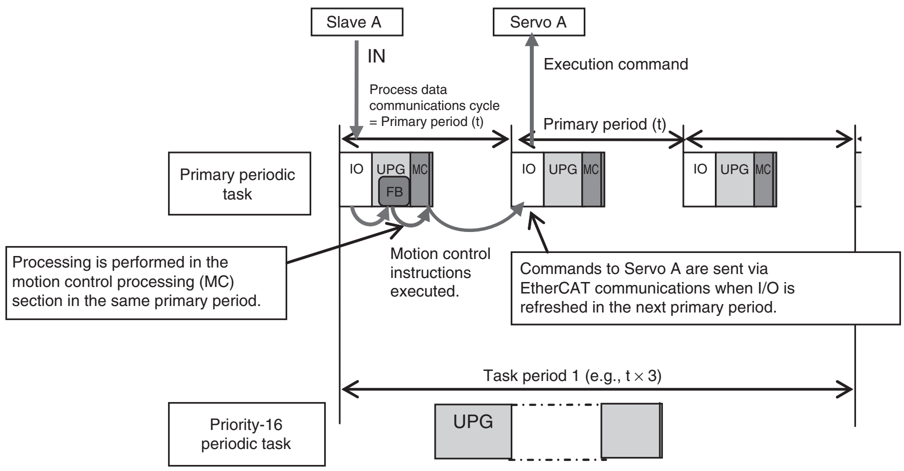
The motion control instructions included in a priority-5 periodic task are executed in the motion control processing (MC) of the priority-5 periodic task.

The I/O response times that include EtherCAT communications times are given below.

● Programming Motion Control Instructions in the Primary Periodic Task

The motion control instructions are processed in the next motion control processing (MC) section of the primary periodic task. The results of processing are output via EtherCAT communications to the Servo Drive to which the axis is assigned during the I/O refresh period in the next primary periodic task.
The Controller makes a response in the following I/O response time.

I/O response time = Primary period (= process data communications cycle)



Note: The above diagram shows only one input and one output.

However, the I/O response time may be as follows depending on the timing of the input from the slave.

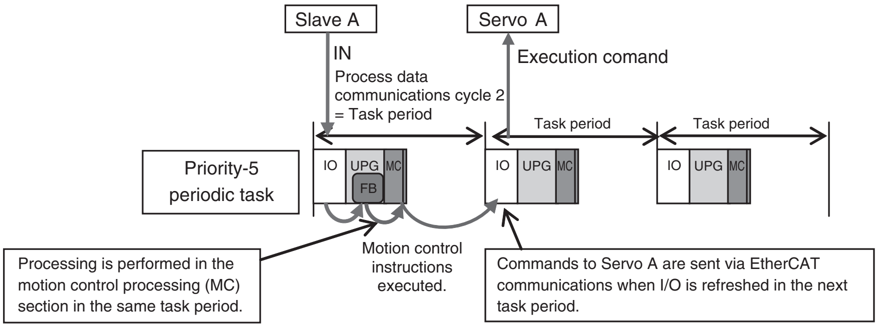
Maximum I/O response time = Primary period (= process data communications cycle) × 2

● Programming Motion Control Instructions in the Priority-5 Periodic Task

The motion control instructions are processed in the immediate motion control processing (MC) section in the priority-5 periodic task. The results of processing are output via EtherCAT communications to the Servo Drive to which the axis is assigned during the I/O refresh period in the next priority-5 periodic task.

The Controller makes a response in the following I/O response time.

I/O response time = Priority-5 periodic task period (= process data communications cycle)



Note: The above diagram shows only one input and one output.

However, the I/O response time may be as follows depending on the timing of the input from the slave.

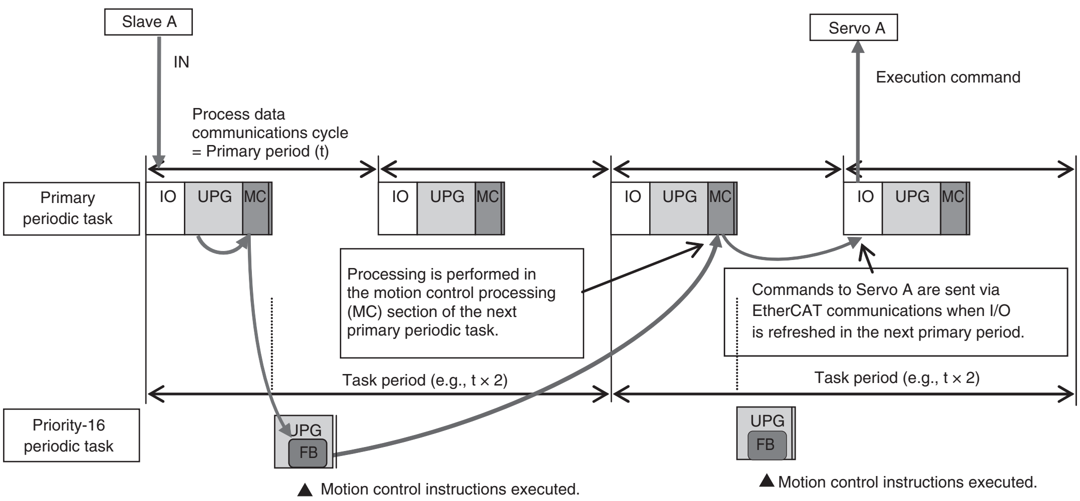
Maximum I/O response time = Priority-5 periodic task period (= process data communications cycle 2) × 2

● Programming Motion Control Instructions in the Priority-16 Periodic Task

The motion control instructions are processed in the next motion control processing (MC) section of the primary periodic task after the priority-16 periodic task. The results of processing are output via EtherCAT communications to the Servo Drive to which the axis is assigned during the I/O refresh period in the next primary periodic task.

The Controller responds in the following I/O response time regardless of the execution timing of the motion control instructions.

Minimum I/O response time = Priority-16 periodic task period + Primary period (= process data communications cycle)



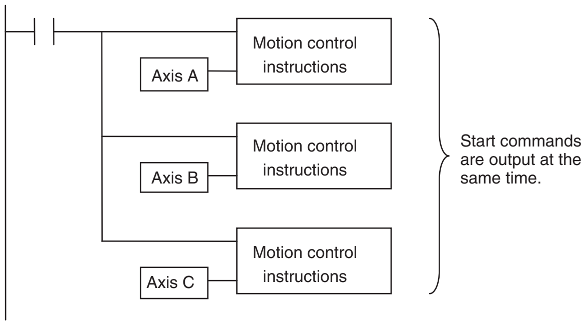
Note: The above diagram shows only one input and one output.

However, the response time may be as follows depending on the timing of the input from the slave.

Maximum I/O response time = Priority-16 periodic task period + Primary period (= process data communications cycle) × 2

● Simultaneous Execution of More Than One Axis

If more than one axis is controlled in the same task period by the programs in the same task, they can be started at the same time.



Additional Information

You can access the values of Axis Variables in the tasks other than those for axis control.

For detailed usage and precautions, refer to the *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

6

Programming

This section describes programming, including the programming languages, and the variables and instructions that are used in programming.

6-1 Overview of Programming Procedures

This section provides an overview of programming procedures.

The shaded steps in the overall procedure that is shown below are related to programming.

Step 1. Software Design

- Step 1-1 Designing I/O and Processing
- Step 1-2 Designing Tasks
- Step 1-3 Designing Programs

Step 2. Software Setups and Programming

- Step 2-1 Slave and Unit Configurations
- Step 2-2 Controller Setup
- Step 2-3 Programming
- Step 2-4 Offline Debugging

Step 3. Mounting and Setting Hardware

Step 4. Wiring

Step 5. Checking Operation and Starting Operation on the Actual System

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series Controller* for details.

POU (Program Organization Unit) Design	Reference
● **Determine which processes to put into which POUs and design the POUs.** **Note** Functions cannot contain function block instructions or function blocks.	*6-2 POUs (Program Organization Units)*
● **Determine which languages, such as ladder diagrams, inline ST, and ST, to use to create each process.** **Note** Inline ST is structured text that is written as an element of a ladder diagram.	*6-5 Programming Languages*

Variable Design	Reference
● **Design the user-defined variables that you need to create.**	*6-3-1 Variables* *6-3-2 Types of Variables*
● **Separate variables into those that you use in more than one POU (global variables) and variables that you use in only specific POUs (local variables).**	*6-3-3 Types of User-defined Variables in Respect to POUs*
● **Determine if you need to automatically generate the variable names for the device variables that you use to access slaves and Units or if you need to define them yourself.**	*3-3 I/O Ports and Device Variables*
● **Design the attributes for the variables.** Variable Name, Data Type, AT Specification, Initial Value, Retain, Constant, and Network Publish Decide the data types of your variables (including array specifications, range specifications, structures, and enumerations).	*6-3-4 Attributes of Variables* *6-3-5 Data Types* *6-3-6 Derivative Data Types*
● **Keep the following precautions in mind when you design variables.** • Retention: Set the Retain attributes to determine the values that are used for variables when the power supply is turned ON or when the operating mode changes. • Structures: When a structure is used for a variable in an instruction, design the program to use the same structure data type for the input parameter, output parameter, or in-out parameter. Example: Communications Instructions • Array Specifications: When an array variable is used for the variable for an instruction, design the program to use an array variable for the input parameter, output parameter, or in-out parameter. Examples: Shift Instructions, Stack Instructions, and Table Instructions • AT Specifications: Use AT specifications for the variables used for input parameters to certain instructions. Example: Fixed or user I/O allocations for DeviceNet Units • Network Publishing: Design the variables for EtherNet/IP tag data links.	*6-3-4 Attributes of Variables* *6-3-5 Data Types* *6-3-6 Derivative Data Types*

6-2 POUs (Program Organization Units)

The user program that runs on an NJ/NX-series CPU Unit is made from a combination of POUs (program organization units).
This section describes the configuration and specifications of POUs.
Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details on creating POUs in the Sysmac Studio.

6-2-1 What Are POUs?

A POU (program organization unit) is a unit that is defined in the IEC 61131-3 user program execution model. A POU includes a local variable table and an algorithm (i.e., a series of code or logic). It is the basic unit used to build the user program.
You combine POUs to build a complete user program.

There are three types of POUs, as described below.

- Programs
  A program corresponds to a main routine. It is the main type of POU that is used for algorithms.
  You can place any instruction, function, or function block in the algorithm of a program.
- Function Blocks (FBs)
  A function block can output different values even with the same inputs. Function blocks are executed when they are called from a program or another function block.
- Functions (FUNs)
  A function always outputs the same values for the same inputs. Functions are executed when they are called from a program, another function, or a function block.

The POUs consists of a combination of these three types of POUs. You can create many POUs.
You assign the programs to tasks to execute them.

6-2-2 Overview of the Three Types of POUs

Programs

Executing Programs and Execution Conditions

- You execute a task to execute the programs that are assigned to that task.
- Programs are always executed.

Notation

- The POUs must include at least one program. You can assign up to 128 programs to a single task.

Function Blocks (FBs)

Executing Function Blocks and Execution Conditions

- You can call function blocks from programs or other function blocks to execute them.
- Function blocks are always executed.
- If you want a function block to execute only when a condition is met, you must define an input variable that sets the execution condition.

Notation

- You can use any instruction, user-defined function, or user-defined function block in the algorithm of a function block.
- You can retain the values of internal variables. Therefore, you can retain status, such as for timers and counters.
- There are both user-defined and system-defined function blocks.
  User-defined function blocks are called user-defined function blocks. System-defined function blocks are sometimes called FB instructions.

For details on function blocks, refer to *6-2-5 Details on Function Blocks*.

Functions

Executing Functions and Execution Conditions

- You can call functions from programs, other functions, or function blocks to execute them.
- The *EN* input variable specifies the execution condition. A function is executed only once each time *EN* changes to TRUE.

Notation

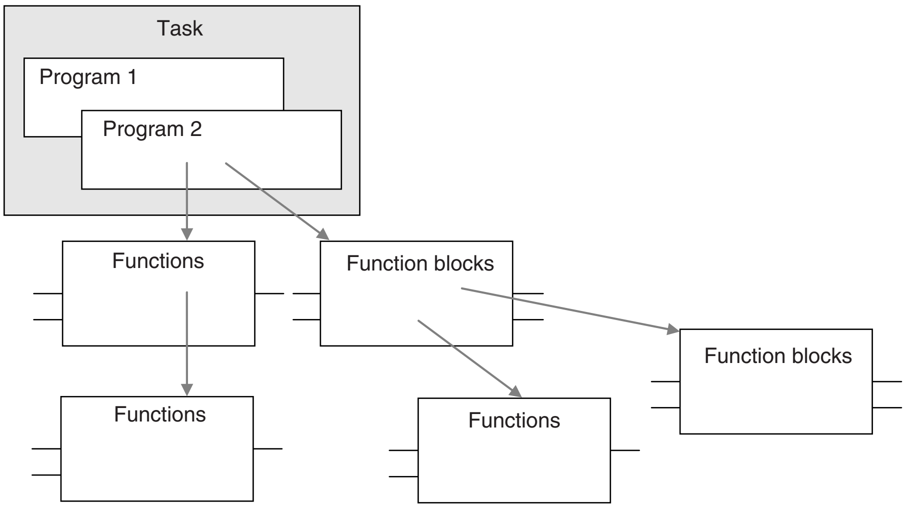
- You cannot use FB instructions or user-defined function blocks in algorithms.
- The values of internal variables are not retained. Therefore, the output value remains constant.
- There are both user-defined and system-defined function blocks.
  User-defined functions are called user-defined functions. System-defined functions are sometimes called FUN instructions.

For details on functions, refer to *6-2-6 Details on Functions*.

6-2-3 Differences between Programs, Functions, and Function Blocks

Item	POU type	Programs	Function blocks	Functions
Execution method		Executed upon execution of assigned task.	Called from a program or another function block.	Called from a program, function, or function block.
Algorithm	**Any instructions**	Supported.	Supported.	Not supported.
	User-defined functions	Supported.	Supported.	Supported.
	User-defined function blocks	Supported.	Supported.	Not supported.
Execution condition		Executed each period.	Executed each period. Specify the execution condition with an input variable.	Specify the execution condition with the EN input.

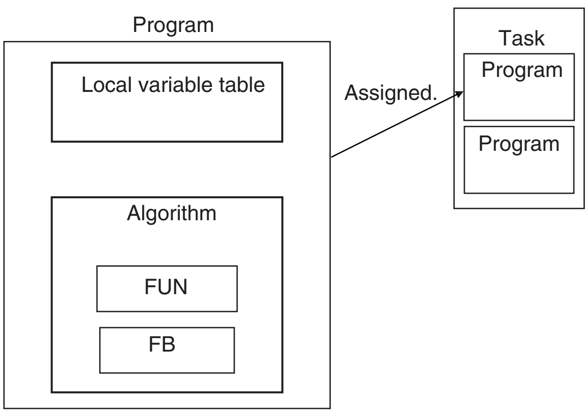
The hierarchical relationships between programs, functions, and function blocks are shown in the following figure.



6-2-4 Details on Programs

Program Structure

Programs consist of a local variable table and an algorithm.
You can use any function or function block in the algorithm of a program.



You cannot call programs from other POUs.

Program Execution Conditions

Programs are executed when the task they are assigned to is executed.

Order of Execution

You can set the order of execution of all programs in a task. You specify this order under **Task Settings - Program Assignment Settings** in the Sysmac Studio.

Related System-defined Variables

All programs have the following system-defined variables in the local variables.

Variable name	Meaning	Function	Data type	Read/write
P_First_RunMode	First RUN Period Flag	This flag is TRUE for only one task period after the operating mode of the CPU Unit is changed from PROGRAM mode to RUN mode if execution of the program is in progress. This flag remains FALSE if execution of the program is not in progress. Use this flag to perform initial processing when the CPU Unit begins operation.	BOOL	Read
P_First_Run*1	First Program Period Flag	This flag is TRUE for one task period after execution of the program starts.*2 Use this flag to perform initial processing when execution of a program starts.	BOOL	R
P_PRGER	Instruction Error Flag	This flag changes to and remains TRUE when an instruction error occurs in the program or in a function/function block called from the program. After this flag changes to TRUE, it stays TRUE until the user program changes it back to FALSE.	BOOL	Read/write
P_CY	Carry Flag	This flag is updated by some instructions.	BOOL	Read

*1 A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required.

*2 To enable or disable the program, use the PrgStart or PrgStop instruction. You can make setting for the PrgStart instruction so that it executes the program without changing *P_First_Run* to TRUE.

6-2-5 Details on Function Blocks

Procedure to Create Function Blocks

A function block consists of a function block definition that is made in advance and instances that are used in the actual programs. Create function blocks in the following order.

1. Create the function block definition.

   Create the algorithm.

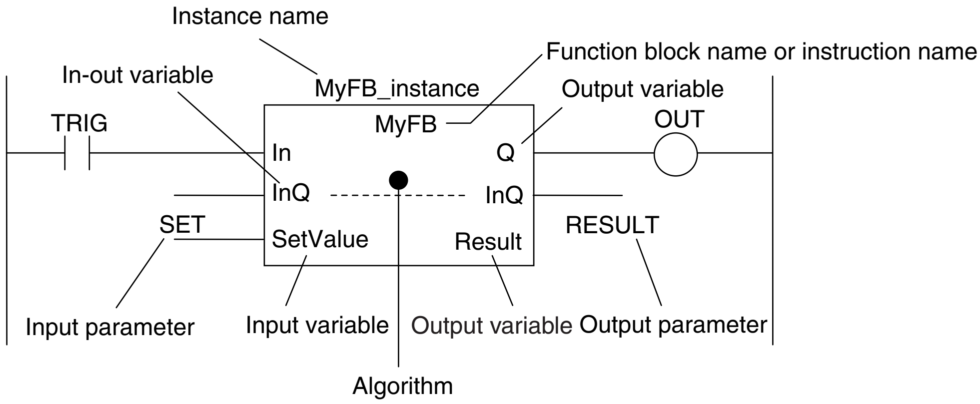
2. Placing an Instance of the Function Block Definition in a Program

   Call the function block definition from a program or another function block.
   You can call the same function block definition from more than one program or function block.
   After you place an instance of a function block definition in a program or in another function block, you can manipulate and execute it as an independent entity.

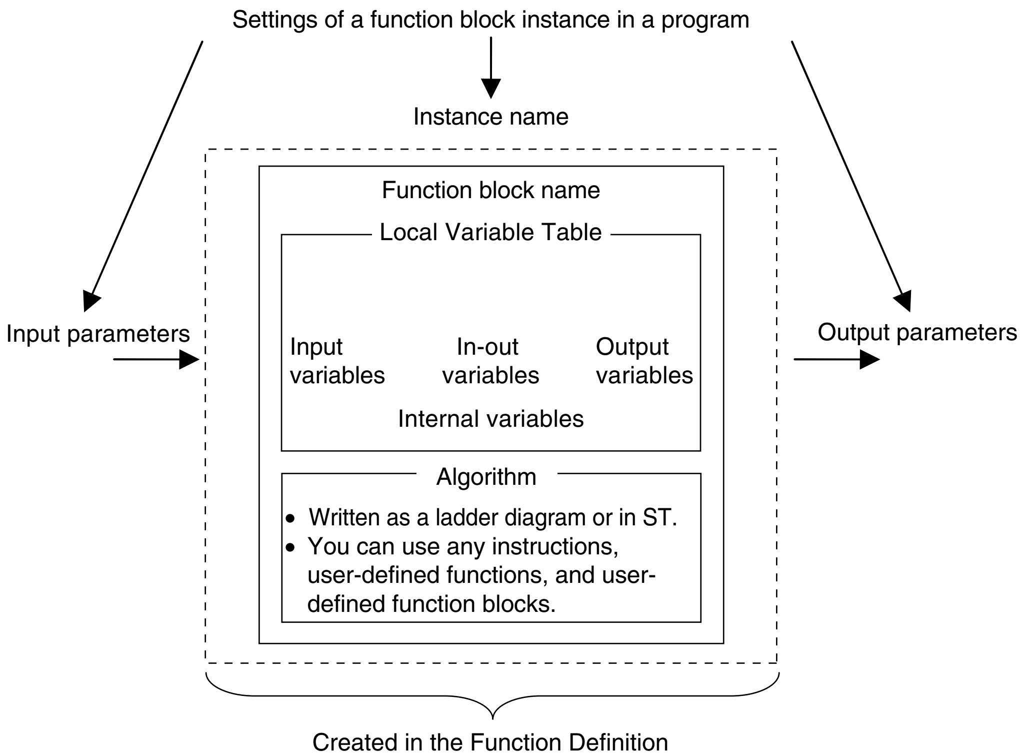
Structure of Function Blocks

In a ladder diagram, function blocks are represented as rectangular boxes as shown below. Refer to *Calling Function Blocks from ST* on page 6-10 for details about how to express function blocks in ST. Function blocks consist of the following parts.

- Function Block in Ladder Diagram:



- Function Block Settings
  When you create an instance of a function block definition, make the following settings.



● Function Block Name or Instruction Name

This is the function block name or instruction name assigned in the function block definition when the function block is created.

● Instance Name

You give an instance name to a function block instance in a program to enable managing it. You specify an instance name when you call a function block definition from a program or another function block.

● Algorithm

You can code the algorithm either as a ladder diagram or in ST. You can use any instruction, user-defined function, or user-defined function block in the algorithm.

● Local Variable Table

The local variable table is used to define input variables, output variables, in-out variables, internal variables, and external variables.
Refer to *Variable Designations for Function Blocks* on page 6-11 for details.

● Parameters

Input Parameters to Input Variables

An input parameter passes a value to an input variable in a function block when function block execution begins. An input parameter can be either a variable or a constant.

Output Parameters from Output Variables

An output parameter receives a value from an output variable in a function block when function block execution is completed. A variable is given as the parameter.

In-Out Parameters Shared between In-Out Variables

The value of the in-out parameter changes within the function block. The same variable is used for both the input and output.

Additional Information

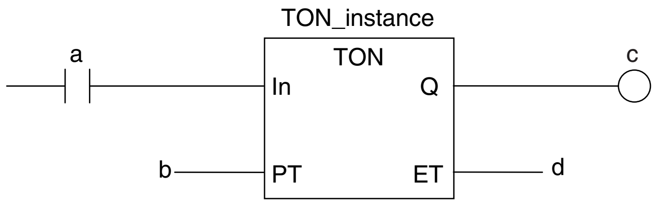
You can omit input and output parameters. Refer to information on operation when parameters are omitted in *Operation When Parameters Are Omitted* on page 6-23 for details.

Calling Function Blocks from ST

The following example shows how to call function blocks from ST.

instance_name(input_variable_1:=input_parameter_1, ... input_variable_N:=input_parameter_N,in-out_variable_1:=in-out_parameter_1, ... in-out_variable_N:=in-out_parameter_N,output_variable_1=>output_parameter_1, ... output_variable_N=>output_parameter_N);

You can also omit input variable names and other variable names, and give only the parameters. (If you do, the parameters must be given in the order that they are given in the function block definition.) Also, the number of parameters must match the number of input variables and other variables in the function block definition.



Function Blocks Expressed in ST:

Instance name

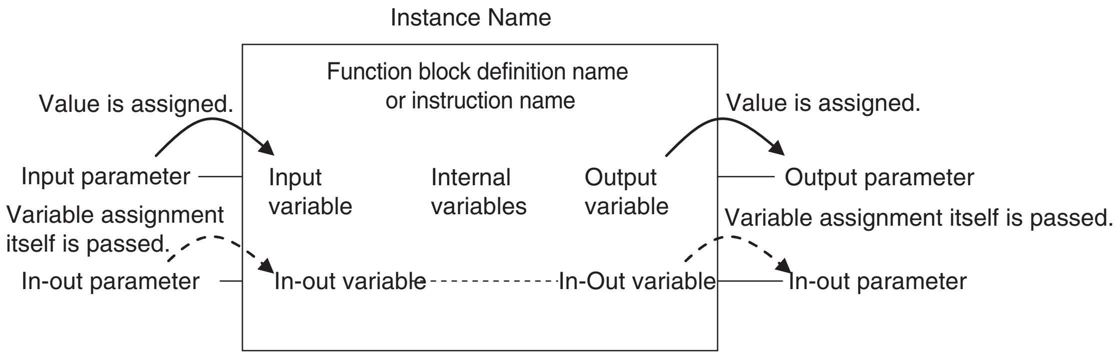
TON_instance(In:=a, PT:=b, Q=>c, ET=>d);

TON_instance(In:=a, PT:=b, Q=>c); (*The *ET* output is omitted here.*)

TON_instance(a,b,c,d); (*Input and output variables are omitted here.*)

Refer to *Function Block Calls* in *ST Language Statement* on page 6-104 for details.

Variable Designations for Function Blocks



The specifications for variables in function blocks are given below.

Variables	Number	Specification
Input variables[*1]	1 to 64	Input variables are used as input arguments within the function block. They cannot be changed inside the function block.
		• When the function block is executed, the input variables are set to the values of the input parameters. • You can specify either constants or variables for input parameters. • Omitting Input Parameters: Refer to information on operation when parameters are omitted in *Operation When Parameters Are Omitted* on page 6-23. • You can specify to detect when the variable changes to TRUE or changes to FALSE. • You can access and change the values from outside the function block. Access these values using the following format: *InstanceName.InputVariableName*.[*2]
Output variables[*3]	1 to 64	Output variables are used as output arguments from the function block.
		• The output parameters are set to the values of the output variables at the end of function block execution. • You cannot specify a constant or a variable with constant attribute for an output parameter. • You can omit output parameter connections. If you omit an output parameter, the value of the output variable is not assigned to any parameter. • You can access the values of output variables from outside of the function block. Access these values with the following format: *InstanceName.OutputVariableName*. However, you cannot write values directly to an output variable.
In-out variables	0 to 64	In-out variables are used as inputs to and outputs from the function block. They can be changed inside the function block.
		• The value of an in-out parameter is passed to an in-out variable and the value of the in-out variable is then passed to the in-out parameter. • You cannot specify a constant or a variable with constant attribute for an in-out parameter. • If you change the value of an in-out variable within a function block, the value of the in-out parameter changes at that time. • You cannot omit in-out parameters.
Internal variables	No limit	Internal variables are used for temporary storage within a function block.
		• The values of internal variables are retained regardless of whether the function block is executed. • Internal variables can have Retain attributes. • You cannot access the values of internal variables from outside of the function block.

Variables	Number	Specification
External variables	No limit	External variables are used to access global variables.
EN	0	An *EN* variable cannot be used in a function block. (This applies to both user-defined function blocks and FB instructions.)
ENO	0 or 1	Generally, this is a BOOL output variable that is set to TRUE for a normal end, and to FALSE for an error end.
		• You can also omit it for some FB instructions. • Refer to *ENO* on page 6-20 for details.

*1 In the Sysmac Studio version 1.01 or lower, at least one BOOL input variable is required when you use function blocks in a ladder diagram or in ST.

*2 In the Sysmac Studio version 1.07 or lower, it is impossible to change the value of an input variable from outside the function block. However, accessing it from outside the function block is possible.

*3 In the Sysmac Studio version 1.01 or lower, at least one BOOL output variable (including ENO) is required when you use function blocks in a ladder diagram or in ST.
In the Sysmac Studio version 1.02 or higher, at least one BOOL output variable (including ENO) is required only when you use function blocks in a ladder diagram.

Refer to *6-3-4 Attributes of Variables* for details on setting variable attributes.

Additional Information

If you define an external variable with the same name as a global variable in a function block, it is defined automatically based on that global variable.

● ENO

- When *ENO* is FALSE, the previous values of all other output variables are retained.

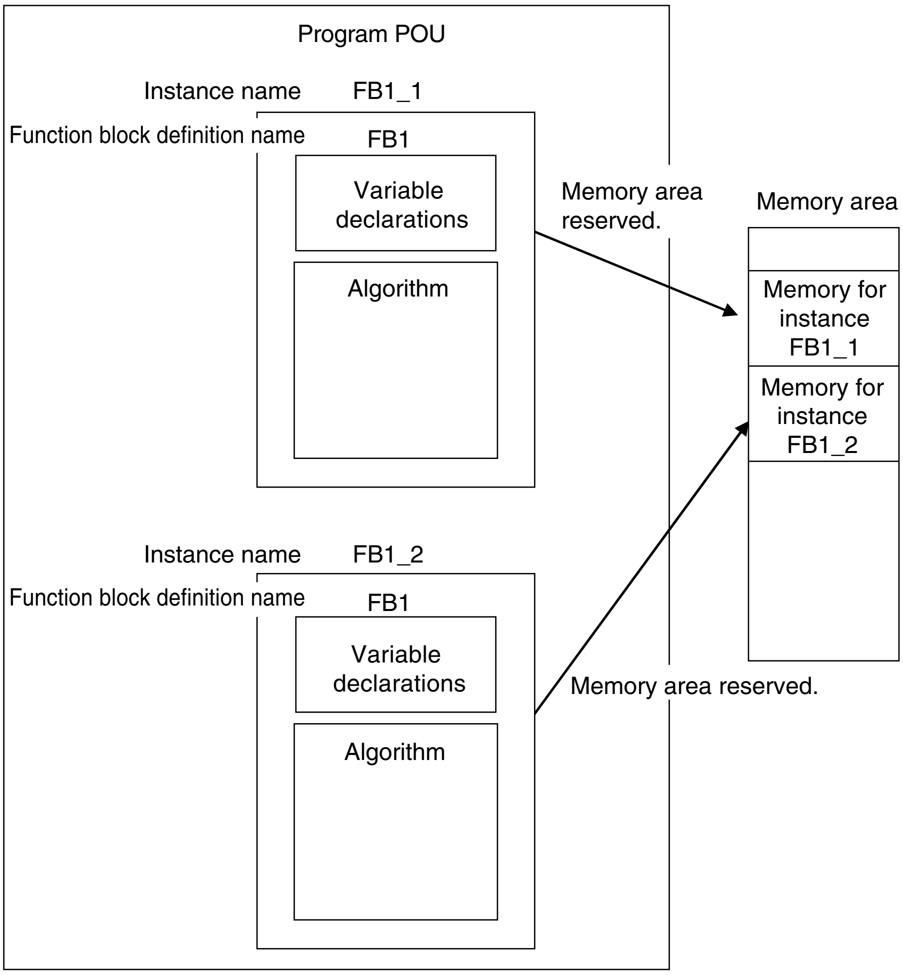
Function Block Definitions and Instances

A function block consists of a function block definition that is made in advance and instances that are then used in the actual programs. All instances of a function block are based on the function block definition.
A function block definition consists of an algorithm and a local variable table.

● Function Block Instance

When you place an instance of a function block definition in a program or another function block, the function block definition is treated as a part of that program or function block.
Function block definitions that are called from a program or another function block are called instances.
Every instance of a function block has an identifier known as an instance name associated with it, and every instance uses memory.

You can create instances of a function block definition to process different I/O data in the same way.

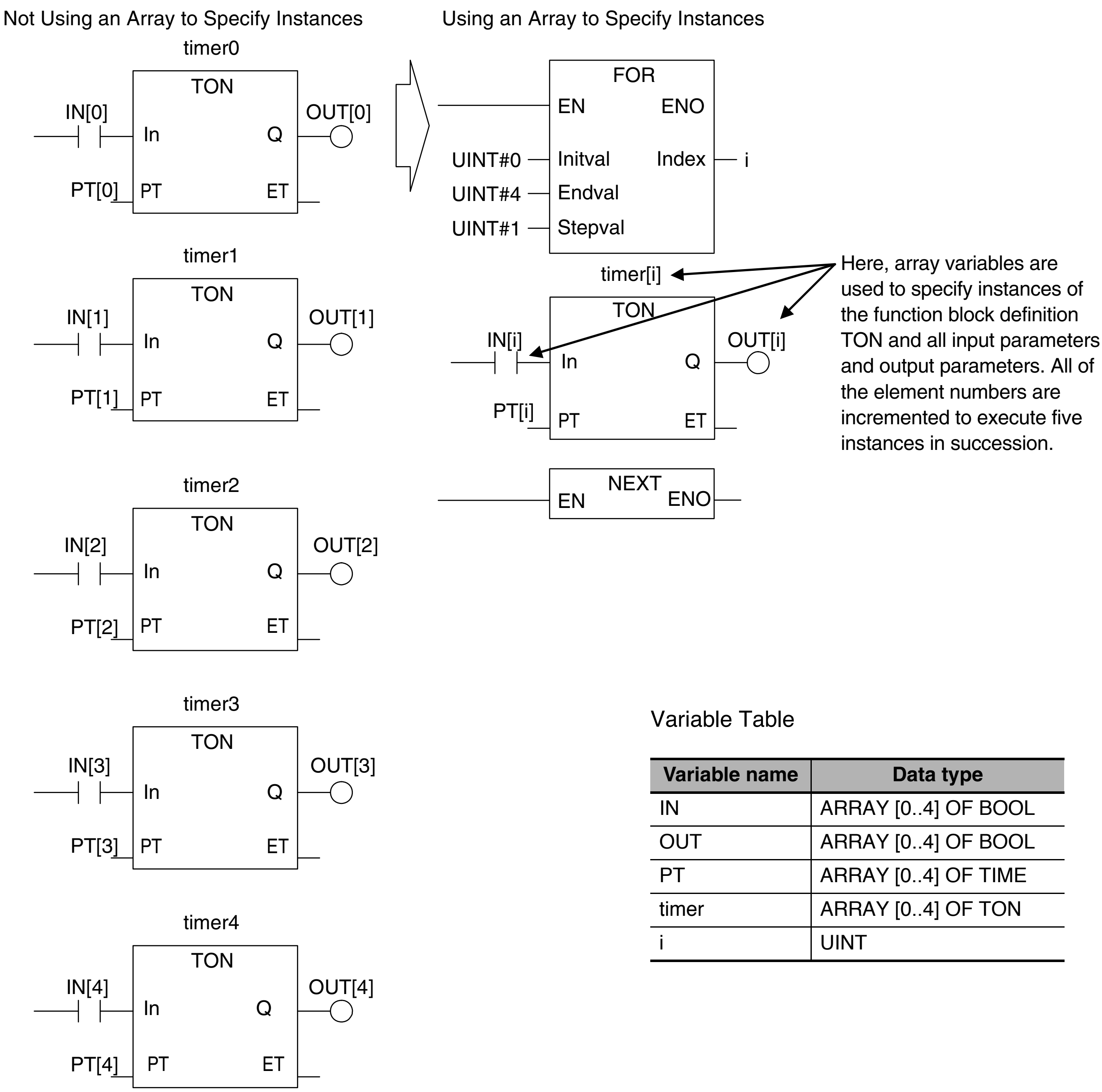


Instances cannot be read from other programs or function blocks. If an instance with the same name as another instance is placed in a different program or another function block, that instance will operate as a completely separate instance.

Array Specifications for Instances

Array specifications can be made for instances. You can indirectly specify an array element number with a variable to execute multiple instances with one instance name. Furthermore, you can switch input sources and output destinations and effectively execute multiple instances with a single instance name if you use an array specification for the input parameter and output parameter and specify the element numbers with the same variable.

Example:



Variable Table

Variable name	Data type
IN	ARRAY [0..4] OF BOOL
OUT	ARRAY [0..4] OF BOOL
PT	ARRAY [0..4] OF TIME
timer	ARRAY [0..4] OF TON
i	UINT

Execution Conditions for Function Blocks

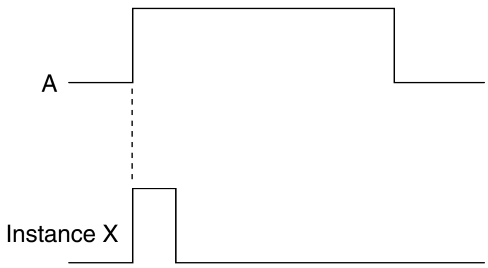
Function blocks do not have an EN input like functions. They are executed each period.

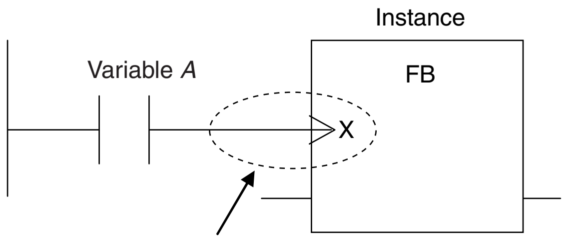
Processes That Require Constant Data Monitoring

Case	Algorithm in FB		ENO	Operations other than ENO
Normal operation	Executed.	Normal end	TRUE	Output parameters: Values are updated according to the internal algorithm. In-out parameters: Values are updated according to the internal algorithm.
		Error end	FALSE	Output parameters: Retained In-out parameters: Values are updated according to the internal algorithm.
Inside a master control region	Executed when the state of the power flow input is FALSE.		User-specified	One of the above, depending on the value of ENO.

Refer to *6-5-2 Ladder Diagram Language* for details on power flow output and parameter output.

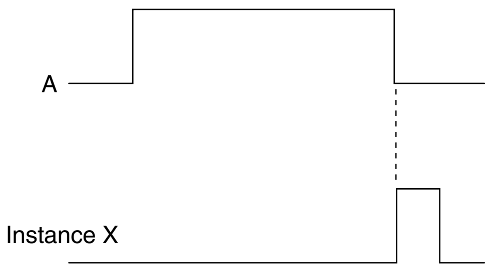
You can specify the edge for an input variable to make the variable TRUE only when the input parameter changes to TRUE.

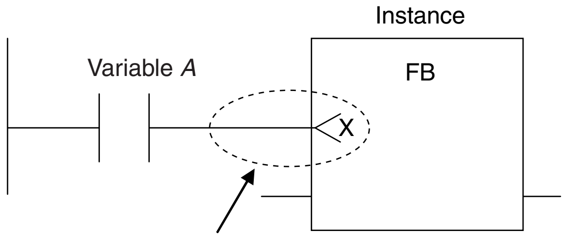




Specifies detecting a change to TRUE.

You can specify falling edges too.





Specifies detecting a change to FALSE.

Accessing Variables in a Function Block from Outside the Function Block

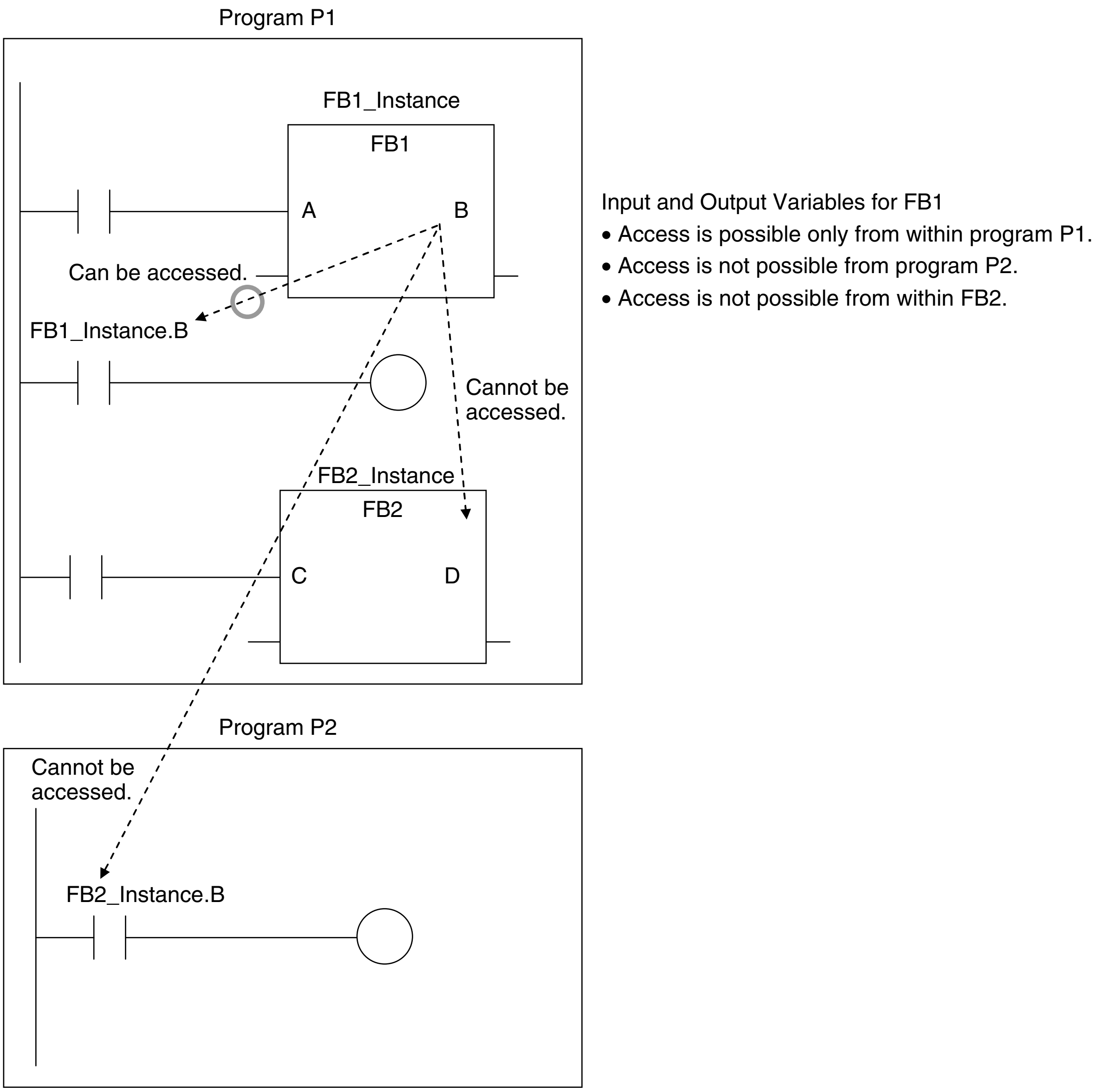
You can access the input and output variables of a function block from outside the function block. Variables are written as follows:

InstanceName.VariableName

Example: To Access Output Variable B of Function Block Instance *FB1_Instance*

FB1_Instance.B

You can access the input and output variables for a function block only within the program that contains the function block. However, you cannot access these variables from within other function block instances even if they are in the same program. You cannot access them from other programs.



The following variables cannot be accessed from external devices. If these variables are accessed, a building error will occur.

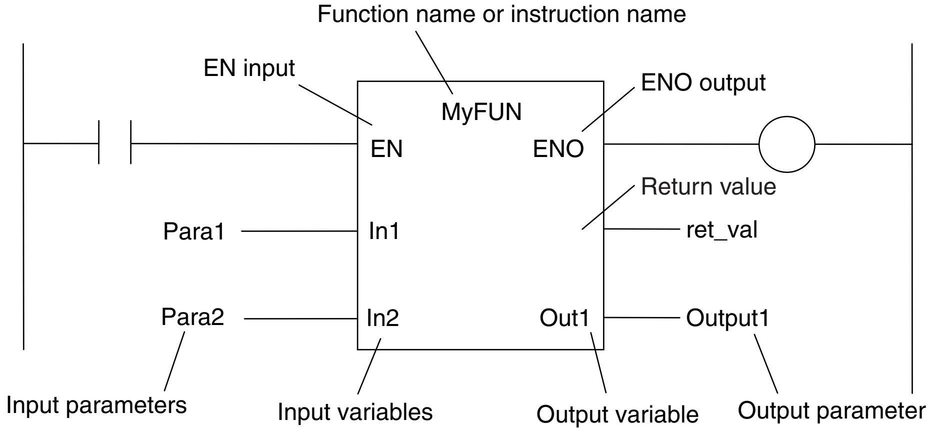
- In-out variables for function blocks
- Input variables for FB instructions for which the default value is not applied if an input parameter is omitted

6-2-6 Details on Functions

Structure of Functions

In a ladder diagram, functions are represented as rectangular boxes as shown below. Refer to *Expressing Functions in ST* on page 6-18 for details about how to express functions in ST. A function consists of the following parts.

Function in Ladder Diagram:



● Function Name or Instruction Name

This is the function name or instruction name assigned in the function definition when the function is defined.

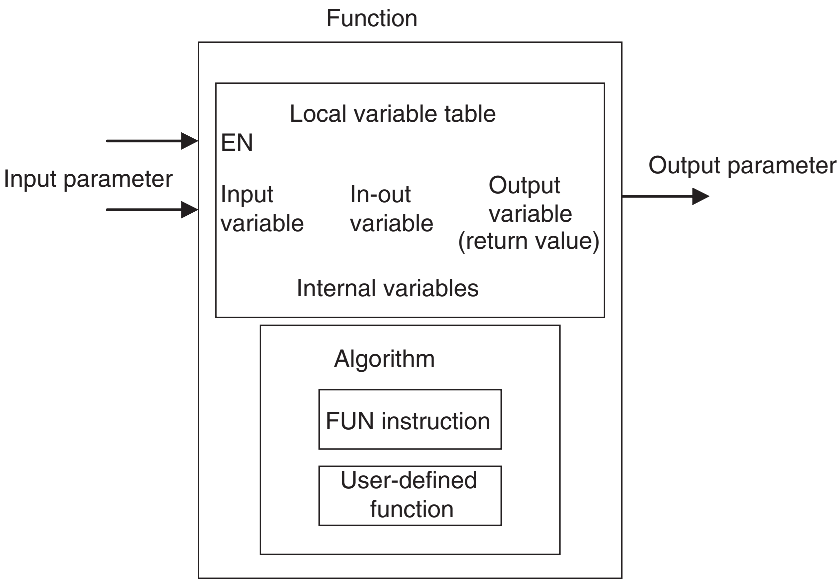
● Instance Name

Functions do not have instance names.

● Algorithm

You can code the algorithm either as a ladder diagram or in ST. You can use function instructions or user-defined functions in the algorithm of a function. You cannot use any FB instructions or user-defined function blocks. You also cannot use a differentiated instruction (e.g., R_TRIG or UP).

You cannot use the *P_First_RunMode* and *P_First_Run* system-defined variables.



- **Local Variable Table**

A local variable table defines the input variables, output variables, in-out variables, internal variables, and external variables.

Refer to *Variable Designations for Functions* on page 6-19 for details.

- **Parameters**

Input Parameters to Input Variables

An input parameter passes a value to an input variable in a function when function execution begins. An input parameter can be either a variable or a constant.

Output Parameters from Output Variables

An output parameter receives a value from an output variable in a function when function execution is completed. A variable is given as the parameter.

In-Out Parameters Shared between In-Out Variables

The value of the in-out parameter changes within the function. The same variable is used for both the input and output.

Expressing Functions in ST

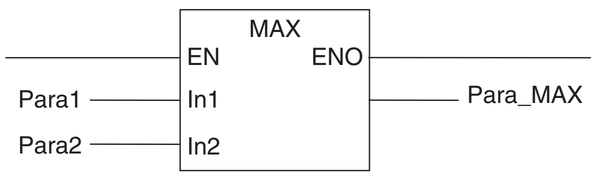
The following example shows how to call functions from ST.

return_value:=function_name (input_variable_1:=input_parameter_1, ... input_variable_N:=input_parameter_N,in-out_variable_1:=in-out_parameter_1, ... in-out_variable_N:=in-out_parameter_N,output_variable_1=>output_parameter_1, ... output_variable_N=>output_parameter_N);

However, you can also omit the return value.

You can also omit input variable names and other variable names, and give only the parameters. (If you do, the parameters must be given in the order that they are given in the function definition.) Also, the number of parameters must match the number of input variables and other variables in the function definition.

Functions Expressed in ST:

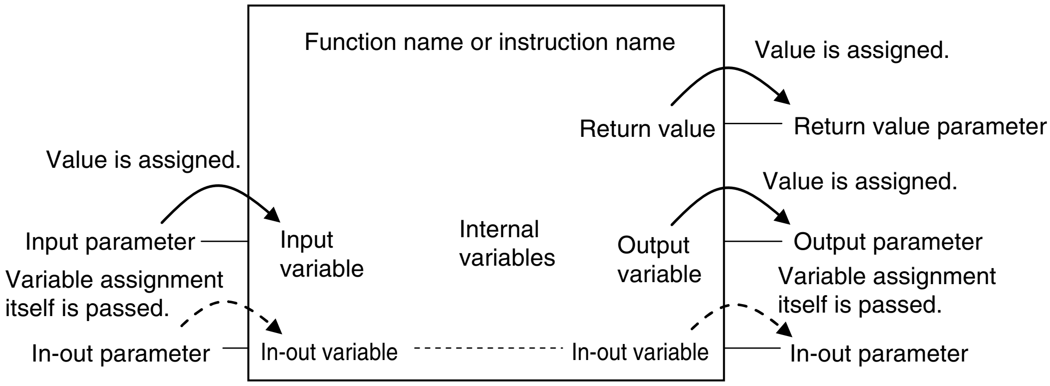


Function name

```
Para_MAX := MAX(In1:=Para1, In2:=Para2);
Para_MAX := MAX(Para1, Para2);          (*The input variables are omitted here.*)
```

Refer to *Function Calls* on page 6-107 for details.

Variable Designations for Functions



The specifications for variables in functions are given below.

Variables	Number	Specification
Input variables	0 to 64	Input variables are used as input arguments within the function. They cannot be changed inside the function.
		• When the function is executed, the input variables are set to the values of the input parameters. • You can specify either constants or variables for input parameters. • Omitting Input Parameters: Refer to information on operation when parameters are omitted in *Operation When Parameters Are Omitted* on page 6-23. • Unlike function blocks, you cannot specify to detect changes to TRUE or FALSE. • You cannot access the values of input variables from outside of the function. • Some of the instructions provided by OMRON can have varying numbers of input variables, but you cannot make a user-created function that has a varying number of input variables.
Output variables	0 to 64	Output variables are used as output arguments from the function.
		• The output parameters are set to the values of the output variables at the end of function execution. • You cannot specify a constant or a variable with constant attribute for an output parameter. • At least one BOOL output variable (including *ENO* and the return value) is required. • You can omit output parameter connections. If you omit an output parameter, the value of the output variable is not assigned to any parameter. • You cannot access the values of output variables from outside of the function.
In-out variables	0 to 64	In-out variables are used as inputs to and outputs from the function. They can be changed inside the function.
		• In-out parameters (variable designations) are directly passed to or received from the in-out variables. • You cannot specify a constant or a variable with constant attribute for an in-out parameter. • If you change the value of an in-out variable within a function, the value of the in-out parameter changes at that time. • You cannot omit in-out parameters. • You cannot access the values of in-out variables from outside of the function.
Internal variables	No limit	Internal variables are used for temporary storage within a function.
		• The value is not retained after execution is completed. • You cannot access the values of internal variables from outside of the function.

Variables	Number	Specification
External variables	No limit	External variables access global variables.
EN	1	This is a BOOL input variable used to execute the function.
		• The function is executed when *EN* is TRUE. • You must have one *EN* variable. (This applies to both user-defined functions and FUN instructions).
ENO	0 or 1	Generally, this is a BOOL output variable that is set to TRUE for a normal end, and to FALSE for an error end.
		• You can omit the *ENO* variable from user-defined functions. • Refer to *ENO* on page 6-20 for details.
Return value	1	The return value is the value that is returned to the calling instruction. It represents the results of the process after the algorithm in the function is executed.
		• Each function must have one return value. • You can specify enumerations of all basic data types. You cannot specify an array, structure, or union. • If you do not set the return values for algorithms in user-defined functions, the return values are not dependable. That is, the return values may be different even if the values of the input parameters are the same. Therefore, if you use the return value at the process that called the function, always set the return value in the algorithm in the function. • Refer to *Return Values*, below, for details.

Refer to *6-3-4 Attributes of Variables* for details on setting variable attributes.

Additional Information

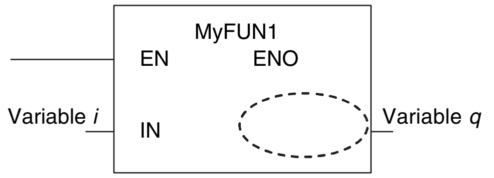
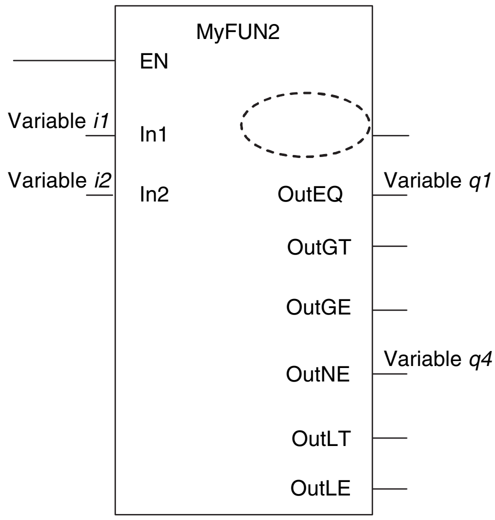
You can register global variables as external variables in a function variable table to access global variables. We recommend that you create your functions so that they produce output values uniquely based on their input parameter values. Algorithms that access global variables and use them to affect the output values are not recommended. When you check the program on the Sysmac Studio, a message will appear that says that it is not recommended to use global variables in functions. Take appropriate measures if necessary.

● ENO

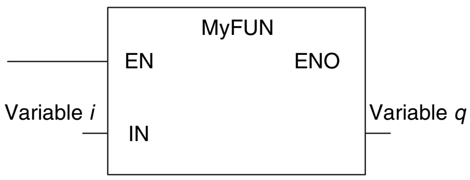
- When *ENO* is FALSE, the previous values of all other output variables are retained.

● Return Values

- Return values are blank in ladder diagrams.

Case	Ladder diagram notation	ST language notation
Using return values		*variable_q*:= MyFUN1(*variable_i*);
Not using a return value		MyFUN2(In1:=*variable_i1*,In2:=*v ariable_i2*, OutEQ=>*variable_q1*, OutNE=>*variable_q4*);

- The calling instruction is not required to use the return value in either a ladder diagram or ST.
- If you set the return value within a function algorithm, set the value to a variable with the same name as the function.
  For example, the return value of a function called *MyFUN* is *MyFUN*.



Definition of MyFUN

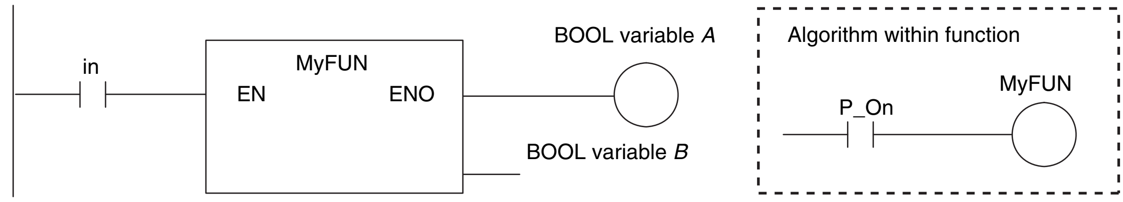
```
IF IN>100
   MyFUN:=100;
ELSE
   MyFUN:=IN;
END_IF;
```

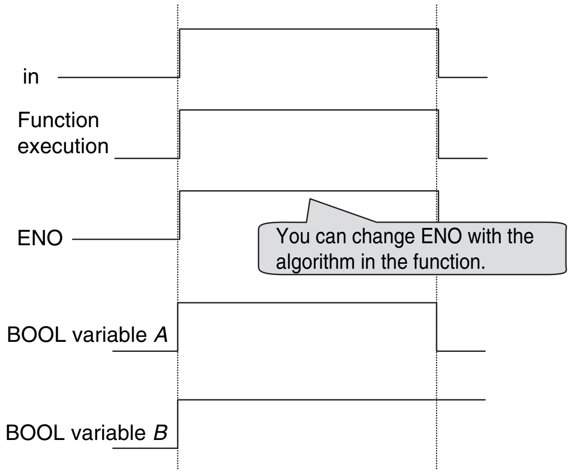
Execution Conditions for Functions

A function is executed when *EN* is TRUE. The function stops processing when *EN* changes to FALSE.

Input variables	Algorithm in FUN		ENO	Operations other than ENO
EN = TRUE	Executed.	Normal end	TRUE	Output parameters: Values are updated according to the internal algorithm. In-out parameters: Values are updated according to the internal algorithm.
		Error end	FALSE	Output parameters: Values are retained. In-out parameters: Values are updated according to the internal algorithm.
EN = FALSE	Not executed.		FALSE	Output parameters and in-out parameters: Values are retained.
Inside a master control region	Not executed.		FALSE	Output parameters and in-out parameters: Values are retained.

Example:





6-2-7 Operation That Applies to Both Functions and Function Blocks

Using or Omitting *EN* and *ENO*

The following table shows when you can use and when you can omit *EN* and *ENO* in functions and function blocks.

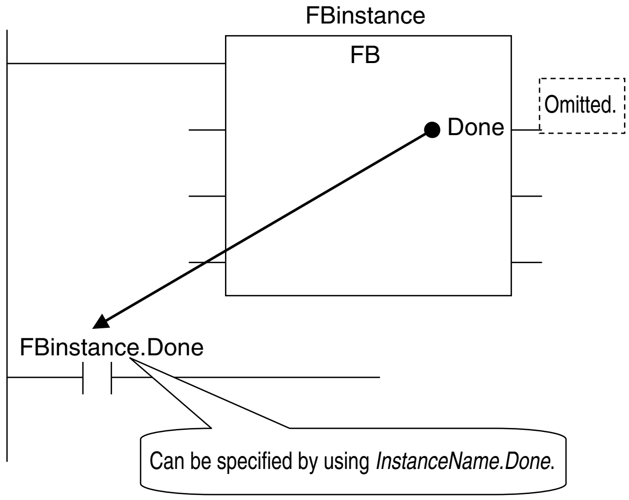
POU		EN	ENO
FB	User-defined functions	Cannot be used. A building error occurs if you try to define *EN* in the variable table from the Sysmac Studio.	Can be used or omitted. You define *ENO* as an output variable in the Sysmac Studio.
	Instruction	All FB instructions do not use *EN*.	Some instructions use *ENO*, and others do not.
FUN	User-defined functions	Required. When you create a function, the Sysmac Studio automatically adds *EN* to the variable table by default.	Can be used or omitted. You define *ENO* as an output variable in the Sysmac Studio.
	Instruction	All FUN instructions use *EN*.	Some instructions use *ENO*, and others do not.

Operation When Parameters Are Omitted

You can omit both input and output parameters.

Parameters omitted in	Operation when omitted	
	FB	FUN
Input parameters to input variables	• When the first time the instance is executed, the initial value is used. • Thereafter, the function block is executed with the previous value (if the input variable is omitted, the initial value is always used).	The initial value is used for operation.
Output parameters from output variables	Can be omitted. You can access the results of the operation outside of the instruction by using *InstanceName.OutputVariableName.**	You can omit the output parameter. If it is omitted, there is no way to retrieve the result of the operation.
In-out parameters to/from in-out variables	Cannot be omitted.	Cannot be omitted.

* You can access the input and output variables of a function block from outside of the function block (but only within the same program) with *InstanceName.VariableName*. However, you cannot access the input and output variables of a function from outside the function.



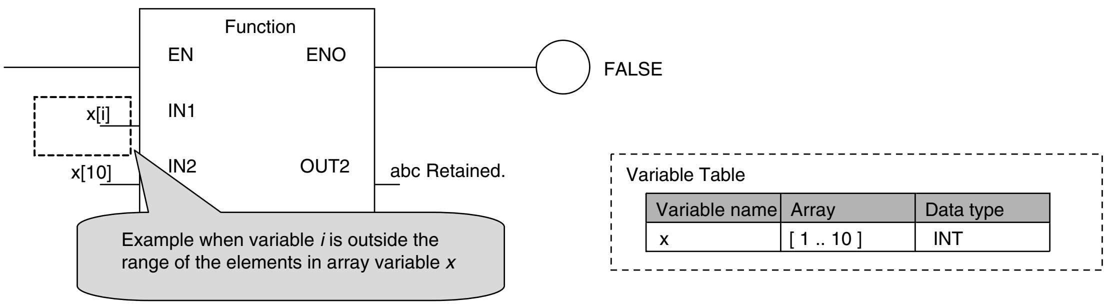
Operation for Parameter Errors

The following operation occurs when there is an error in an input parameter, output parameter, or in-out parameter.

● Errors in Input Parameters

If an error is detected in an input parameter, the function or function block is not executed and *ENO* is FALSE. The power flow output is also FALSE, but all other values are retained.

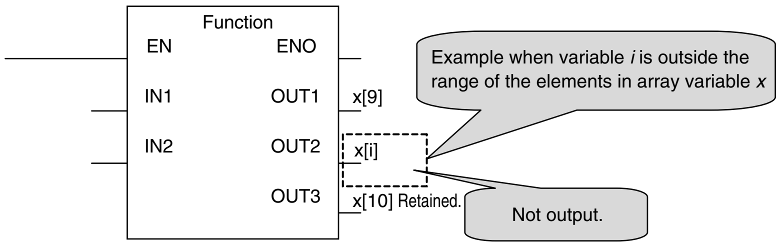
Example:



● Errors in Output Parameters

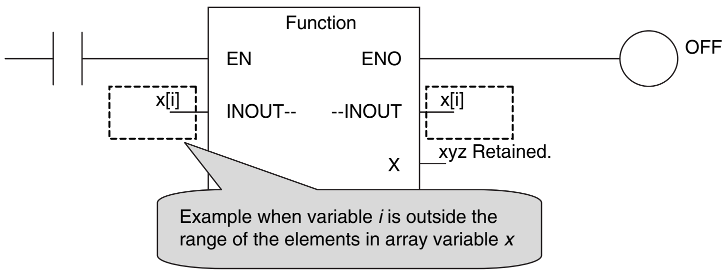
If an error is detected in an output parameter, all values after that parameter are not output but their values are retained.

Example:



● Errors in In-Out Parameters

If an error is detected in an in-out parameter, the function or function block is not executed and *ENO* is FALSE. The power flow output is also FALSE, but all other values are retained.



Recursive Calling

The following recursive calls are not allowed for functions or function blocks. They will result in an error when you build the user program on the Sysmac Studio.

- A function or function block cannot call itself.
- A called function or function block cannot call the calling parent.

6-2-8 POU Restrictions

This section describes the restrictions in the creation of POUs.

Names

Refer to *6-3-12 Restrictions on Variable Names and Other Program-related Names* for restrictions on POU names and function block instance names.

Passing Multiple Arguments

If you need to pass multiple arguments to a function or function block, use an array specification or structure to pass the required data.
This will make your program simpler. However, be aware that if you use an in-out variable, the data passed to the function block or function as a parameter is written and the original data is not retained.

Additional Information

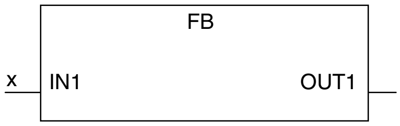
Specifying an Array Variable or Structure Variable as a Parameter

You can also specify an array variable or a structure variable as an input or output parameter. However, it will take longer to pass and receive data for these data types in comparison to a variable with a basic data type (depending on the size). Therefore, when handling array variables or structure variables in a function block, we recommend that you design them in such a way that these variables are passed to and received from in-out variables.

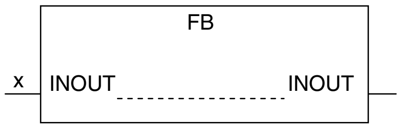
Example 1: Specifying an Array

Program Variable Table

Variable name	Data type
x	ARRAY[0..199] OF INT



Do not assign INT variable *x [0..199]* to the function block input variable *IN1* (Data type of the *IN1* input variable in the function block: ARRAY[0..199] OF INT).

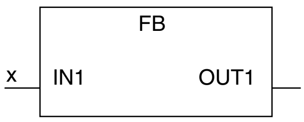


Instead, assign INT variable *x [0..199]* to the *INOUT* in-out variable of the function block. (Data type of INOUT in-out variable in the function block: ARRAY[0..199] OF INT)

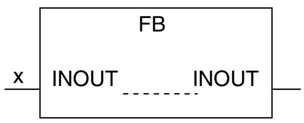
Example 2: Specifying a Structure Variable

Program Variable Table

Variable name	Data type
x	MyStructure



Do not assign MyStructure variable x to the *IN1* input variable of the function block (Data type of *IN1* input variable of the function block: MyStructure).



Instead, assign MyStructure variable x to the *INOUT* in-out variable of the function block. (Data type of *INOUT* in-out variable of the function block: MyStructure)

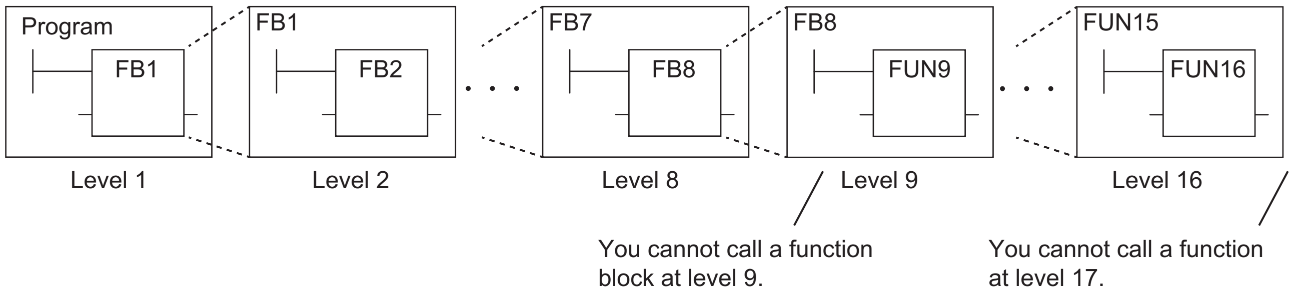
Nesting Levels

Calling another function or function block from a function or function block that was called from a program is called nesting. The limits that are given in the following table apply to the POUs that you can call from a user-defined function or function block and the number of nesting levels. A building error will occur if these limits are exceeded.

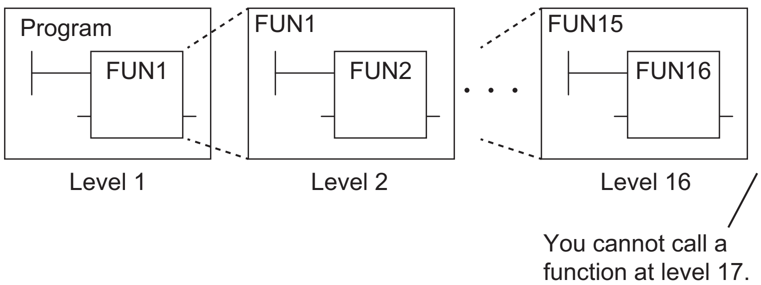
POU	Called POUs	Nesting depth
Function blocks	Functions and function blocks	8 levels max.
Functions	Functions	16 levels max.*1

*1 A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required. For other versions, the limit is 8 levels.

Example 1: From a program, you can call function blocks to a depth of 8 levels. You can then call functions to a depth of 16 levels.



Example 2: From a program, you can call functions to a depth of 16 levels.



6-3 Variables

In the NJ/NX-series System, variables are used to exchange I/O information with external devices, to perform data calculations, and to perform other processes. This section describes variable designations in detail.
Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details on setting variables with the Sysmac Studio.

6-3-1 Variables

Variables store I/O data for exchange with external devices or temporary data that is used for internal POU processing. In other words, a variable is a container for data with a name, data type, and other attributes.

You do not need to assign a memory address to a variable. However, you can assign a specific memory address if necessary (see note). The NJ/NX-series CPU Unit automatically allocates memory addresses in the memory area for variables.

* This is done to use specific functions for some CJ-series Special Units. You must specify the CJ-series Unit memory address in the AT Specification attribute of the variable. Refer to *AT Specification* on page 6-56 for details.

* You can use CJ-series Units only with NJ-series CPU Units.

6-3-2 Types of Variables

Variables are broadly classified into the following three types.

● User-defined Variables

The user defines all of the attributes of a user-defined variable. The rest of this section describes user-defined variables.

● Semi-user-defined Variables

These variables are used to access specific devices and data. There are two types of semi-user-defined variables: device variables and cam data variables. Refer to *2-3-1 Types of Variables* and *3-3-1 I/O Ports* for details on device variables.

● System-defined Variables

System-defined variables are provided in advance in an NJ-series CPU Unit. The names and all attributes are defined by the system. They have specific functions. System-defined variables are supplied for each function module. Refer to *A-4 System-defined Variables* for details.

Refer to *2-3-1 Types of Variables* for details on the different types of variables.

6-3-3 Types of User-defined Variables in Respect to POUs

There are six types of user-defined variables as defined according to their function in a POU.

Type of user-defined variable		POU type		
		Programs	FB	FUN
Local variables	Internal variables	Supported.	Supported.	Supported.
	Input variables	Not supported.	Supported.	Supported.
	Output variables	Not supported.	Supported.	Supported.
	In-out variables	Not supported.	Supported.	Supported.
Global variables		Supported (see note).	Supported (see note).	Supported (see note).
External variables		Supported.	Supported.	Supported.

* You can define global variables as external variables to access the global variables through the external variables.

Local Variables

Local variables can be read and written only in the POU (program, function, or function block) in which it is defined. Local variables are the same as internal variables if the POU is a program. If the POU is a function block or a function, “local variable” is a collective term for internal variables, input variables, output variables, in-out variables, and external variables.

● Internal Variables

A local variable can be used only within one POU. An internal variable is declared in the local variable table for the POU. You cannot access the values of internal variables from outside of the POU. You can declare internal variables with the same names in different POUs. Each of those variables is assigned to a different memory area.

● Input Variables

When a POU is called, the input variables are assigned to the values of the input parameters from the calling POU. An input variable is declared in the local variable table of the POU.

● Output Variables

Before processing a POU is completed, the output parameters returned to the calling POU are assigned to the output variables. An output variable is declared in the local variable table of the POU.

● In-Out Variables

When a POU is called, the in-out variables are assigned to the in-out parameters themselves (variable designations) from the calling POU. If you change the value of an in-out variable within a POU, the value of the in-out parameter changes at that time. An in-out variable is declared in the local variable table of the POU.

● External Variables

External variables are used to access data outside of a POU. You can access global variables from POUs.

Global Variables

A global variable is declared in the global variable table.

Device variables that are automatically generated from the Unit configuration and slave configuration and axis/axes group variables that are generated from the Axis Setting Table are automatically registered as global variables.

6-3-4 Attributes of Variables

You can set the following attributes for variables.

Variable Attributes According to Variable Type

● Attributes of Variables

Attribute	Description	Specification	Default
Variable Name	The variable name is used to identify the variable.		
Data Type	The data type defines the format of the data that is stored in the variable.		BOOL
AT Specification	If you want to handle a specific address for a CJ-series Unit as a variable, specify the address to assign to that variable.*1	• Not specified. • Specify.	Not specified.
Retain	Specify whether to retain the value of the variable in the following cases. • When power is turned ON after a power interruption • When the CPU Unit changes to RUN mode • When a major fault level Controller error has occurred.	• Retain: Value specified on the left is retained if there is a Battery. • Non-retain: Changes to initial value.	Non-retain: Reset to initial value
Initial Value	You can select to set or not set an initial value. Initial value setting: Specify the value of the variable in the following cases and do not specify the Retain attribute. • When power turned ON • When operating mode changes • When a major fault level Controller error occurs If the initial value is not set, the value is not retained.	Initial Value • Yes • None	Depends on the data type. (Refer to the section on initial values.)
Constant	If you set the Constant attribute, you can set the initial value of the variable when it is downloaded, but you cannot overwrite the value afterwards.	Specify making the value a constant or not a constant.	
Network Publish	This attribute allows you to use CIP communications and data links to read/write variables from outside of the Controller.	• Do not publish • Publish Only • Input • Output	Do not publish
Edge	An Edge attribute allows you to detect when the input parameter of a function block changes to TRUE or changes to FALSE. This can be used only on BOOL input variables.	• None • Change to TRUE • Change to FALSE	None

*1 You can use CJ-series Units only with NJ-series CPU Units.

Additional Information

Exclusive Control between Tasks

You can restrict writing to global variables to a single task to prevent changes to the values of global variables during processing. Specify this as a task setting, not as a variable attribute.

Attributes Supported by Each Type of Variable

Type of variable		Variable Name	Data Type	AT Specification	Retain	Initial Value	Constant	Network Publish	Edge
Global variables		Supported.	Supported.	Supported.	Supported.	Supported.	Supported.	Supported.	Not supported.
Programs	Internal variables	Supported.	Supported	Supported	Supported	Supported	Supported	Not supported.	Not supported.
	External variables	Not supported.	Not supported.	Not supported.	Not supported.	Not supported.	Supported.	Not supported.	Not supported.
Function blocks	Internal variables	Supported.	Supported.	Supported.	Supported.	Supported.	Supported.	Not supported.	Not supported.
	Input variable	Supported.	Supported.	Not supported.	Supported.	Supported.	Supported.	Not supported.	Supported.
	Output variables	Supported.	Supported.	Not supported.	Supported.	Not supported.	Not supported.	Not supported.	Not supported.
	In-out variables	Supported.	Supported.	Not supported.	Not supported.	Not supported.	Supported.	Not supported.	Not supported.
	External variables	Not supported.	Not supported.	Not supported.	Not supported.	Not supported.	Supported.	Not supported.	Not supported.
Functions	Internal variables	Supported.	Supported.	Not supported.	Not supported.	Supported.	Supported.	Not supported.	Not supported.
	Input variables	Supported.	Supported.	Not supported.	Not supported.	Supported.	Supported.	Not supported.	Not supported.
	Output variable	Supported.	Supported.	Not supported.	Not supported.	Not supported.	Not supported.	Not supported.	Not supported.
	In-out variables	Supported.	Supported.	Not supported.	Not supported.	Not supported.	Supported.	Not supported.	Not supported.
	External variables	Not supported.	Not supported.	Not supported.	Not supported.	Not supported.	Supported.	Not supported.	Not supported.

6-3-5 Data Types

The Data Type attribute defines the type of data and range of data that is expressed by a variable.

The amount of memory that is allocated when you declare a variable depends on the data type of that variable. The more memory allocated, the larger the range of values that the variable can express.

The data types for the input, output, and in-out variables of instructions depend on the instruction. Set the data types of input, output, and in-out parameters for the instruction arguments according to the data types of the input, output, and in-out variables for that instruction.

Basic Data Types and Derivative Data Types

There are two kinds of data types: basic data types, which have predefined specifications, and derivative data types, which are defined according to user specifications.

Basic Data Types

The different kinds of basic data types are listed below.

Classification	Definition
Boolean	A data type with a value of either TRUE or FALSE.
Bit string	A data type that represents a value as a bit string.
Integer	A data type that represents an integer value.
Real number	A data type that represents a real number.
Duration	A data type that represents a time duration (days, hours, minutes, seconds, and milliseconds).

Classification	Definition
Time of day	A data type that represents a specific time of day (hour, minutes, and seconds).
Date	A data type that represents a date (year, month, and day).
Date and time	A data type that represents a date and time (year, month, day, hour, minutes, seconds, and milliseconds).
Text string	A data type that contains a value that represents a text string.

There are a total of twenty different basic data types. The specifications are given in the following table.

The meanings of the data size and alignment columns in the following table are as follows:

- Data size: The actual size of the value.
- Alignment: The unit used to allocate memory.

Classification	Data type	Data size	Alignment	Range of values	Notation
Boolean	**BOOL**	16 bits	2 bytes	FALSE or TRUE	BOOL#1, BOOL#0, TRUE or FALSE
Bit strings	**BYTE**	8 bits	1 byte	BYTE#16#00 to FF	BYTE#2#01011010 BYTE#2#0101_1010 BYTE#16#5A You can also use the "_" character as a separator.
	WORD	16 bits	2 bytes	WORD#16#0000 to FFFF	
	DWORD	32 bits	4 bytes	DWORD#16#00000000 to FFFFFFFF	
	LWORD	64 bits	8 bytes	LWORD#16#0000000000000000 to FFFFFFFFFFFFFFFF	
Integers	**SINT**	8 bits	1 byte	SINT#-128 to +127	100 INT#2#00000000_01100100 INT#8#144 INT#10#100 INT#16#64 –100
	INT	16 bits	2 bytes	INT#-32768 to +32767	
	DINT	32 bits	4 bytes	DINT#-2147483648 to +2147483647	
	LINT	64 bits	8 bytes	LINT#-9223372036854775808 to +9223372036854775807	
	USINT	8 bits	1 byte	USINT#0 to +255	
	UINT	16 bits	2 bytes	UINT#0 to +65535	
	UDINT	32 bits	4 bytes	UDINT#0 to +4294967295	
	ULINT	64 bits	8 bytes	ULINT#0 to +18446744073709551615	
Real numbers	**REAL**	32 bits	4 bytes	REAL#–3.402823e+38 to –1.175495e-38 0 –1.175495e-38 to 3.402823e+38 +∞ /–∞	REAL#3.14 LREAL#3.14 3.14 –3.14 1.0E+6 1.234e4
	LREAL	64 bits	8 bytes	LREAL#–1.79769313486231e +308 to –2.22507385850721e-308 0 2.22507385850721e-308 to 1.79769313486231e+308 +∞ /–∞	
Durations[*1*2]	**TIME**	64 bits	8 bytes	T#-9223372036854.775808ms (T#-106751d_23h_47m_16s_854.775808ms) to T#+9223372036854.775807ms (T#+106751d_23h_47m_16s_854.775807ms)	T#12d3h3s T#3s56ms TIME#6d_10m TIME#16d_5h_3m_4s T#12d3.5h T#10.12s T#61m5s (Equivalent to T#1h1m5s) TIME#25h_3m
Date	**DATE**	64 bits	8 bytes	D#1970-01-01 to D#2106-02-06 (January 1, 1970 to February 6, 2106)	Add "DATE#", "date#", "D#", or "d#" to the beginning of the string and express the date in the yyyy-mm-dd format. Example: d#1994-09-23

Classification	Data type	Data size	Alignment	Range of values	Notation
Time of day*2	**TIME_OF_DAY**	64 bits	8 bytes	TOD#00:00:00.000000000 to TOD#23:59:59.999999999 (00:00:0.000000000 to 23:59:59.999999999)	Add “TIME_OF_DAY#”, “time_of_day#”, “TOD#”, or “tod #” to the beginning of the string and express the time of day in the hh-mm-ss format. Example: tod#12:16:28.12
Date and time*2	**DATE_AND_TIME**	64 bits	8 bytes	DT#1970-01-01-00:00:00.000000000 to DT#2106-02-06-23:59:59.999999999 (January 1, 1970 00:00:0.000000000 to February 6, 2106, 23:59.999999999 seconds.)	Add “DT#” or “dt#” to the beginning of the string and express the date and time in the yyyy-mm-dd-hh:mm:ss format. Example: dt#1994-09-23-12:16:28.12
Text strings	**STRING**	(Number of single-byte characters plus 1) $\times$ 8 bits*3	1 byte	The character code is UTF-8. 0 to 1,986 bytes (1,985 single-byte alphanumeric characters plus the final NULL character, for Japanese, this is approximately equal to 0 to 661 characters)*4 The default size is 256 bytes.	Enclose the string in single-byte single quotation marks (‘). Example: ‘OMRON’‘PLC’

*1 Use the NanoSecToTime and TimeToNanoSec instructions to convert between durations and integer data. Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for detailed instruction specifications.

*2 Variables are compared with nanosecond precision for comparison instructions. To change the precision for comparison, use the TruncTime, TruncDt, or TruncTod instruction. Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for detailed instruction specifications.

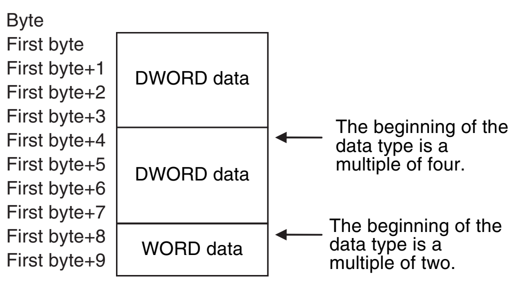
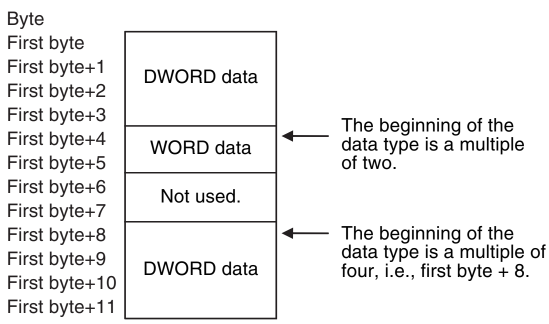
*3 A NULL character (1 byte) is added to the end of text strings. Therefore, reserve memory for one more character than the number of handled characters. For example, if a maximum of 10 single-byte characters are handled, define a STRING variable for 11 characters (11 bytes). STRING[11]

*4 If you want to insert tabs, line break codes, or other special characters, you can use a single-byte dollar sign ($) as an escape character before them. Refer to *Text Strings* on page 6-72 for a list of the escape characters.

Precautions for Correct Use

The total amount of memory required by all variables is not equal to the total of the data sizes of each of those variables. This is because the first position where data is stored in memory is automatically set to a multiple of the alignment value for that data type. This results in some empty space in memory between data types. For example, even if the data types are the same, the overall memory space required depends on the order of data types, as shown below.

Example:

DWORD -> DWORD -> WORD: Requires a total of 10 bytes.	DWORD -> WORD -> DWORD: Requires a total of 12 bytes.
	

You must be aware of the alignment values for different data types when you exchange data such as structure variables between devices so that you can properly align the position of the data in memory. Refer to *A-8 Variable Memory Allocation Methods* for details.

Additional Information

- You cannot compare the sizes of bit string data types (BYTE, WORD, DWORD, and LWORD). If value comparisons are necessary, use instructions such as the WORD_TO_UINT instruction to convert to integer data and compare the values of the integer data variables.
  Example:
  BCD_data: WORD
  IF WORD_BCD_TO _UINT (BCD_data) > UINT#1234 THEN
- You cannot perform logic processing on integer data types (SINT, INT, DINT, LINT, USINT, UINT, UDINT, and ULINT). If logic processing is necessary, use instructions such as the INT_TO_WORD instruction to convert to bit string data and perform the logic processing on the bit string data variables.
  Example:
  In the following sample programming, 1 is added to variable *a* if the value of INT variable *a* is an odd number.

```
IF (INT_TO_WORD (a) AND WORD#16#0001) = WORD#16#0001 THEN
    a = a+1;
END_IF;
```

● Derivative Data Types

A derivative data type is a data type with user-defined specifications. Derivative data types are registered in the Data Type View in the Sysmac Studio. The following is a list of the derivative data types.

Type	Description
Structures	This data type consists of multiple data types placed together into a single layered structure.
Unions	This data type allows you to handle the same data as different data types depending on the situation.
Enumerations	This data type uses one item from a prepared name list as its value.

Refer to *6-3-6 Derivative Data Types* for details.

● Specifications for Data Types

The following array specifications and range specifications are possible for all data types.

Type	Description
Array specification	An array is a group of elements with the same data type. You specify the number (subscript) of the element from the first element to specify the element. You can specify arrays for both basic data types and derivative data types.
Range specification	You can specify a specific range for a data type in advance. You can specify a range for any integer basic data type.

Refer to *6-3-7 Array Specifications and Range Specifications for Data Types* for details.

Additional Information

In addition to basic data types and derivative data types, there are also POU instance data types. A POU instance data type is the data type of a function block instance. To create a function block instance, the instance name is registered as a variable and the function block definition name is registered as a data type in the local variable table.

Restrictions on Using Data Types

A list of the data types that you cannot use in different POUs is given below.

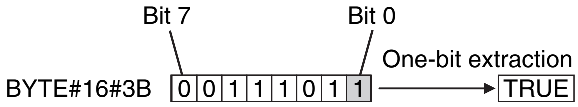
POU type	Type of variable	Unusable data types	
		Basic data types	Derivative data types
Programs	Internal variables	None	
	Global variables	None	
FUN	Input variables, output variables, and in-out variables	None	Unions
	Internal variables	None	
	Return values	None	A structure or union
FB	Input variables, output variables, and in-out variables	None	Unions
	Internal variables	None	

Bit String, Real Number, and Text String Data Formats

This section describes the data formats for bit string data, real number data, and text string data.

● Bit String Data Format

Bit 0 is the least significant bit of a bit string variable. Bit values are represented by values of either 1 or 0. However, you can also represent the value of a single bit as a BOOL variable where 1 equals TRUE and 0 equals FALSE.



● Real Numbers (REAL and LREAL Data)

REAL and LREAL data have a real number data format. This section describes how to express real numbers and how to perform data processing with real number data types.

Data Size

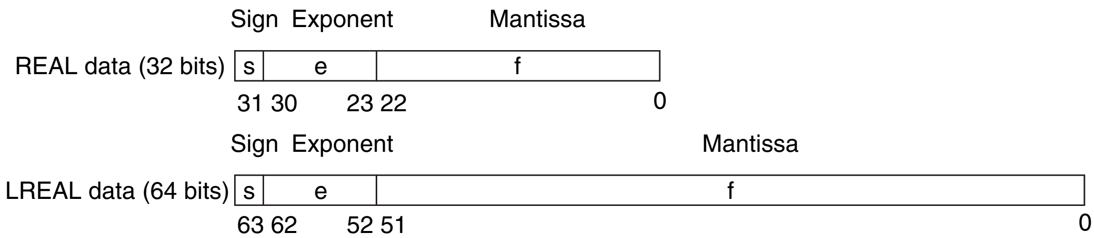
REAL data is 32 bits, while LREAL data is 64 bits.

Data Formats

The floating-point format is a way to express a real number as a combination of a sign, an exponent, and a mantissa. To express a real number as shown below, the value of s is the sign, the value of e is the exponent, and the value of f is the mantissa.

- REAL Data
  Number = $(-1)^s 2^{e-127}(1+f \times 2^{-23})$
- LREAL Data
  Number = $(-1)^s 2^{e-1023}(1+f \times 2^{-52})$

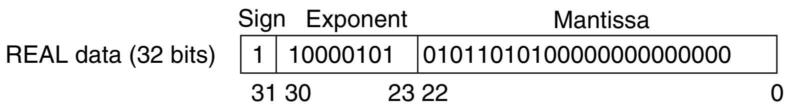
This floating-point format follows the IEEE 754 standard. The formats are given below.



Example: Expressing –86.625 as REAL Data

1. This is a negative number, so s =1.
2. 86.625 in binary is 1010110.101.
3. Normalizing this value gives us 1.010110101×2^6.
4. From the above expression we can determine that e–127 = 6, so e = 133 (or 10000101 in binary).
5. Next we take the value after the decimal part of 1.010110101, which is 010110101. This is not enough for the 23-bit mantissa, so *f* is this number with the required amount of zeroes added to the end. Therefore, f = 01011010100000000000000.

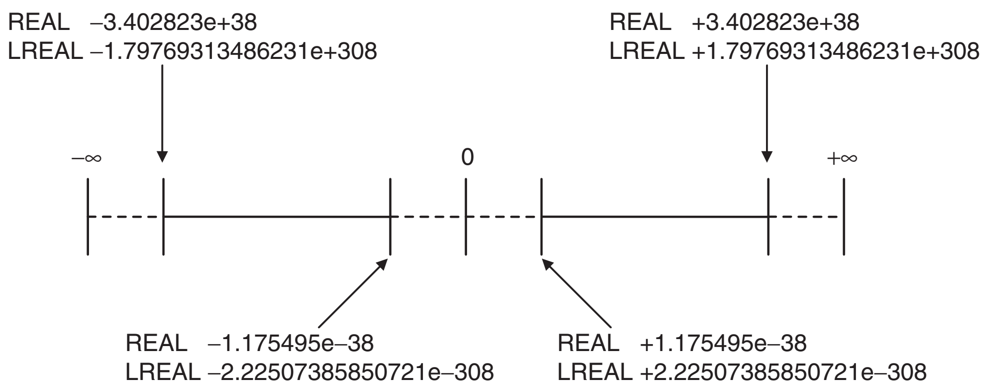
Therefore, you can express –86.625 as shown in the following figure.



Valid Ranges

The valid ranges for REAL and LREAL data are shown in the following table. There are a range of values that you cannot express as you approach 0.

Data type	−∞	Negative numbers	0	Positive numbers	+∞
REAL	−∞	−3.402823e+38 to −1.175495e-38	0	+1.175495e−38 to +3.402823e+38	+∞
LREAL	−∞	−1.79769313486231e+308 to −2.22507385850721e−308	0	+2.22507385850721e−308 to +1.79769313486231e+308	+∞



Special Values

Values such as positive infinity, negative infinity, +0, −0, and nonnumeric data are called special values. Nonnumeric data refers to data that you cannot express as a floating-point number and therefore cannot be treated as a numeric value. Although +0 and −0 both mathematically mean 0, they are different for the purpose of data processing. This is discussed later in this section. The values for the sign s, exponent e, and mantissa f of special numbers are given in the following table.

Data type name	Special values	Sign *s*	Exponent *e*	Mantissa *f*
REAL	+∞	0	255	0
	−∞	1	255	0
	+0	0	0	0
	−0	1	0	0
	Nonnumeric	---	255	Not 0

Data type name	Special values	Sign *s*	Exponent *e*	Mantissa *f*
LREAL	+∞	0	2047	0
	−∞	1	2047	0
	+0	0	0	0
	−0	1	0	0
	Nonnumeric	---	2047	Not 0

Subnormal Numbers

You cannot use the floating-point format to express values close to 0 (i.e., values with an extremely small absolute value). Therefore, you can use subnormal numbers to expand the valid range of numbers near 0. You can use subnormal numbers to express values with a smaller absolute value than with the normal data format (normal numbers). Any number where the exponent $e = 0$ and the mantissa $f \neq 0$ is a subnormal number and its value is expressed as shown below.

- REAL Data
  Number = $(-1)^s 2^{-126}(f \times 2^{-23})$
- LREAL Data
  Number = $(-1)^s 2^{-1022}(f \times 2^{-52})$

Example: Expressing 0.75×2^{-127} as REAL Data

1. This is a positive number, so $s = 0$.
2. 0.75 in binary is 0.11.
3. From $(0.11)_2 \times 2^{-127} = 2^{-126}(f \times 2^{-23})$ we can see that $f = (0.11)_2 \times 2^{22}$.
4. From the above expression, f = 01100000000000000000000.

Therefore, you can express 0.75×2^{-127} as shown in the following figure.

	Sign	Exponent	Mantissa
REAL data (32 bits)	0	00000000	01100000000000000000000
	31	30 … 23	22 … 0

Subnormal numbers have less effective digits than normal numbers. Therefore, if a calculation with normal numbers results in a subnormal number or if a subnormal number results in the middle of such a calculation, the effective digits of the result may be less than the effective digits of a normal number.

Data Processing

The floating-point format expresses only an approximate value. Therefore, there may be a difference between the floating-point number and its true value. There is also a limited number of effective digits for these values. Therefore, the following actions are taken when you perform calculations with the floating-point format.

Rounding

If the real value exceeds the effective digits of the mantissa, the value is rounded off according to the following rules.

- The result of the calculation will be the closest value to the value that can be expressed as a floating-point number.
- If there are two values that are the closest to the real value (e.g., if the real value is the median value of two approximate values), the mantissa with a least significant bit value of 0 is selected as the result of the calculation.

Precautions for Correct Use

When you determine if two values are equal, consider the true values and error.

A real number is expressed in the floating-point decimal format. Because of this, there is a slight error from the actual value. When you try to determine if two values are equal, this error may cause unintended results. For example, if you compare 0.1 + 0.2 with 0.3 using *boolv := (0.1 + 0.2 = 0.3);*, the BOOL variable *boolv* will not be TRUE. It will be FALSE. To prevent this situation, do not use the EQ, =, NE, or <> instruction to determine if two real numbers are equal. Instead, use the value comparison instructions and determine if the absolute value of the difference between the two values is within a sufficiently small range. For example, the following programming can be used to check to see if the sum of REAL variables *real_a* and *real_b* is equal to 0.3. If the value of *boolv* is TRUE, the two values are considered to be equal.

```
boolv := (ABS((real_a + real_b) - 0.3) < 0.000001);      // Here, an allowable error
                                                         // of 0.000001 is used.
```

Overflows and Underflows

An overflow occurs when the absolute value of the true value is larger than the maximum value that can be expressed in the floating-point format. An underflow occurs when the absolute value of the true value is smaller than the minimum value that can be expressed in the floating-point format.

- If an overflow occurs and the true value is positive, the result of the calculation is positive infinity. If the true value is negative, the result of the calculation is negative infinity.
- If an underflow occurs and the true value is positive, the result of the calculation is positive zero. If the true value is negative, the result of the calculation is negative zero.

Special Value Calculations

Calculations that involve special values (i.e., positive infinity, negative infinity, +0, −0, and nonnumeric data) are performed according to the following rules.

- Addition of positive and negative infinity results in nonnumeric data.
- Subtraction of two infinite values of the same sign results in nonnumeric data.
- Multiplication of +0 or −0 with infinity results in nonnumeric data.
- Division of +0 by itself, −0 by itself, or infinity by itself results in nonnumeric data.
- Addition of positive and negative zero results in positive zero.
- Subtracting +0 from itself or −0 from itself results in +0.
- Any arithmetic that involves nonnumeric data results in nonnumeric data.
- Comparison instructions (such as for the Cmp instruction) treat +0 and −0 as equal.
- If you compare nonnumeric data with anything else, the result is always not equal.

● Text String Data Format

All STRING variables are terminated with a NULL character (character code BYTE#16#00).

Converting Data Types

When you use a variable of a different data type, the data type is automatically converted in some cases. You can also perform the conversion yourself with a data type conversion instruction.

Data Type Conversion

All variables must have data types. Programs must operate properly according to these data types. For example, the left and right sides of an assignment expression should normally use the same data type. In some cases, however, it may be necessary to assign data of a different data type to a variable in order to program something successfully.

Example:

var3 := var1; ——— Assigning a value to a variable of a different data type

var1 is a variable of data type INT.

var3 is a variable of data type REAL.

In order to assign the data in *var1* to the data type of *var3*, the data must first be converted. This type of conversion is called “data type conversion” or just “type conversion” for short.

● When Data Type Conversion Occurs

Converting between data types occurs in the following two cases.

(1) Conversion by User Execution of Data Type Conversion Instructions

(2) Automatic Conversion for Assignments and Instructions

- ST assignments
- Connecting lines in ladder diagrams

Additional Information

Use the NanoSecToTime and TimeToNanoSec instructions to convert between INT and TIME data. Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for details.

6-3-6 Derivative Data Types

A derivative data type has a configuration that is based on one of the basic data types. The following is a list of the derivative data types.

- Structures
- Unions
- Enumerations

Refer to *6-3-12 Restrictions on Variable Names and Other Program-related Names* for restrictions on the number of characters in data type names and other restrictions when you create a derivative data type.

Additional Information

NJ/NX-series Controllers come with three different types of system-defined derivative data types.

- System-defined variables that are structures
- Structures used for input, output, and in-out variables for instructions
- Structures for Special Unit expansion memory (You must register these in the Unit Editor to use them.)

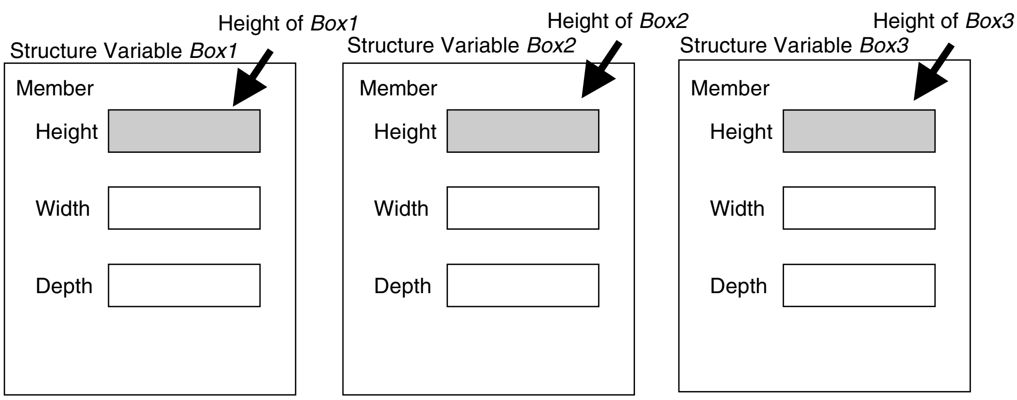
Structures

A structure is a derivative data type that groups together data with the same or different variable types. You can easily change data and add new data if you place your data into a structure.

For example, you can define a “Box” structure that has three members (Width, Height, and Depth) in order to organize and group your data.

You can then use this structure data type to add a variable called *Box1*. You can then use it to access the different levels of the data by placing a period after the variable name followed by the name of the data you want to access. For example, *Box1.Width* or *Box1.Height*.

If you need to create a new variable to store more box data, you can perform the same steps to add a new variable called *Box2* to the variable table.



When a structure is used for a variable in an instruction, it is necessary to select a structure for the input parameter, output parameter, or in-out parameter, and register the variable.

Example: Communications Instructions

- **Expressing Structure Variables and Structure Variable Members**

Specifying Members

The individual pieces of data that make up a structure are called “members.” You can express individual members of a structure by putting a period after the variable name that represents the entire structure followed by the member name that you want to access. You can even have a structure that is the member of another structure.

Example: *abc.x*: Member *x* of structure variable *abc*

abc.Order.z: Member *z* of member structure variable *Order* of structure variable *abc*

Specifying the Structure

The structure represents all members that make up the structure. A structure is expressed by the name of the structure variable. In the example above, you would write *abc*.

- **Creating a Structure**

1 Create a structure data type in the Data Type Table.

Specify the data type name, members, and the data type.

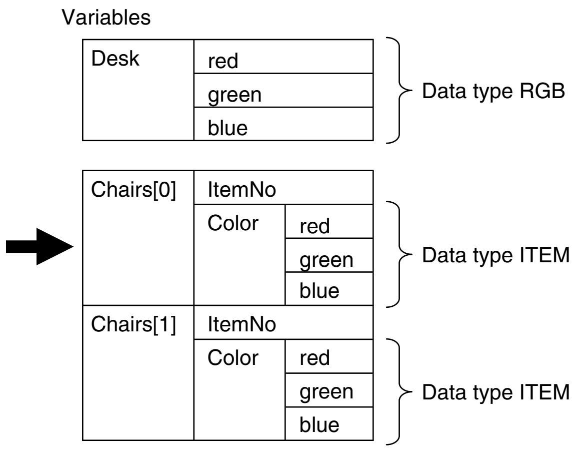
Data Type Table

Name	Member	Data type
RGB	red	INT
	green	INT
	blue	INT
ITEM	ItemNo	INT
	Color	RGB

2 Specify the member name and the structure data type from above as the data type and register the variable in the variable table.

Variable Table

Variable name	Data type
Desk	RGB
Chairs	ARRAY[0..1]OF ITEM



● Structure Specifications

The specifications of structure data types are given in the following table.

Item	Specification
Structure names	Names are not case sensitive. Prohibited characters and character length restrictions are the same as for variable names.
Member data types	Refer to the table on the data types of structure members that is given below for details.
Member attributes	Member Name Comment
Number of members	1 to 2,048
Nesting depth of structures	Maximum of 8 levels (however, a member name must be 511 bytes or less, including the variable name)
Maximum size of one structure variable	NX701-□□□□: 8 MB NJ501-□□□□: 4 MB NJ301-□□□□: 2 MB NJ101-□□□□: 2 MB

Data Types of Structure Members

Classification	Data type	Usage
Basic data types	Boolean, bit string, integer, real, duration, date, time of day, date and time, or text string data	Supported.
	Array of Boolean, bit string, integer, real, duration, date, time of day, date and time, or text string data	Supported.
Derivative data types	Arrays (see note), unions, and enumerations **Note** Recursions and loops are not allowed. (An error will occur when the program is checked.)	Supported.
	Array specifications for structures, unions, and enumerations	Supported.
POU instances		Not supported.

● Arrays and Structures

You can set an array in which the elements are structures. You can also set a structure in which the members are arrays.

● Specifying Structure Member Offsets

When you specify an offset for a member, you can set the memory configuration of the members as required for each structure data type. This allows you to align the memory configuration of the members of the structure data type when you use tag data links with CJ-series CPU Units or with other external devices.

You can select *NJ*, *CJ*, or *User* as the offset type for structure members. If you select *NJ*, the memory configuration that is optimum for the NJ/NX-series is automatically used. Refer to *A-8 Variable Memory Allocation Methods* for details on the memory configuration of NJ/NX-series Controllers. Refer to *A-8-2 Important Case Examples* for examples of tag data lings with CJ-series CPU Units.

The meanings of the offset type are as follows:

Offset type	Meaning
NJ	The memory configuration that is optimum for the NJ/NX-series Controllers is automatically used and operating speed is maximized.
CJ	The memory configuration for CJ-series PLCs is automatically used. This allows you to use the same memory configuration as a CJ-series CPU Unit.
User	You can set the memory offsets for each member. This allows you to use the same memory configuration as external devices other than CJ-series CPU Units.

Version Information

The following table gives the unit version of the CPU Units and the Sysmac Studio version that are required to specify member offsets.

Unit version of CPU Unit	Sysmac Studio version		
	1.01 or lower	1.02	1.03 or higher
1.01 or later	Not possible.	Possible.*	Possible.
1.00	Not possible.	Not possible.	Not possible.

* You cannot select the memory offset type. You can set member offsets.

Setting Offsets

If you set the memory offset type to *User*, you can set memory offsets for each member of the structure. There are byte offsets and bit offsets. If you set the memory offset type to *NJ* or *CJ*, the memory configuration is determined automatically. You do not need to set offsets.
The meanings of the offsets are as follows:

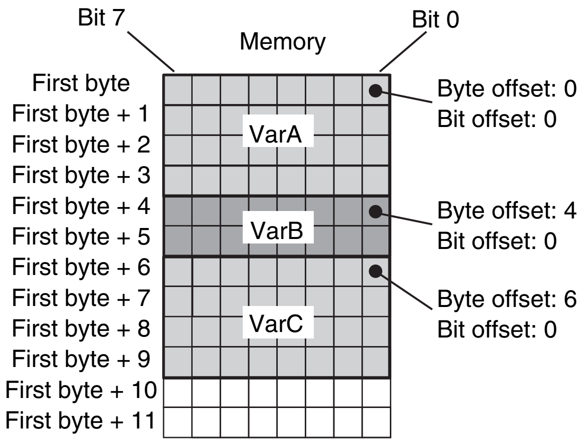
Offset	Meaning	Unit	Range of values
Byte offset	The byte offset is the offset of a member from the start of the structure. Bytes offsets are used for all basic data types and derivative data types.	Byte	0 to 8191*1
Bit offset	The bit offset is the offset of a member from the start of the byte position that is specified with the byte offset.	Bit	0 to 63

*1 For NJ-series CPU Units, the range of values is 0 to 1023.

Example:

This example shows the memory configuration when the following settings are made with the Structure Editor.

Name	Data type	Offset type	Byte offset	Bit offset
StrA	STRUCT	User		
VarA	DINT		0	0
VarB	INT		4	0
VarC	DINT		6	0



Offsets That You Can Set

Even if you set the memory offset type to *User*, the offsets cannot be changed for some data types. The following table shows when offsets can be set.

Classification	Data type	Byte offsets	Bit offsets
Boolean	BOOL	Can be set.	Can be set.
Bit strings	BYTE, WORD, DWORD, LWORD	Can be set.	Fixed.
Integers	SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT	Can be set.	Fixed.
Real numbers	REAL, LREAL	Can be set.	Fixed.
Durations	TIME	Can be set.	Fixed.
Dates	DATE	Can be set.	Fixed.
Times of day	TIME_OF_DAY	Can be set.	Fixed.
Dates and Times	DATE_AND_TIME	Can be set.	Fixed.
Text strings	STRING	Can be set.	Fixed.
Arrays		Can be set.	Can be set only for BOOL elements.
Structures		Can be set.	Fixed.
Unions		Can be set.	Fixed.
Enumerations		Can be set.	Fixed.
POU instances		Fixed.	Fixed.

Restrictions in Specifying Member Offsets

The following restrictions apply to setting member offsets. If you specify member offsets for a structure, the same restrictions apply to structures that are members of that structure.

- If you set the memory offset type to *User* for a structure, you must set offsets for all members of the structure.
- You cannot set initial values for members of structures for which offsets are set. The default initial value for each data type is used. Refer to *When the Initial Value Specification Is Left Blank* on page 6-60.
- The memory size that is required for the structure is determined by the sizes of the members, the alignment values of the data types, and the memory configuration.

Errors in Specifying Member Offsets

The following error can occur when setting member offsets.

Error name	Meaning	Offset type	Correction
Offset Out of Range Error	A value that is out of range was specified for an offset.	User	Change the value of the offset to a suitable value.
Offset Not Set Error	There is a member for which the offsets are not set.	User	Set offsets for all members.
Memory Configuration Overlap Error	The same memory location is allocated to more than one member.	User	Change the values of the offsets to suitable values.
Initial Value Setting Error	An initial value was set for a structure member for which an offset was specified when creating the variable table.	CJ or User	Do not set an initial value.

● Instructions That Take a Structure as a Parameter

Some instructions pass structure variables as parameters. To do so, specify the structure variable as the input parameter.

Example: Passing a Member of a Structure Variable to the MOVE Instruction and Passing a Structure Variable to the MOVE Instruction

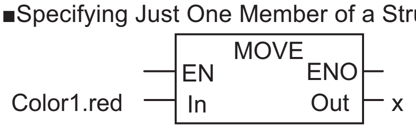
Data Type Table

Name	Member	Data type
RGB	red	UINT
	green	UINT
	blue	UINT

Variable Table

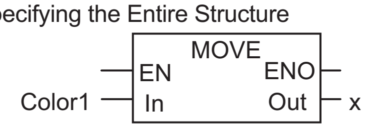
Variable name	Data type
Color1	RGB

■Specifying Just One Member of a Structure



Transferring only the red member of the *Color1* RGB structure to *x*.

■Specifying the Entire Structure



Transferring all members of the *Color1* RGB structure to *x*.

Passing Values to System-defined Structure Input Variables for Certain Instructions

Some instructions take a predefined structure variable as an input variable.

Example: The *Port* input variable for the Serial Communications Instructions (which specifies the target port) is a structure with a data type name of *_sPORT*. When you use one of these instructions, follow the procedure provided below to create a user-defined structure variable and specify that variable for the input parameter to the instruction.

1. The system-defined data type for the instruction is registered in the Sysmac Studio in advance.

   Select that system-defined data type in the Sysmac Studio and add a user-defined structure variable to the variable table.
2. Use the user program or initial values to set the member values of that structure.
3. Specify the structure variable for the input parameter to the instruction.

Unions

A union is a derivative data type that enables access to the same data with different data types. You can specify different data types to access the data, such as a BOOL array with 16 elements, 16 BOOL variables, or a WORD variable.

● Expressing Unions and Union Members

Specifying Members

When you define a union, you must name each data type that can be accessed. These names are called members. You can express individual members of a union by putting a period after the variable name that represents the entire union followed by the member name that you want to access.

Example:

Define the data type as a union as shown for *My Union* in the following example.

Data Type Definition

Name	Member	Data type
My Union	data	WORD
	bit	ARRAY [0..15] OF BOOL

Variable Table

Variable name	Data type
Output	My Union

Output.bit[0]:	This notation specifies the 0th element, or value at bit 00, of union *Output* when it is treated as a 16-bit BOOL array variable.
Output.data:	This notation specifies the value when union *Output* is treated as a single WORD variable.

Specifying the Union

The union represents all members that make up the union. Unions are expressed by the name of the union variable. In the example above, you would write *Output*.

Creating Unions

1 Create a union data type in the Union Table.

Specify the data type names and different data types of the members of the union.

2 Specify the union data type from above as the data type and register the variable in the variable table.

Example:

Here, *OUT16_ACCESS* is defined as the data type of a union. The members of this union are a BOOL array with 16 elements and a WORD variable. The variable *Output* is registered with a data type of OUT16_ACCESS. You can now read/write variable *Output* as a BOOL value for any of the 16 bits and as a WORD value.

Data Type Definitions

Name	Member	Data type
OUT16ACCESS	BoolData	ARRAY[0..15] of BOOL
	ByteData	ARRAY[0..1] of BYTE
	WordData	WORD

Variable name	Data type
Output	OUT16ACCESS

BoolData [15]	. . .	BoolData [8]	BoolData [7]	. . .	BoolData [0]
ByteData [1]			ByteData [0]		
WordData					

```
Output.WordData := WORD#16#1234;
```

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BoolData	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0
ByteData	16#12								16#34							
WordData	16#1234															

```
Output.BoolData[11] :=TRUE;
```

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BoolData	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*
ByteData	Bit 04 of *ByteData[1]* is TRUE								No change							
WordData	Bit 11 of *WordData* is TRUE															

Union Specifications

Item	Specification
Data types that can be specified for members	Refer to the table on the valid data types for union members that is given below.
Number of members	4 max.
Setting initial values	Not supported. Always zero.

Data Types of Union Members

Classification	Data type	Usage
Basic data types	Boolean and bit strings	Supported.
	BOOL and bit string data array specifications	Supported.
	Other basic data types	Not supported.
Derivative data types	Array specification for structures, unions, and enumerations	Not supported.
POU instances		Not supported.

Restrictions

- The initial values for unions are always zero.
- You cannot move unions.
- You cannot specify unions for parameters to POUs.

Enumerations (ENUM)

An enumeration is a derivative data type that uses text strings called enumerators to express variable values. To use an enumeration, you must first set the values that can be obtained from that variable as enumerators (text strings). Use enumerations to make it easier for humans to understand the meaning behind the values of a variable.

● Expressing Enumerations

When you define an enumeration, you must define the possible values of the variable as enumerators and give the enumeration a name.

● Creating Enumerations

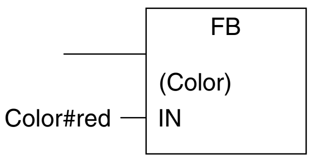
1 Create an enumeration data type in the Enumeration Table.

Set the enumerators and their values for the enumeration.

2 Specify the enumeration data type from above as the data type and register the variable in the variable table.

Example:

Here, *Color* is defined as the data type of an enumeration. For this example, we will set three enumerators: *red*, *yellow*, and *green*. The numbers associated with these enumerators are as follows: *red* = 0, *yellow* = 1, *green* = 2. The variable *DiscColor* will change to one of the following: *red* (0), *yellow* (1), or *green* (2).



Enumeration Table

Data type	
Color	ENUM
Enumerator	**Value**
red	0
yellow	1
green	2

Variable Table

Variable name	Data type
DiscColor	Color

● Enumeration Specifications

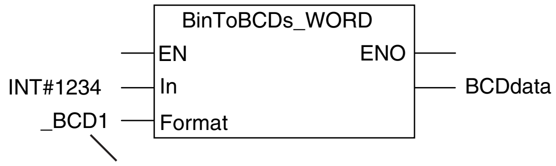
Item	Specification
Enumerator names	Enumerator names consist of single-byte alphanumeric characters. They are not case sensitive. A building error will occur if you specify the same enumerator more than once. A building error will occur if you specify an enumerator with the same name as a variable in the user program or if you specify an enumerator that already exists in another enumeration.
Values	Valid range: Integers between –2,147,483,648 and 2,147,483,647 Values do not have to be consecutive. A compiling error will occur if you specify the same value more than once. **Note** You cannot perform size comparisons with enumeration variables. You can only test to see if the enumerators are the same.
Number of enumerators	1 to 2,048

Notation to Use an Enumerator as a Function Block or Function Parameter

There are the following two notations that you can use to specify an enumerator for a function or function block parameter.

Enumerator Only

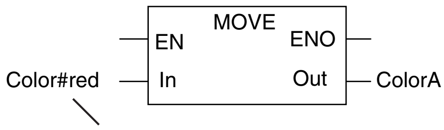
For a function or function block for which the parameter specifies an enumerator, you can just specify the enumerator.
Example: Passing an Enumerator to the BCDToBins_WORD Instruction



The *Format* input variable to the BinToBCDs_WORD is specified as an enumerator.
Therefore, it is necessary to specify only the enumerator.

Enumeration#Enumerator Notation

For a function or function block for which the data type of the parameter is not specified, specifying just the enumerator is not valid. A building error will occur. To clarify that the parameter is an enumerator, the following notation is used: *Enumeration#Enumerator*.
Example: Passing an Enumerator to the MOVE Instruction



The data type of the *In* input variable is not specified.
To pass an enumerator, use the *Enumeration#Enumerator* notation.

Additional Information

For a function or function block for which the parameter specifies an enumerator, you can also use the *Enumeration#Enumerator* notation. Therefore, for the above BinToBCDs_WORD instruction, the following notation can be used to pass the parameter to Format: _eBCD_FORMAT# _BCD1.

Value Checks

When a value is written to an enumerated variable through execution of an instruction, an error will not occur even if that value is not defined as one of the enumerators of that variable. Therefore, if it is necessary to confirm that a value is defined as an enumerator of an enumeration, write the user program to check the value.

6-3-7 Array Specifications and Range Specifications for Data Types

You can specify the following attributes for variables with each data types.

- Array specifications
- Range specifications

Array Specifications (ARRAY[]OF)

Use an array specification for a data type that handles a group of data with the same attributes as a single entity. You can use an array specification for the basic data types and derivative data types. Arrays are useful when you want to handle multiple pieces of data together as you would, for example, coordinate values for motion control.

● Expressing Arrays and Array Elements

Specifying Elements

The individual pieces of data that make up an array are called "elements."
The elements of an array are expressed by adding a subscript (element number) from the start of the array to the name of the variable that represents the entire array.
Enclose the subscript in single-byte braces []. Subscripts can be either constants or variables. In ST, you can also use expressions to express subscripts.

Examples:

Variable Table

Variable name	Data type
Mem	ARRAY[0..99] OF INT

x:=10;
Mem[x]: This expression specifies the xth element of the array variable *Mem* (the variable *x* has a value of 10, so this would point to the 10th element).

Variable Table

Variable name	Data type
Data	ARRAY[0..99] OF INT

x:=10;
y:=20;

Data[x+y]: This expression specifies the x+yth element of the array variable *Data* (the variable *x* has a value of 10 and variable *y* has a value of 20, so this would point to the 30th element).

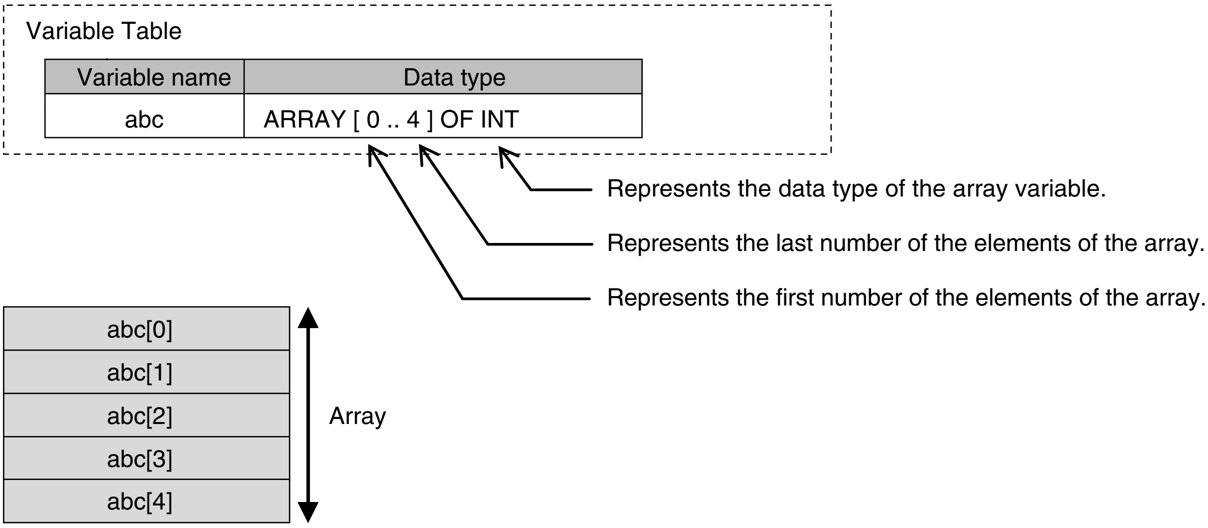
Specifying An Array (i.e., the Entire Array)

The array represents all elements that make up the array. Arrays are expressed by the name of the array variable. In the above examples, the arrays are written as *Mem* and *Data*.

● Creating an Array

1 Enter “A” into the *Data type* Column of the variable table and select *ARRAY[?..?] OF ?* from the list of possible data type name candidates.

2 Enter the number of the first element in the array for the left question mark and the last number for the right question mark in the “[?..?]” section. Next, enter the data type for the question mark in the “OF ?” section and register the variable.



● Array Variable Specifications

<table>
<tr><th>Item</th><th>Specification</th></tr>
<tr><td>Maximum number of elements for an array variable</td><td>65,535</td></tr>
<tr><td>Element numbers</td><td>0 to 65535
The number for the first element in an array does not have to be 0.</td></tr>
<tr><td>Subscripts</td><td>Constants: Integer value between 0 and 65535
Variables:
<table>
<tr><th>Classification</th><th colspan="2">Data type</th><th>Usage</th></tr>
<tr><td rowspan="3">Basic data type</td><td rowspan="2">Integer</td><td>SINT, INT, DINT, USINT, UINT, or UDINT</td><td>Supported.</td></tr>
<tr><td>LINT or ULINT</td><td>Not supported.</td></tr>
<tr><td colspan="2">Boolean, bit string, real, duration, date, time of day, date and time, or text string data</td><td>Not supported.</td></tr>
<tr><td>Derivative data types</td><td colspan="2">Structures, unions, and enumerations</td><td>Not supported.</td></tr>
<tr><td>POU instances</td><td colspan="2"></td><td>Not supported.</td></tr>
</table>
Arithmetic expressions: Arithmetic expressions can be specified only in ST.
Example: y:= x[a+b];</td></tr>
<tr><td>Maximum size of one array variable</td><td>NX701-□□□□: 8 MB
NJ501-□□□□: 4 MB
NJ301-□□□□: 2 MB
NJ101-□□□□: 2 MB</td></tr>
</table>

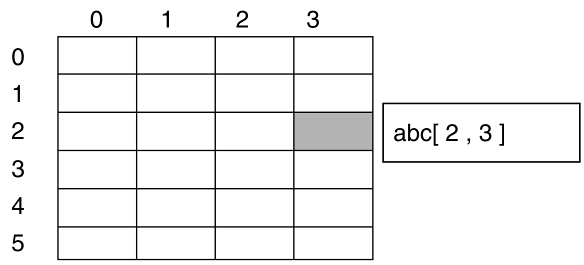
● Dimensions of Array Variables

You can regard the elements of a one-dimensional array as one-dimensional data lined up in a single row. You can set two-dimensional and three-dimensional arrays in the same way. The array elements are expressed by adding the same number of subscripts to the array variable name as the number of dimensions. Arrays can have a maximum of three dimensions.

Two-dimensional Array Specifications

Variable Table

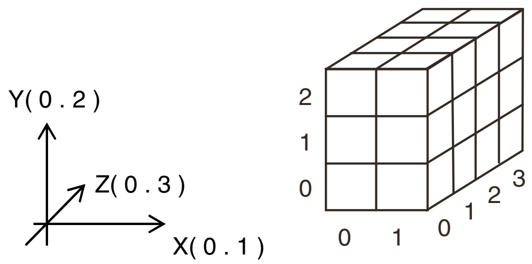
Variable name	Data type
abc	ARRAY [0..5 , 0..3] OF INT



Three-dimensional Array Specifications

Variable Table

Variable name	Data type
ITEM	ARRAY [0..1, 0..2, 0..3] OF INT



● Arrays and Structures

You can set an array in which the elements are structures. You can also set a structure in which the members are arrays.

Arrays with Structure Elements

Date Type Table

Data type	Member	Member
Str	x y	INT DINT

Variable Table

Variable name	Data type
abc	ARRAY [1..4] OF Str

Variables

abc[1].x
abc[1].y
:
abc[4].x
abc[4].y

Structure with Array Members

Data Type Table

Data type	Member	Data type
Str	x	ARRAY [0..1] OF INT
	y	DINT

Variable Table

Variable name	Data type
abc	ARRAY [1..4] OF Str

Variables

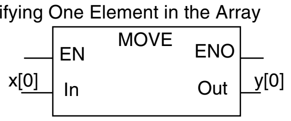
abc[1].x[0]
abc[1].x[1]
abc[1].y
:
abc[4].x[0]
abc[4].x[1]
abc[4].y

● Instructions with an Array Parameter

Some instructions pass array variables as parameters. To do so, specify only the name of the array variable as the input parameter.

Example: Passing a Single Array Element to the MOVE Instruction and Passing an Array to the MOVE instruction

Specifying One Element in the Array

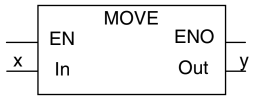


Moves INT variable *x[0]* to *y[0]*

Variable Table

Variable name	Data type
x	ARRAY [0 .. 199]OF INT
y	ARRAY [0 .. 199]OF INT

Specifying the Array

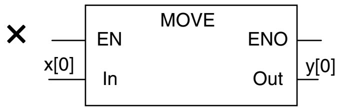


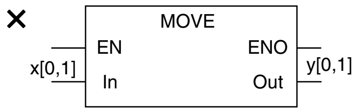
Moves INT variable *x[0..199]* to *y[0..199]*

Restrictions:
When you specify an array variable, it must be moved to a variable of the same data type with the same range of element numbers.

Additional Information

You cannot specify part of a multi-dimensional array as a parameter.





Variable Table

Variable name	Data type
x	ARRAY [0..9 , 0..9 , 0..9]OF INT
y	ARRAY [0..9 , 0..9 , 0..9]OF INT

● Array Protection

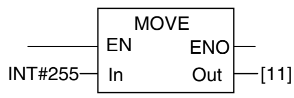
The following errors occur if you attempt to access an element that exceeds the number of elements in an array.

When the Subscript Is a Constant

An error is displayed when you input the variable or when you check the program on the Sysmac Studio.

Variable Table

Variable name	Data type
x	ARRAY [1..10] OF INT

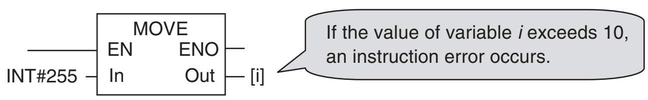


When the Subscript Is a Variable

When an output parameter is assigned to an output variable, the CPU Unit checks to see if the number of elements was exceeded after it executes the instruction. When a subscript variable exceeds the range of the elements of the array variable, an instruction error occurs. Even if this error occurs, the value of *ENO* will be TRUE because internal processing of the instruction ends normally.

Variable Table

Variable name	Data type
x	ARRAY [1..10] OF INT



Range Specifications ((..))

Use the range specification to restrict the values of the following integer variables to specific ranges of values.

Classification	Data type
Integers	SINT, INT, DINT, LINT, USINT, UINT, UDINT, and ULINT

You can check to make sure that the entered value is within the allowed range in the following cases.

- When you specify an initial value for a variable
- When you write a value to a variable with CIP message communications

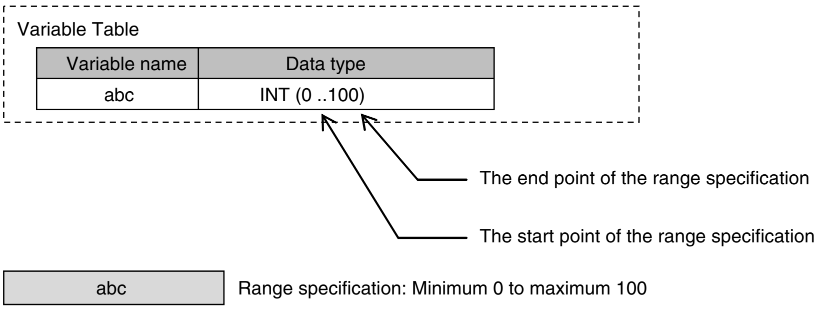
● Making a Range Specification

Input the start point and end point after the data type name in the *Data type* Column in the variable table.

Start point: The minimum value that you can store in the variable.

End point: The maximum value that you can store in the variable.

Example:



● Specifications of Range Specifications

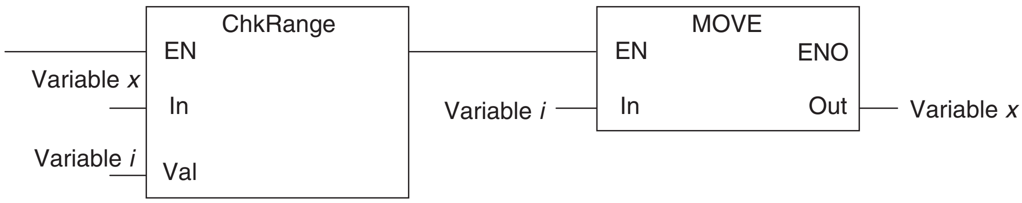
<table>
<tr><th>Item</th><th>Specification</th></tr>
<tr><td>Data types that you can specify</td><td>Variable type</td></tr>
<tr><td>Operation for attempts to write out-of-range value</td><td>
<table>
<tr><th colspan="2">Case</th><th colspan="2">Operation</th></tr>
<tr><td colspan="2">User program</td><td colspan="2">An error does not occur and the value is written. The CPU Unit does not perform a range check when the value of a variable changes due to the execution of an instruction.</td></tr>
<tr><td rowspan="6">Communi-cations</td><td rowspan="5">Write from the Sysmac Studio or a CIP mes-sage</td><td>When the value is an integer</td><td rowspan="3">A command error occurs.</td></tr>
<tr><td>For an element of an integer array variable</td></tr>
<tr><td>For a member of an integer structure</td></tr>
<tr><td>For an integer struc-ture</td><td rowspan="2">A command error does not occur and the value is writ-ten.</td></tr>
<tr><td>For an integer array</td></tr>
<tr><td>Tag data links (both via built-in EtherNet/IP ports and Eth-erNet/IP Units)</td><td colspan="2">An error occurs if you attempt to write to a single member that specifies a range. An error does not occur if you attempt to write to a structure that contains a member for which a range is specified.</td></tr>
<tr><td colspan="2">Input refreshing from slaves and Units</td><td colspan="2">An error does not occur and the value is written.</td></tr>
<tr><td colspan="2">Forced refreshing values</td><td colspan="2">An error does not occur and the value is written.</td></tr>
</table>
</td></tr>
</table>

Precautions for Correct Use

Variables with range specifications are not checked for changes in variable values that result from the execution of instructions in the user program. To check the range of values for a variable that are set from execution of the user program, use instructions that perform range checks.

Variable Table

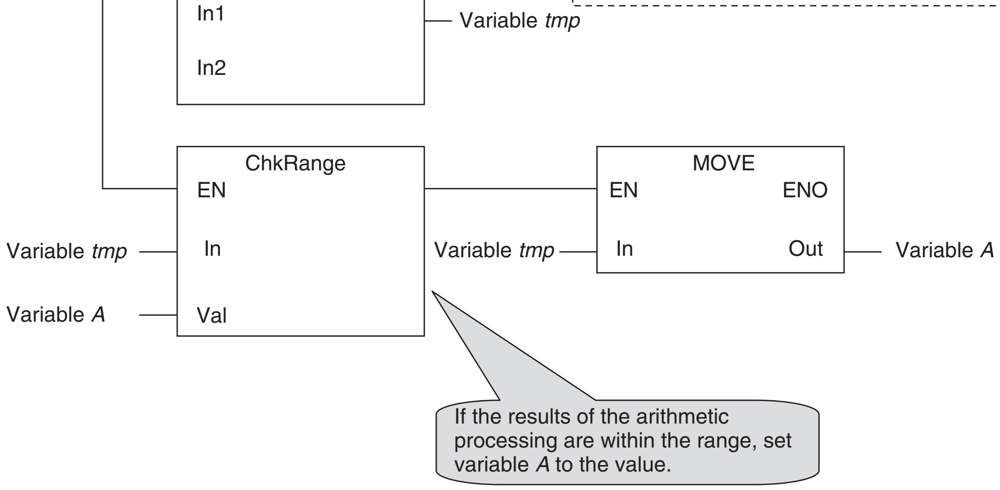
Variable name	Data type
Variable *x*	INT(10 .. 100)



You cannot perform any checks beforehand if you set data with arithmetic processing results. In this case, check the range of values after arithmetic processing (e.g., ADD).

Variable Table

Variable name	Data type
Variable *A*	INT(10 .. 100)



Make sure that the initial value is within the range specified for the Range Specification. If the initial value field on the Sysmac Studio is left blank, an initial value of 0 is used. This applies even if a range that does not include 0 is set for a Range Specification.

6-3-8 Variable Attributes

This section describes the variable attributes other than the Data Type.

Variable Name

The variable name is used to identify the variable. Each variable in a POU must have a unique name. However, you can declare local variables with the same variable name in different POUs. These are treated as two separate variables.

Refer to *6-3-12 Restrictions on Variable Names and Other Program-related Names* for restrictions on variable names.

AT Specification

Use the AT Specification attribute to specify the internal I/O memory address of a variable in memory used for CJ-series Units. AT specifications are used mainly to specify specific memory addresses for the following Special Units.

- Addresses in fixed allocations for DeviceNet Units
- Addresses in user-specified allocations for DeviceNet Units or CompoNet Master Units from the CX-Integrator (A CompoNet Master Unit must be set to communications mode 8 to use the CX-Integrator.)
- Addresses in expansion memory for High-speed Counter Units
- Addresses in expansion memory for Process I/O Units
- Addresses in target node information and user-defined status areas that are used with EtherNet/IP Units

If this attribute is not set, the variable is automatically assigned to an address in variable memory.

Precautions for Correct Use

You can use memory for CJ-series Units only with NJ-series CPU Units.

Additional Information

When you assign a device variable to an I/O port, they are automatically given an AT specification internally.

● Allocation Areas

You can specify addresses in the following areas.

Area	Expression
CIO	CIO 0 to CIO 6143
Work	W0 to W511
Holding	H0 to H1535
DM	D0 to D32767
EM	NJ501-□□□□: E0_0 to E18_32767 NJ301-□□□□: E0_0 to E3_32767 NJ101-□□□□: E0_0 to E3_32767

The following table gives the data assignments by variable data type.

Variable data type	Assignment position
BOOL	You can specify an assignment for each bit.
BYTE/SINT/USINT	You can specify bit 0 or bit 8 of the specified CJ-series address as the start position of the data assignment. Example 1: AT Specification at Bit 0 of D100 (%D100) D100: 16#**12....One-byte data (12) is stored from bit 0. Example 2: AT Specification at Bit 8 of D100 (%D100.8) D100: 16#12**....One-byte data (12) is stored from bit 8.
WORD/INT/UINT	Stored in increments of the data size from bit 0 of the specified CJ-series address.
DWORD/DINT/UDINT **REAL**	
LWORD/LINT/ULINT **LREAL**	
STRING	You can specify bit 0 or bit 8 of the specified CJ-series address as the start position of the data assignment.
TIME **DATE** **TIME_OF_DAY** **DATER_AND_TIME**	Stored in increments of the data size from bit 0 of the specified CJ-series address.

● Variables for Which You Can Set AT Specifications

AT specifications are made separately for each variable. Set them for all elements and members of array, structure, and union variables.

● Attributes of Variables with AT Specifications

	Specification	Remarks
Name	Supported.	
Data Type	Supported.	
Retain	Supported.	An error occurs if the setting of the Retain attribute does not agree with the attribute of the CJ-series Unit memory where the address is assigned.
Initial Value	Supported.	Set the initial value setting to *None* if you want to use the memory value as it is.
Constant	Supported.	You cannot write to a constant with an instruction.
Network Publish	Supported.	
Edge	Not supported.	(You can specify the Edge attribute only for function block input variables.)

● Entering and Displaying AT Specifications

When you specify the AT Specification attribute, input the following in the Allocated Address Box of the variable table in the Sysmac Studio. The following is displayed in the Allocated Address Box of the variable table or the I/O Map.

Type of variable	Entries and displays in the *AT* field.	Example
User-defined variables with AT specifications to word addresses	%[*word_address*]	%D100
User-defined variables with AT specifications to bit addresses	%[*word_address*].[*bit_position*]	%W0.00

The following variables are also allocated an address internally. The following is displayed in the Allocated Address Box.

Type of variable	Displays in the *AT* field.	Example
Device variables for CJ-series Units	IOBus://rack#[*rack_number*]/slot#[*slot_number*]/[*I/O_port_number*]	Basic I/O Units: IOBus://rack#0/slot#1/Ch1_In/Ch1_In00 Special Units: IOBus://rack#0/slot#1/PeakHoldCmd/ch1_PeakHoldCmd
Device variables for EtherCAT slaves	For NX Units on EtherCAT Slave Terminals: ECAT://node#[*node_address.NX_Unit_number*]/[*I/O_port_name*]	ECAT://node#[10.15]/Input1
	Other device variables: ECAT://node#[node_address]/[I/O_port_name]	ECAT://node#1/Input1
Axis Variables	MC://_MC_AX[] MC://_MC1_AX[]*1 MC://_MC2_AX[]*1	MC://_MC_AX[1]
Axes Group Variables	MC://_MC_GRP[] MC://_MC1_GRP[]*1 MC://_MC1_GRP[]*1	MC://_MC_GR[1]

*1 These system-defined variables are available only for the NX-series CPU Unit.

Precautions for Correct Use

You can assign the same address to more than one variable. However, this is not recommended as it reduces readability and makes the program more difficult to debug. If you do this, set an initial value for only one of the variables. If you set a different initial value for each individual variable, the initial value is not stable.

Retain

Use the Retain attribute to specify whether a variable should retain its value in the following cases.

- When power is turned ON after power interruption
- When the operating mode is changed
- When a major fault level Controller error occurs

If the Retain attribute is not set, the value of the variable is reset to its initial value in the above situations.

You can specify the Retain attribute when you need to retain the data that is stored in a variable (such as the manufacturing quantities) even after the power to the Controller is turned OFF.

For a variable with an AT specification, the setting of the Retain attribute must agree with address in the memory area where the address is assigned.
(Retained areas: Holding, DM, and EM Areas
Non-retained areas: CIO and Work Areas)

● Conditions Required to Enable the Retain Attribute

The CPU Unit must contain a Battery.

● Using Initial Values for Retain Variables

When you download the user program, select the *Clear the present values of variables with Retain attribute* Check Box.

Operation with and without the Retain Attribute

The following table shows when variable values are retained or not.

Case		Values of variables: Retain attribute specified	Values of variables: Retain attribute not specified
When power is turned ON after power interruption		Retained.	Not retained.
When the operating mode is changed			
When a major fault level Controller error occurs			
When you download the user program	**When the *Clear the present values of variables with Retain attribute* Check Box is selected.**		
	When the check box is not selected.	Not retained.	

Variables for Which You Can Specify the Retain Attribute

AT specifications are made separately for each variable. Set them for all elements and members of array, structure, and union variables.

Initial Value

The variable is set to the initial value in the following situations.

- When power is turned ON
- When changing between RUN mode and PROGRAM mode
- When you select the *Clear the present value of variables with Retain attribute* Check Box, and download the user program
- When a major fault level Controller error occurs

You can set an initial value for a variable in advance so that you do not have to write a program to initialize all of the variables. For example, you can preset data such as a recipe as initial values. You do not have to set any initial values.

Types of Variables That Can Have Initial Values

You can set initial values for only some types of variables. A list is provided below.

Type of variable	Initial Value
Global variables	Supported.
Internal variables	
Input variables	
Output variables	
Return values of functions	Not supported.
In-out variables	
External variables	

● Enabling an Initial Value

You can specify whether a variable has an initial value when you create the variable.

Initial Value Specified

Initial value

(Blank) Or

Initial value
3.14

No Initial Value Specified

Initial value
None

The following table shows the variables for which you can set an initial value.

Type		Example	Enabling an Initial Value
Basic data type variables		**aaa**	Supported.
Array variables	**Arrays**	**bbb**	Supported.
	Elements	**bbb[2]**	Not supported.
Structure variables	**Structures**	**ddd**	Supported.
	Members	**ddd.xxx**	Not supported.
Union variables	**Unions**	**eee**	Not supported (initial values are always 0).
	Members	**eee.word**	Initial values are always 0.
Enumerated variables		**ccc**	Supported.
POU instances		**instance**	Not supported.

Additional Information

Some Basic I/O Units have more than one access method for the same I/O port, such as bit string data and BOOL data. If you use initial values for this type of I/O port, set the initial values for one of the access types to None.

● When Initial Values Are Set

The initial value is assigned to the variable at the following times.

- When power is turned ON
- When the operating mode changes from PROGRAM to RUN mode or from RUN to PROGRAM mode
- When you select the *Clear the present value of variables with Retain attribute* Check Box, and download the user program
- When a major fault level Controller error occurs

● When the Initial Value Specification Is Left Blank

The following initial values are used for variables for which the initial value specification is left blank.

Data type		Default initial value
Boolean and bit strings	**BOOL, BYTE, WORD, DWORD, and LWORD**	0
Integers	**SINT, INT, DINT, LINT, USINT, UINT, UDINT, and ULINT**	0
Real numbers	**REAL and LREAL**	0.0

Data type		Default initial value
Durations, dates, times of day, and dates and times	**TIME**	T#0S
	DATE	D#1970-01-01
	TIME_OF_DAY	TOD#00:00:00
	DATE_AND_TIME	DT#1970-01-01-00:00:00
Text strings	**STRING**	‘ ‘(blank character)

● Initial Value of Array Variables

Data type	Initial value specifications
Array specifications	• You can specify an initial value for each element. • To specify initial values, you must specify a value or leave the specification blank for each element.

● Initial Values for Derivative Data Types

You do not specify an initial value for the data type itself. You set an initial value for each individual variable.

Data type	Initial value specifications
Structures	• You can specify an initial value for each member. • To specify initial values, you must specify a value or leave the specification blank for each element.
Unions	• Initial values cannot be specified. Always zero.
Enumerations	• Initial values can be specified.

● Variables That Do Not Apply Initial Values

For the following variables, initial values are not applied when the power is turned ON, and the values before the power interruption are retained.

- Variables with Retain attribute
- Variables with AT specifications (retained areas or DM, Holding, or EM Area specifications only)

Precautions for Correct Use

If the CPU Unit has no Battery, the above variables are also initialized.

Constant

If you specify the Constant attribute, the value of the variable cannot be written by any instructions, ST operators, or CIP message communications. Setting the Constant attribute will prevent any program from overwriting the variable. The values of variables with a Constant attribute cannot be written from instructions after the initial value is set. If there is an instruction in a POU that attempts to write a value to a variable with the Constant attribute, an error will occur when the user program is compiled.

- **Operation**

If there is an instruction or operator in a POU that attempts to write a value to a variable with the Constant attribute, the following operations will occur.

Source		Operation for attempts to write the value
User program		An error is detected during the program check. The Sysmac Studio checks the program when it is built. A building error occurs at that time.
Communications	**Writing from Sysmac Studio**	Not supported.
	CIP messages	A command error occurs.
	Tag data links	An error occurs when a tag data link starts. The tag data link will continue to operate. However, the values of variables with the Constant attribute are not written.
Input refreshing from slaves and Units		An error does not occur and the value is written.
Forced refreshing		

- **Range for Constant Attribute Specification**

The Constant attribute is specified separately for each variable. Set them for all elements and members of array, structure, and union variables.

Additional Information

You cannot write to variables with the Constant attribute from the user program.

Network Publish

The Network Publish attribute allows a variable to be read/written from external devices (other Controllers, host computers, etc.) through CIP message communications or tag data links. If this attribute is not set, you can read/write the variable from the Controller that declared the variable and external devices (other Controllers, host computers, etc.) cannot read/write that variable.

Variables that have been published to the network are called network variables.

- **Network Publish Specifications**

There are three specifications for publishing variables to the network: Publish Only, Input, and Output. The specifications are given in the following table.

Network Publish		Specifications
Do not publish		You cannot access a variable with this attribute from external devices. However, Support Software can still access the variable regardless of this setting.
Publish	Publish Only	You can access a variable with this attribute from external devices through CIP communications. Tag data links are not possible for variables with this attribute setting.
	Input	You can access a variable with this attribute from external devices through CIP communications or a tag data link. For tag data links, this will be a variable for data input (from another CPU Unit to the local CPU Unit).
	Output	You can access a variable with this attribute from external devices through CIP communications or a tag data link. For tag data links, this will be a variable for data output (from the local CPU Unit to another CPU Unit).

Ranges for Published to the Network

The Network publish attribute is specified separately for each variable. Set them for all elements and members of array, structure, and union variables.

Edge

The Edge attribute makes the variable pass TRUE to a function block when a BOOL variable changes from FALSE to TRUE or from TRUE to FALSE. You can specify the Edge attribute only for BOOL input variables to function blocks.

Application

Use the Edge attribute when you want the function block to accept the input only when the input parameter changes from FALSE to TRUE or from TRUE to FALSE. For example, you can use this attribute when you want to execute the function block any time there is a change detected in an input parameter.

Operation

- If you specify a change to TRUE, the input variable changes to TRUE only when the input parameter connected to that input variable changes from FALSE to TRUE.
- If you specify a change to FALSE, the input variable changes to TRUE only when the input parameter changes from TRUE to FALSE.

Specification	Value of input parameter	Value of variable
Change to TRUE	FALSE to TRUE	TRUE
	Other	FALSE
Change to FALSE	TRUE to FALSE	TRUE
	Other	FALSE
None	---	Changes according to the input parameter value.

6-3-9 Changes to Variables for Status Changes

The values of variables in the CPU Unit will change as shown in the following table when the power is turned ON, when the operating mode changes, when the variable table is downloaded, when a major fault level Controller error occurs, or during online editing.

<table>
<tr><th rowspan="2">Retain attribute of variable</th><th rowspan="2" colspan="2">Type of variable</th><th rowspan="2">When power is turned ON</th><th rowspan="2">When operating mode changes</th><th colspan="2">After downloading</th></tr>
<tr><th>When the Clear the present values of variables with Retain attribute Check Box is selected.</th><th>When the check box is not selected.</th></tr>
<tr><td rowspan="5">Non-retain</td><td colspan="2">User-defined variables</td><td colspan="4">• If initial values are set, the variables change to the initial values.
• If initial values are not set (None), the variables change to 16#00.*1</td></tr>
<tr><td></td><td>AT specifications for CIO and Work Area addresses in the memory used for CJ-series Units</td><td colspan="2">• If initial values are set, the variables change to the initial values.
• If initial values are not set (None), the variables change to 16#00.*1</td><td colspan="2">• If initial values are set, the variables change to the initial values.
• If initial values are not set (None), the previous value is retained.</td></tr>
<tr><td colspan="2">Device variables for EtherCAT slaves</td><td colspan="4">• If initial values are set, the variables change to the initial values.
• If initial values are not set (None), the variables change to 16#00.*1</td></tr>
<tr><td colspan="2">Device variables for CJ-series Units</td><td colspan="2">• If initial values are set, the variables change to the initial values.
• If initial values are not set (None), the variables change to 16#00.*1</td><td colspan="2">• If initial values are set, the variables change to the initial values.
• If initial values are not set (None), the previous value is retained.</td></tr>
<tr><td colspan="2">CIO and Work Area addresses in the memory used for CJ-series Units*2</td><td colspan="2">16#00</td><td colspan="2">The previous values are retained.</td></tr>
</table>

Retain attribute of variable	Type of variable		When power is turned ON	When operating mode changes	After downloading: When the *Clear the present values of variables with Retain attribute* Check Box is selected.	After downloading: When the check box is not selected.
Retain	User-defined variables	Retain condition is met.*3	No change (retains value before power interruption).	No change (i.e., the values in RUN mode are retained).	• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to 16#00.*1	The values from before the download are retained.
		Retain condition is not met.*3				• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to 16#00.*1
		AT specifications for Holding, DM, and EM Area addresses in the memory used for CJ-series Units			• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to the values of the memory addresses in the AT specifications.	
	Device variables for CJ-series Units				• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to the values of the memory addresses in the AT specifications.	The variables change to the values of the memory addresses in the AT specifications.
	Holding, DM, and EM Area addresses in the memory used for CJ-series Units*2				The previous values are retained. • If device variables for CJ-series Units are assigned to setup data in the DM Area, the initial values that are set for the variables are used. • If user-defined variables are assigned, the initial values of the variables are used.	The previous values are retained.

*1 Values other than 16#00 may be used depending on the data type. For details, refer to *When the Initial Value Specification Is Left Blank* on page 6-60.

*2 This does not include user-defined variables and device variables for CJ-series Units if they have AT specifications.

*3 Retain condition: Indicates that all of the following conditions are met both before and after the download.

- The variable name is the same.
- The data type name and data type size are the same.
- The Retain attribute is set to retain the value of the variable.

Retain attribute of variable	Type of variable	When a major fault level Controller error occurs	During online editing	
			Variable added to a POU for online editing.	Variable in a POU for online editing.
Non-retain	User-defined variables and device variables	• If initial values are set, the variables change to the initial values. • If initial values are not set, the variables change to 16#00.	• If initial values are set, the variables change to the initial values. • If initial values are not set, the variables change to 16#00.	No change
	CIO and Work memory areas for CJ-series Units	16#00	No change	
Retain	User-defined variables and device variables	No change (retains value before error).	• If initial values are set, the variables change to the initial values. • If initial values are not set, the variables change to 16#00.	
	Holding, DM, and EM memory areas for CJ-series Units		No change	
Others	Forced refreshing status	Cleared.	No change	

Precautions for Correct Use

You can use CJ-series Units and memory used for CJ-series Units only with NJ-series CPU Units.

6-3-10 Function Block Instances

Function block instances are added to and displayed in the local variable table as a data type.

Additional Information

A function block instance is treated as a local variable of the program in which the instance is created. As such, the instance is added to and displayed in the local variable table of the program. You cannot treat these instances as global variables.

6-3-11 Monitoring Variable Values

You can monitor the value of variables from a Watch Tab Page on the Sysmac Studio.

1. Select ***Watch Tab Page*** from the View Menu. The Watch Tab Page is displayed.
2. Establish an online connection with the Controller and register the variables in one of the following ways.

   (1) Enter the variable in the name cell in the Watch Tab Page.

   (2) Drag variables to the Watch Tab Page from an editor or variable table.

   The present values of the variables are displayed.

6-3-12 Restrictions on Variable Names and Other Program-related Names

The following is a list of restrictions on program-related names.

Character Restrictions

<table>
<tr><th>Program-related name</th><th>Applicable characters</th><th>Reserved words</th><th>Multibyte character compatibil-ity</th><th>Case sensitiv-ity</th><th>Maximum size (not including NULL)</th><th>Charac-ter encoding</th></tr>
<tr><td>Variable name (including POU instance names)</td><td rowspan="13">● Usable characters
• 0 to 9, A to Z, and a to z
• Single-byte kana
• _ (underlines)
• Multibyte characters (e.g., Japa-nese)
Refer to Reserved Words below for a list of the reserved words.
● Characters that cannot be used together
• A text string that starts with a number (0 to 9)
• Strings that start with “P_”
• A text string that starts in an underline (_) character
• A text string that contains more than one underline (_) character
• A text string that ends in an underline (_) character
• Any text string that consists of an identifier and has a prefix or postfix which contains more than one extended empty space character (i.e., multi-byte spaces or any other empty Uni-code space characters)</td><td rowspan="13">Refer to Reserved Words below.</td><td rowspan="13">Supported.</td><td rowspan="9">Not case sensitive.</td><td rowspan="5">127 bytes</td><td rowspan="13">UTF-8*</td></tr>
<tr><td>POU definition names</td></tr>
<tr><td>Data type</td></tr>
<tr><td>Structure member names and union member names</td></tr>
<tr><td>Enumerators</td></tr>
<tr><td>Task names</td><td>63 bytes</td></tr>
<tr><td>Namespaces</td><td>93 bytes</td></tr>
<tr><td>Full paths of vari-able names</td><td>Network vari-able: 255 bytes
Other: 511 bytes</td></tr>
<tr><td>Device names</td><td rowspan="5">127 bytes</td></tr>
<tr><td>Section names</td><td>Case sen-sitive.</td></tr>
<tr><td>Axis names</td><td rowspan="3">Not case sensitive.</td></tr>
<tr><td>Axes group names</td></tr>
<tr><td>Cam table names</td></tr>
</table>

* For UTF-8, single-byte alphanumeric characters each use 1 byte. Multibyte characters each use more than 1 byte. Japanese characters require approximately 3 bytes.

Reserved Words

An error is detected during the program check for the following names.

- A name that is the same as any of the instructions that are described in *NJ/NX-series Instructions Reference Manual* (Cat. No. W502)
- A name that is the same as any of the instructions that are described in *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508)
- Words that are reserved by the system

Names That Must Be Unique

The following names must be unique. An error is detected during the program check if they are not.

- Global variable names in the same CPU Unit
- Variable names in the same POU
- Section names in the same POU
- Member names in the same union or structure
- Enumerators in the same enumeration
- Local variable names and global variable names
- POU names and data type names
- Data type names and variable names
- Enumerators of an enumeration and enumerators of another enumeration
- Enumerators and variable names

6-4 Constants (Literals)

This section describes constants in detail.

6-4-1 Constants

The value of a variable changes depending on the data that is assigned to that variable. The value of a constant never changes.

Unlike variables, constants are not stored in memory. You can use constants in the algorithm of a POU without the need to declare them.

In the NJ/NX-series Controllers, constants have a data type in the same way as variables,

6-4-2 Notation for Different Data Types

This section gives the notation for constants with different data types. A building error will occur if you use any other notation for a constant.

Boolean Data

“BOOL” is used as the data type name for Boolean data. You can use the following values: 1, 0, TRUE, and FALSE. The meanings of the notations are given in the following table.

Notation	Meaning
TRUE or FALSE	All of the following are equivalent: TRUE, BOOL#1, BOOL#TRUE, and 1. All of the following are equivalent: FALSE, BOOL#0, BOOL#FALSE, and 0.
BOOL#1 or BOOL#0	
BOOL#TRUE or BOOL#FALSE	
1 or 0*1	

*1 Sysmac Studio version 1.03 or higher is required to use 1 and 0. A building error will occur if you use 1 or 0 on Sysmac Studio version 1.02 or lower.

Bit Strings

You can use any of the following data type names for bit string data: BYTE, WORD, DWORD, and LWORD. You can use any of the following bases: 2, 8, 10, and 16. The notations and notation examples are given in the following table.

Notation	Notation example
{data_type_name}#{base}#{numeric_value}	WORD#16#0064
*{base}#{numeric_value}**1	16#0064
*{numeric_value}**1, *2	100

*1 Sysmac Studio version 1.03 or higher is required to omit the data type name. A building error will occur if you omit the data type name on Sysmac Studio version 1.02 or lower.

*2 A base of 10 (i.e., a decimal number) is assumed.

Version Information

Sysmac Studio Version 1.03 or higher is required to use base 10 for bit string data. A building error will occur if you use base 10 on Sysmac Studio version 1.02 or lower.

Integer Data

You can use any of the following data type names for integer data: SINT, USINT, INT, UINT, DINT, UDINT, LINT, and ULINT. You can use any of the following bases: 2, 8, 10, and 16.

The notations and notation examples are given in the following table.

Notation	Notation example
{data_type_name}#{base}#{numeric_value}	INT#10#–1
{base}#{numeric_value}[*1]	10#–1
{data_type_name}#{numeric_value}[*2]	INT#–1
{numeric_value}[*1, *2]	–1

*1 Sysmac Studio version 1.03 or higher is required to omit the data type name. A building error will occur if you omit the data type name on Sysmac Studio version 1.02 or lower.

*2 A base of 10 (i.e., a decimal number) is assumed.

Real Numbers

You can use any of the following data type names for real number data: REAL and LREAL. You can use only base 10 for real number data. The notations and notation examples are given in the following table.

Notation	Notation example
{data_type_name}#{base}#{numeric_value}	LREAL#10#–3.14
{base}#{numeric_value}[*1]	10#–3.14
{data_type_name}#{numeric_value}[*2]	LREAL#–3.14
{numeric_value}[*1, *2]	–3.14

*1 Sysmac Studio version 1.03 or higher is required to omit the data type name. A building error will occur if you omit the data type name on Sysmac Studio version 1.02 or lower.

*2 A base of 10 (i.e., a decimal number) is assumed.

Durations

You can use any of the following data type names for durations: TIME and T. The notations and notation examples are given in the following table.

Notation	Notation example
TIME#*{day}*d*{hour}*h*{minutes}*m*{seconds}*s*{milliseconds}*ms	TIME#61m5s
T#*{day}*d*{hour}*h*{minutes}*m*{seconds}*s*{milliseconds}*ms	T#61m5s

The following rules apply to duration data constants.

- It is not necessary to give all of the following: days, hours, minutes, seconds, and milliseconds. You must give at least one of them.
- You can use decimal points, such as in *TIME#12d3.5h*.
- You can give times that exceed the valid time ranges. For example, *T#-61m5s* expresses the same duration as *T#-1h1m5s*.
- All numeric values are interpreted as decimal values. A building error will occur if any number that is not a decimal number is used.
- You can change the order of the duration units. For example, *T#1h2d* expresses the same duration as *T#2d1h*.

Dates

You can use any of the following data type names for date data: DATE and D. The notations and notation examples are given in the following table.

Notation	Notation example
DATE#*{year}-{month}-{day}*	DATE#2010-1-10
D#*{year}-{month}-{day}*	D#2010-1-10

The following rules apply to date data constants.

- You can add one or more zeroes to the beginning of the year, month, or day. For example, *DATE#2010-01-10* expresses the same date as *D#2010-1-10*.
- A building error will occur if a valid date range is exceeded. For example, *D#2010-01-35* causes an error.
- All numeric values are interpreted as decimal values. A building error will occur if any number that is not a decimal number is used.

Times of Day

You can use any of the following data type names for time of day data: TIME_OF_DAY and TOD. The notations and notation examples are given in the following table.

Notation	Notation example
TIME_OF_DAY#*{hour}:{minutes}:{seconds}*	TIME_OF_DAY#23:59:59.999999999
TOD#*{hour}:{minutes}:{seconds}*	TOD#23:59:59.999999999

The following rules apply to time of day data constants.

- You can add one or more zeroes to the beginning of the hour, minutes, or seconds. For example, *TOD#23:01:01* expresses the same time of day as *TOD#23:1:1*.
- A building error will occur if a valid time range is exceeded. For example, *TOD#24:00:00* causes an error.
- All numeric values are interpreted as decimal values. A building error will occur if any number that is not a decimal number is used.

Dates and Times

You can use any of the following data type names for date and time data: DATE_AND_TIME and DT. The notations and notation examples are given in the following table.

Notation	Notation example
DATE_AND_TIME#*{year}-{month}-{day}:{hour}:{minutes}:{seconds}*	DATE_AND_TIME#2010-10-10-23:59:59.123
DT#*{year}-{month}-{day}:{hour}:{minutes}:{seconds}*	DT#2010-10-10-23:59:59.123

The following rules apply to date and time data constants.

- You can add one or more zeroes to the beginning of the year, month, day, hour, minutes, or seconds. For example, *DT#2010-01-10-23:01:01* expresses the same date and time as *DT#2010-1-10-23:1:1*.
- A building error will occur if a valid date and time range is exceeded. For example, *DT#2010-01-35-00:00:00* or *DT#2010-01-30-24:00:00* causes an error.
- All numeric values are interpreted as decimal values. A building error will occur if any number that is not a decimal number is used.

Text Strings

To give text string data, enclose the text string in single-byte single quotation marks ('). You can also use “STRING” as the data type name. The notations and notation examples are given in the following table.

Notation	Notation example
'{String}'	'This is a string'
STRING#'*{String}*'[*1]	STRING#'This is a string'

*1 Sysmac Studio version 1.08 or higher is required to use “STRING.”

The following rules apply to text string data constants.

- You can also specify a string with 0 characters. To do so, the notation is ”.
- As in the following example, a building error will occur if you specify any strings that span across multiple lines.
  strVar := 'ABC
  DEF'
- If you want to insert tabs, line break codes, or other special characters, you can use a dollar sign ($) as an escape character before them. The escape character names and meanings are given in the following table.

Escape character	Name	Meaning
$$	Single-byte dollar sign	Single-byte dollar sign ($: character code 0x24)
$'	Single-byte single quotation mark	Single-byte single quotation mark (': character code 0x27)
$"	Single-byte double quotation mark	Single-byte double quotation mark (”: character code 0x22)
$L or $l	Line feed	Moves the cursor to the next line. LF control character (line feed: character code 0x0A)
$N or $n	New line	Moves the cursor to the next line. NL control character (new line: character code 0x0A)
$P or $p	Form feed	Moves the cursor to the next page. FF control character (form feed: character code 0x0C)
$R or $r	Carriage return	Moves the cursor to the start of the line. CR control character (carriage return: character code 0x0D)
$T or $t	Horizontal tab	Indicates a tab. Tab character (character code 0x09)
$*(character_code)*	Direct character code specification	Specify the character code with two hexadecimal digits. The range of the character codes is 00 to FF. For example, “$L” is the same as “$0A”. For characters of two bytes or more, add $ for each byte.

- You can also directly designate character codes. To do so, add $ to the front of the character code. Give the character code with two hexadecimal digits. For example, the character code for a line break ($L) is 0x0A, so $0A is given.
- If you designate the character codes directly for characters of two bytes or more, add $ for each byte.

Enumerated Data Types

For enumerated data, the enumeration data type name and enumerator are given. The notations and notation examples are given in the following table.

Notation	Notation example
{enumeration_data_type_name}#{enumerator}	_eDAYOFWEEK#_WED

Additional Information

To pass an enumerator to a function or function block for which the parameter specifies an enumerator, you can omit the enumeration data type name and give only the enumerator.

For example, the *_eBCD_FORMAT* enumeration is specified for the *Format* input variable in the BinToBCDs instruction. Therefore, you can give either the enumeration data type name and enumerator as *_eBCD_FORMAT#_BCD0* or omit the enumeration data type name and give only *_BCD0*.

6-5 Programming Languages

This section describes the programming languages in detail. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details on entering programs with the Sysmac Studio.

6-5-1 Programming Languages

The languages used to express the algorithms in a POU (program, function, or function block) are called the programming languages.

There are two different programming languages that you can use for an NJ/NX-series Controller: ladder diagram language (LD) and ST (structured text) language.

6-5-2 Ladder Diagram Language

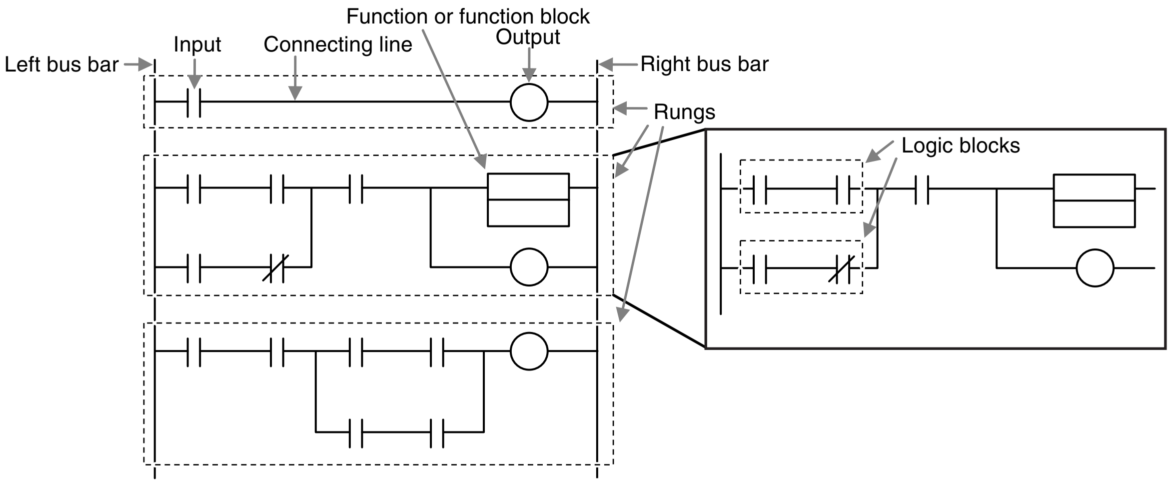
The ladder diagram language (LD) is a graphical programming language that is written in a form that appears similar to electrical circuits. Each object for processing, including functions and function blocks, is represented as a diagram. Those objects are connected together with lines to build the algorithm. Algorithms that are written in the ladder diagram language are called ladder diagrams.

General Structure of the Ladder Diagram Language

A ladder diagram consists of left and right bus bars, connecting lines, ladder diagram structure elements (e.g., inputs and outputs), functions, and function blocks.*

* Only Jump instructions and Label instructions are expressed with symbols that indicate the jumps and labels.

Algorithms are made of multiple rungs connected together. A rung is a connection of all configuration elements between the left bus bar and the right bus bar. A program rung consists of logic blocks that begin with an LD/LD NOT instruction that indicates a logical start.



- **Bus Bars**

The vertical lines on the left and right sides of a ladder diagram are called the bus bars. These bus bars always have a status of either TRUE or FALSE. If you think of the ladder diagram as an electrical circuit, these states represent the flow of current through the circuit. When a POU that is written as a ladder diagram is executed, the value of the left bus bar changes to TRUE. As a result, all inputs and other configuration elements connected to the left bus bar also become TRUE. Execution progresses as elements to the right are also changed to TRUE based on the operation of these configuration elements. This cascade of the TRUE state is called the "power flow." The left bus bar is the source of this power flow.

- **Connecting line**

The straight horizontal lines that connect the bus bar and the configuration elements are called connecting lines. Connecting lines can be either TRUE or FALSE and can transfer the power flow from the left to the right.

- **Inputs**

Inputs are placed along the connecting line to receive the power flow and operate accordingly. There are several different types of inputs and, depending on their specifications, they will either transfer the power flow from the left to the right or prevent the power flow from passing through. When an input transfers the power flow to the right, the connecting line to the right of the input will become TRUE. If the power flow is inhibited, the connecting line to the right of the input will remain FLASE. For detailed specifications on inputs, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

- **Output**

Outputs are placed along the connecting line to receive the power flow and operate accordingly. An output writes the TRUE or FALSE value to a variable. There are different types of outputs. For detailed specifications on outputs, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

- **Functions and Function Blocks**

Functions and function blocks are placed along the connecting line to receive the power flow and operate accordingly. For detailed instruction specifications, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

Order of Execution for Ladder Diagrams

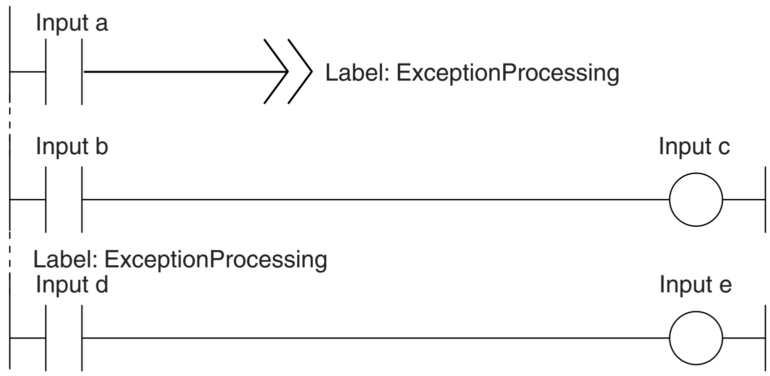
Inputs, outputs, functions, and function blocks are executed when they receive the power flow. The order of execution for a ladder diagram is from top to bottom. Elements at the same level are executed from left to right.

Ladder Diagram Completion

A ladder diagram is executed in order from top to bottom. When the execution reaches the very bottom, the process is completed. However, the process will also end if an END or RETURN instruction is encountered at any point during the process. No processes after those instructions are executed.

Controlling Execution of Ladder Diagrams

Ladder diagrams are generally executed from top to bottom, but you can use execution control instructions to change the execution order. In the following example, when the value of program input a changes to TRUE, execution will move to the point labeled ‘ExceptionProcessing.’



Connecting Functions and Function Blocks in a Ladder Diagram

● Connection Configurations

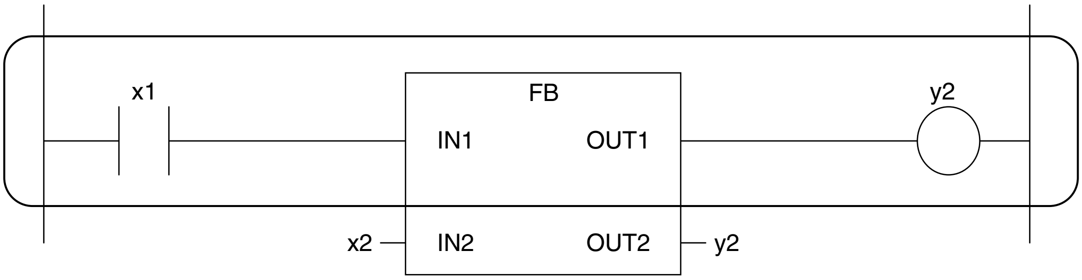
You use the following two types of connections for functions or function blocks.

1) Power Flow Input and Output

In a ladder diagram, the line that connects an input variable of a function or function block and the left bus bar indicates a BOOL input and the line that connects an output variable to the right bus bar indicates a BOOL output.

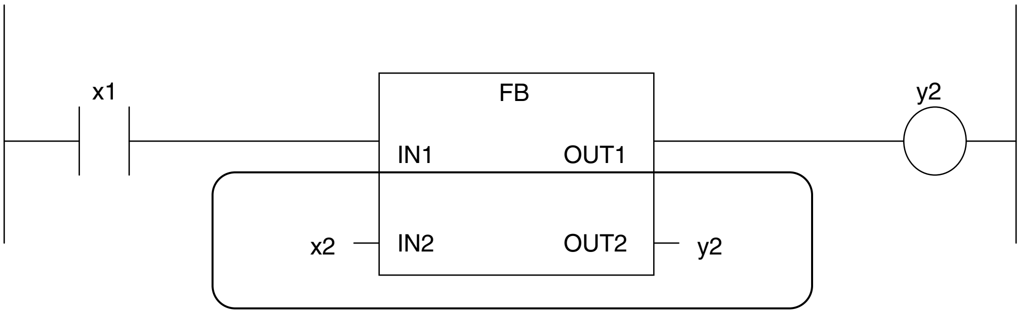
Example:

Inputs are connected in the power flow that connects to the left bus bar. Outputs are connected in the power flow that connects to the right bus bar.



2) Parameter Inputs and Parameter Outputs

In a ladder diagram, parameter inputs and outputs are specified when the input and output variables of a function or function block are not connected to the left and right bus bars.



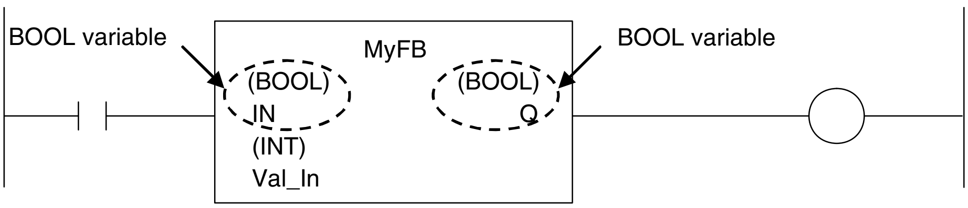
As shown below, you can specify either variables or constants for input and output parameters.

Function/function block variables	Input parameters	Output parameters
Input variables	You can specify variables or constants.	---
Output variables	---	You can specify only variables.
In-out variables	You can specify only variables.	You can specify only variables.

● Number of BOOL Variables

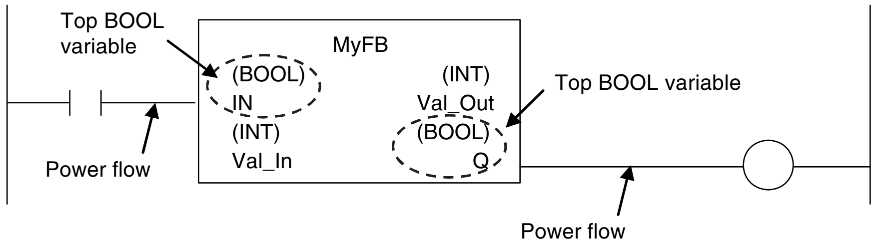
At least one BOOL variable each is required for the input and the output (such as EN and ENO) of a function or function block.

Example:

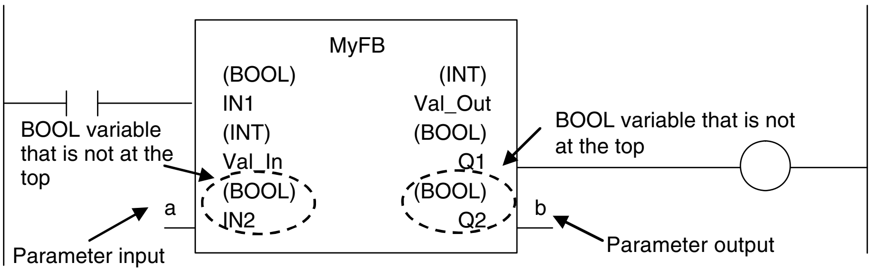


● Connections Based on the BOOL Variable Positions

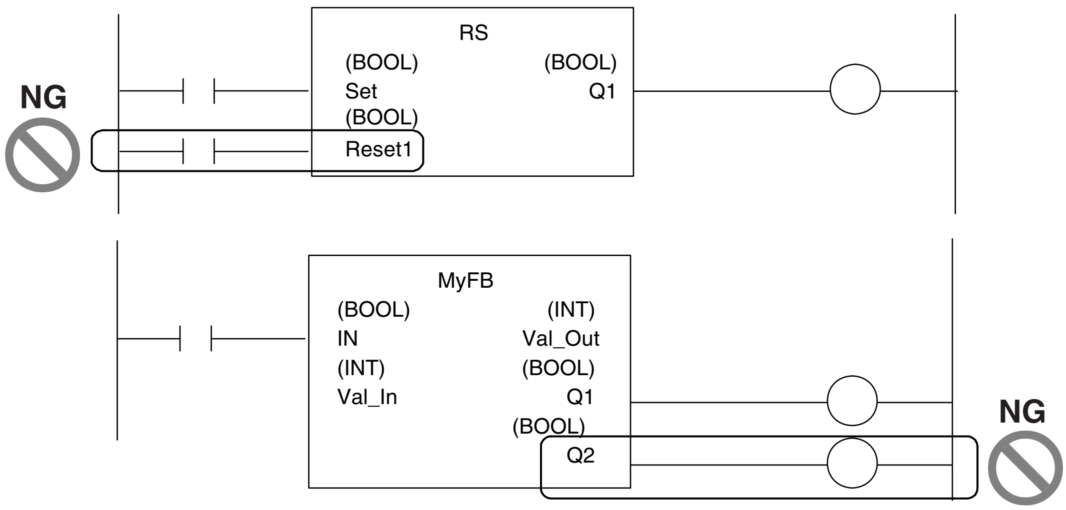
The top BOOL variables are connected to the left and right bus bars. In other words, they become the power flow input and power flow output.



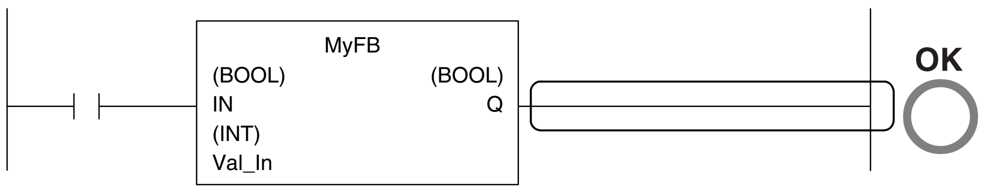
There is only one power flow input and one power flow output for each function or function block. All other BOOL variables that are not at the top are for parameter inputs and parameter outputs.



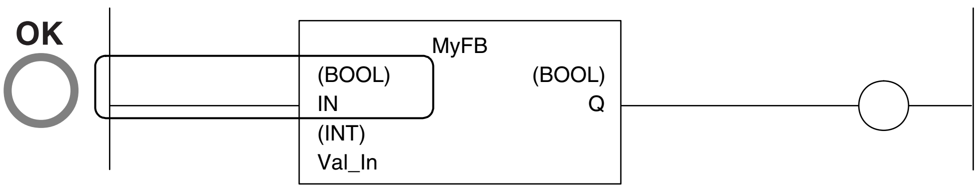
You cannot connect multiple BOOL variables to the left bus bar or the right bus bar as shown below.



You do not have to connect an OUT instruction to the right bus bar. You can connect the function or function block directly.



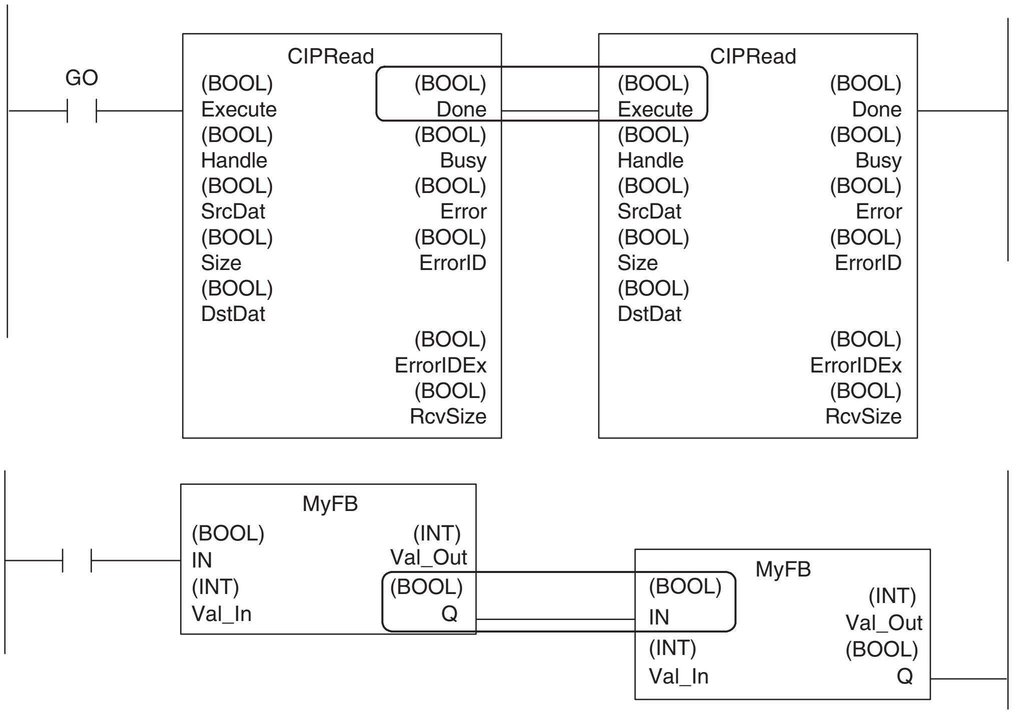
A LD instruction is not necessarily required. You can also connect directly to the left bus bar.



● Cascade Connections

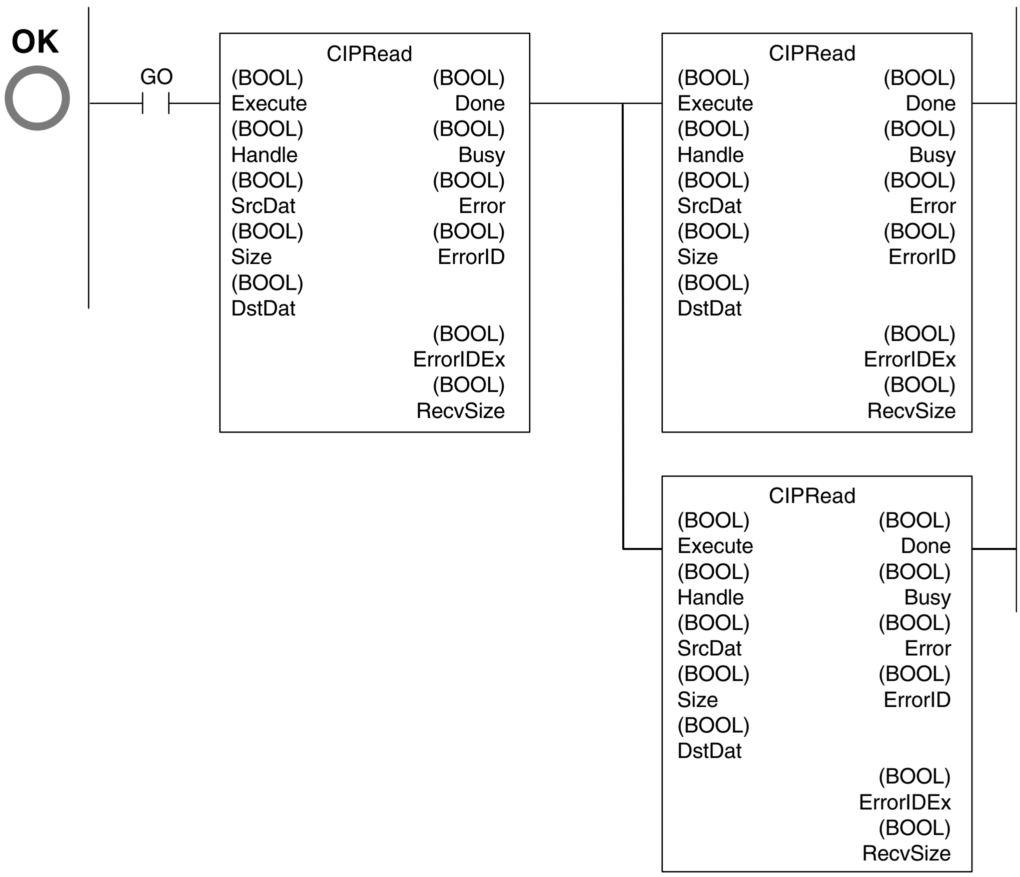
Cascade connections in which the output of a function or function block is connected to the input of another function or function block are allowed only for power flow outputs and inputs.

Example:



You can branch the power flow output.

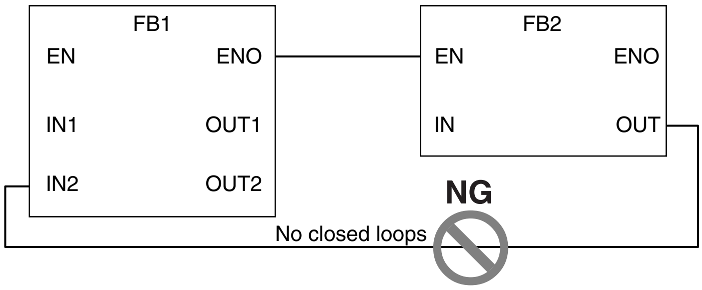
Example:



Restriction

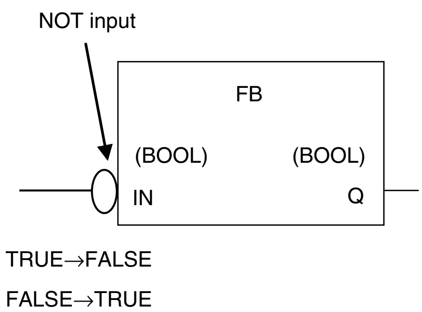
- You cannot create closed loops or intersect connecting lines.

Example:



● Reversing Inputs

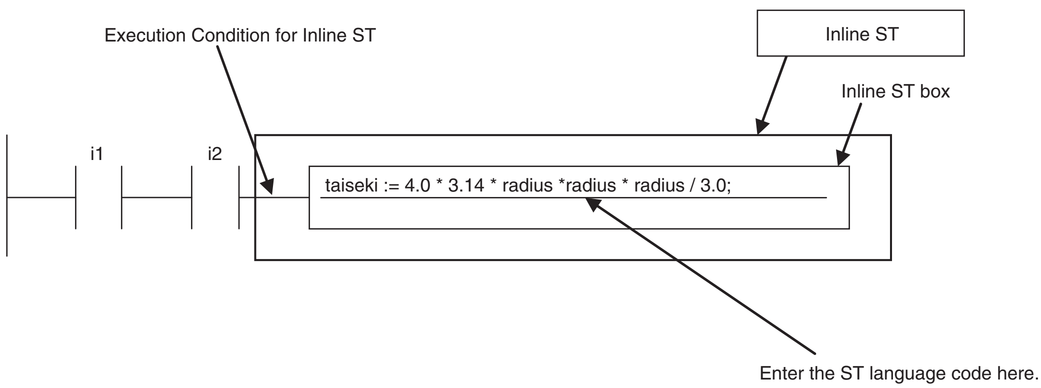
You can reverse the value of a BOOL input variable when you input it to an instruction.



Inline ST

● Introduction

Inline ST is a ladder diagram programming element in which you can write ST language code in a box called an inline ST box (a blank text input area) within a ladder diagram. This allows you to easily code numeric data processing and text string processing within ladder diagrams. The connecting line to an inline ST box becomes its execution condition. The ST code inside of the box is executed based on that connecting line. Refer to the following figure.



Inline ST is treated as a rung element in a ladder diagram. Therefore, unlike functions and function blocks, they have no input, output, or in-out variables.

● Restrictions for Inline ST

You can write ST language code in inline ST boxes.

Execution Conditions for Inline ST

The execution conditions for inline ST are shown in the following table.

Status	Operation
TRUE execution condition	Operation follows the execution condition. You can use the execution condition at any point in the power flow (e.g., you can connect the inline ST directly to the left bus bar). To specify a change to TRUE or a change to FALSE, specify it for an input in the execution condition.
FALSE execution condition	Nothing is done.
Resetting in a master control region	Nothing is done.

Scope of Variables in Inline ST

The scope of variables that you can access from inline ST is the same as the POU of the ladder diagram that contains the inline ST.

Restrictions for Inline ST

Item	Description
Number of inline ST boxes per rung	1

6-5-3 Structured Text Language

The ST (structured text) language is a high-level language code for industrial controls (mainly PLCs) defined by the IEC 61131-3 standard. The standard control statements, operators, and functions make the ST language ideal for mathematical processing that is difficult to write in ladder diagrams. The features of ST are described below.

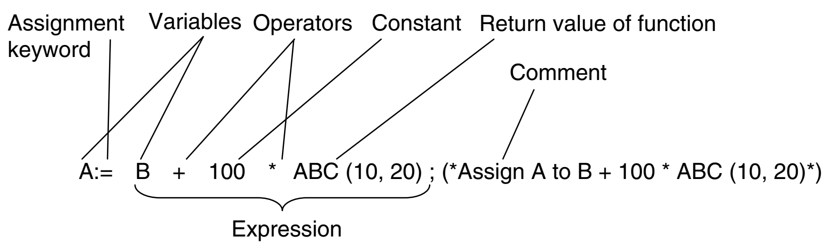
- Loop constructs and control constructs such as IF THEN ELSE are provided.
- You can write programs like high-level languages such as C, and you can include comments to make the program easy to read.

```

// Determine TableNo
FOR i:=0 TO ItemNum DO

    IF (MinNo[i] <= ItemBox[i]) AND (ItemBox[i] <= MaxNo[i]) THEN    // Normal
        TableNo[i] := ItemBox[i];
        RangeOK[i] := TRUE;

    ELSIF (ItemBox[i] > MaxNo[i]) THEN                               // Upper
        TableNo[i] := MaxNo[i];
        RangeOK[i] := FALSE;

    ELSE                                                             // Lower
        TableNo[i] := MinNo[i];
        RangeOK[i] := FALSE;

    END_IF;

END_FOR;
```

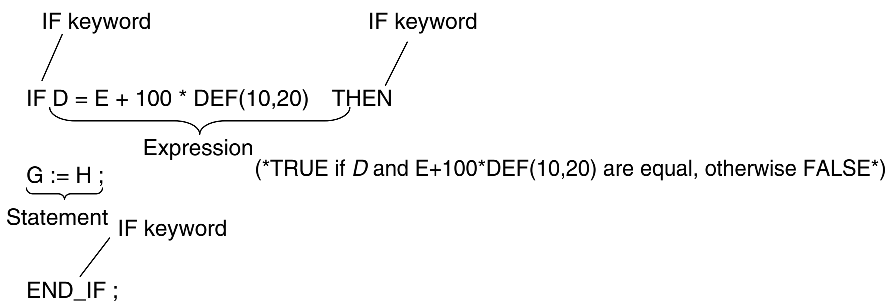
Structure of ST

ST code consists of one or more statements. One statement is the equivalent of one process. Statements are executed from top to bottom, one line at a time, until the process is completed. Statements are made up of keywords and expressions. A keyword is a symbol or string that expresses assignment or execution control. An expression is a code that calculates a value from variables, constants, function return values, and/or a combination of those, along with various operators. A statement represents a process that completes by itself. Expressions form a statement by using a combination of values and keywords.

Example of an Assignment Statement:



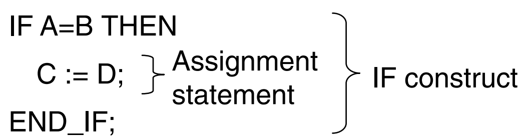
Example of an IF Construct:



ST Language Expressions

● Statement Separators

- Statements must end with a single-byte semicolon (;). Statements are not considered complete with only a carriage return at the end. This allows you to write long statements across multiple lines.
- One statement must end with one single-byte semicolon (;). In the following example, the IF construct contains a single assignment statement. Each statement must be ended with a single-byte semicolon (;).



● Comment

- You can write comments in your program to make the code easier to understand.
- Statements written as comments are not executed.

- The two methods to insert comments are described below.

Comment notation	Examples	Remarks
Enclose the comment in single-byte parenthesis and asterisks, for example, "(*This is a comment*)".	(* Commenting out multiple lines IF ErrCode = 3 THEN Value := 1000; END_IF; down to here. *)	This type of comment can span over multiple lines. Comments cannot be nested.
Begin the comment with two forward slashes (//) and end it with a carriage return.	// Comment // A := SIN(X)^2;	You can comment out only single lines.

● Spaces, Carriage Returns, and Tabs

- You can place any number of spaces, carriage returns, and tabs in your code at any location. This allows you to add spaces or tabs before statements and carriage returns between operators/keywords and expressions in order to make your code easier to read.
- Always enter a token separator, such as a space, carriage return, or tab, between operators/keywords and variables.

Example: The square boxes indicate where you must insert a token separator, such as a space, carriage return, or tab.

```
IF ■ A>0 ■ THEN ■ X:=10;
ELSE ■
X:=0;
END_IF;
```

● Lowercase/Uppercase, Single-byte/double-byte Characters

- Operators, keywords, and variable names are not case sensitive.
- Operators, keywords, and variable names must always be in single-byte characters. A syntax error will occur if you input double-byte characters.

● Variables and Prohibited Characters

Refer to *6-3-12 Restrictions on Variable Names and Other Program-related Names* for restrictions on variable names.

● Text Strings

Refer to *6-3-12 Restrictions on Variable Names and Other Program-related Names* for restrictions on text strings.

ST Keywords and Operators

● Statement Keywords

Keyword	Meaning	Example
:=	Assignment	d := 10;
	Calling functions and function blocks	FBname(para1 := 10, para2 := 20); Refer to *Function Block Calls* on page 6-104.
RETURN	Return	
IF	If	IF d < e THEN f := 1; ELSIF d = e THEN f :=2; ELSE f := 3; END_IF;

Keyword	Meaning	Example
CASE	Case	CASE f OF 1: g :=11; 2: g :=12; ELSE g :=0; END_CASE;
FOR	For	FOR i: = 100 TO 1 BY -1 DO Val[i] := i; END_FOR;
WHILE	While	WHILE Val < MaxVal DO Val := Val + 1; END_WHILE;
REPEAT	Repeat	REPEAT Val := Val + 1; UNTIL(Val > 4) END_REPEAT;
EXIT	Exit the loop.	FOR i := 1 TO 100 DO FOR j := 1 TO 10 DO IF Val[i, j]>100 THEN EXIT; END_IF; END_FOR; END_FOR;
;	Empty statement	Val[i] := i ; (* Empty statement *) WHILE(Var <>0) DO ; (* Empty statement *) END_WHILE;
(* Text *)	Comments	(* Commenting out multiple lines IF MyFun (ErrorCode) = 3 THEN ReturnValue := GetDetail(); END_IF; down to here. *)
//Text	Comment	A := SIN(X) ^ 2 + COS (Y) ^2 + 10; // A := SIN(X) ^ 2 + COS (Y) ^2 + 5;

● Operators

The following table gives the operators and their order of priority.
If operators with different priorities are mixed in one expression, the operators with the highest priorities are executed first. You can use up to 64 operators in one expression.
Example: X:=(1+2)–3*4; In this case, variable X is assigned a value of –9.

1. The operation in parentheses (priority of 1) is performed.
2. The multiplication (priority of 5) is performed.
3. The subtraction (priority of 6) is performed.

(1+2)–3*4 → 3–3*4 → 3–12 → –9

Operation	Operator	Notation example and evaluated value	Priority
Parentheses	()	(1+2)*(3+4) Value: 21	1
Function/function block call		FUN1(FUN2(Var2A, Var2B), Var1B) When function and function block calls are nested, the function or function block at the lower level is called first. In the above example, FUN2 is executed first, and then FUN1 is executed.	2
Sign	+, –	+100 –100	3
NOT	NOT	NOT TRUE Value: FALSE	
Exponent	**	-2**2 Value: 4 A minus sign is given priority over an exponent operator. Therefore, -2**2 in the above example is the same as (-2)**2, so the value is 4. 2**3**2 Value: 64 If there is more than one exponent operator, calculations are performed for them left to right. Therefore, 2**3**2 in the above example is the same as (2**3)**2, so the value is 64.	4
Multiplication	*	100*200 Value: 20,000	5
Division	/	100/200 Value: 0.5	
Remainder	MOD	10 MOD 7 Value: 3 –17 MOD 6 Value: –5 –17 MOD (–6) Value: –5 17 MOD 6 Value: 5 17 MOD (–6) Value: 5	

Operation	Operator	Notation example and evaluated value	Priority
Addition	+	100+200 Value: 300	6
Subtraction	–	100-200 Value: –100	
Comparison	<, >, <=, >=	100<200 If the comparison result is TRUE, the value is set to TRUE. Otherwise, the value is set to FALSE. In the above example, 100 is less than 200, so the value is TRUE.	7
Matches	=	100=200 If the two values match, the value is set to TRUE. Otherwise, the value is set to FALSE. In the above example, 100 does not equal 200, so the value is FALSE.	8
Does not match	< >	100<>200 If the two values do not match, the value is set to TRUE. Otherwise, the value is set to FALSE. In the above example, 100 does not equal 200, so the value is TRUE.	
Logical AND	AND,&	Applies 1-bit AND logic to all bits. The results of 1-bit AND logic are as follows: 0 AND 0 = 0 0 AND 1 = 0 1 AND 0 = 0 1 AND 1 = 1 0101 AND 1100 Value: 0100	9
Logical exclusive OR	XOR	Applies 1-bit exclusive OR logic to all bits. The results of 1-bit exclusive OR logic are as follows: 0 XOR 0 = 0 0 XOR 1 = 1 1 XOR 0 = 1 1 XOR 1 = 0 0101 XOR 1100 Value: 1001	10
Logical OR	OR	Applies 1-bit OR logic to all bits. The results of 1-bit OR logic are as follows: 0 OR 0 = 0 0 OR 1 = 1 1 OR 0 = 1 1 OR 1 = 1 0101 OR 1100 Value: 1101	11

Precautions for Correct Use

The intended operation may not occur if a function is nested under itself. Always separate the functions into different statements as shown below.

Example of incorrect notation: out := MyFunc(In1:=x1, In2:=MyFunc(In1:=x2, In2:=x3));

Example of correct notation:

```
tmp := MyFunc( In1:=x2, In2:=x3 );
out := MyFunc( In1:=x1, In2:=tmp );
```

Precautions for Correct Use

The order of priority for operators is sometimes different for different standards and manufacturers. Special attention is necessary for the priority of exponent operators. We therefore recommend that you use parentheses to ensure that calculations are performed in the intended order.

Example: For X:= -2**3**4; we recommend that you use the following expression:
X:= ((-2)**3)**4;.

Additional Information

Calculations are performed based on the data types. For example, the result of calculations with integer data will be integer data. Therefore, if the expression *A/B* is calculated with INT variables *A* = 3 and *B* = 2, the result would not be 1.5 because all values after the decimal point are truncated. In this case, the expression (A/B)*2 would evaluate to 2 instead of 3.

● Data Types for Operator Operands

If all the operands for an operator have the same data type, any data type given as “Supported” in the following table can be set as operands. However, if an operand with a different data type is set for the operator, an implicit cast is required. Refer to *Implicit Casts* on page 6-110 for details on implicit casting.

Data type	Assignment operator	Argument setting operator	Numeric operators	Modulo-division operator	Power operator	Comparison operators	Equality operators	Logic operators	Positive/negative signs
	:=	:= =>	+ – * /	MOD	**	< <= => >	= <>	NOT AND & OR XOR	+ –
Boolean	OK	OK	---	---	---	---	OK	OK	---
Bit string	OK	OK	---	---	---	---	OK	OK	---
Integer	OK	OK	OK	OK	OK	OK	OK	---	OK
Real number	OK	OK	OK	---	OK	OK	OK	---	OK
Duration	OK	OK	---	---	---	---	---	---	OK
Date	OK	OK	---	---	---	---	---	---	---
Time of day	OK	OK	---	---	---	---	---	---	---
Date and time	OK	OK	---	---	---	---	---	---	---
Text string	OK	OK	---	---	---	---*	---*	---	---
Enumeration	OK	OK	---	---	---	---	OK	---	---
Structure parent	OK	OK	---	---	---	---	---	---	---
Array parent	OK	OK	---	---	---	---	---	---	---

OK: Possible

---: A building error will occur.

* Do not use operators to compare text string variables. Use instructions (such as EQascii) instead.

ST Language Statements

● Assignment

Overview:

This statement assigns the right side (i.e., the value of the expression) to the left side (i.e., the variable).

Reserved Words:

:=

Combination of a colon (:) and an equals sign (=)

Statement Structure:

```
<variable>:=<expression>;
<variable>:=<variable>;
<variable>:=<constant>;
```

Application:

Use this statement to assign a value to a variable. For example, use it to set initial values or to store the results of a calculation.

Description:

This statement assigns (or stores) the *<expression_value>* to the *<variable>*.

Example:

Example 1: The following statement assigns the result of the expression *X*+1 to variable *A*.

```
A:=X+1;
```

Example 2: The following statement assigns the value of variable *B* to variable *A*.

```
A:=B;
```

Example 3: The following statement assigns a value of 10 to variable *A*.

```
A:=10;
```

Precautions:

- Either the source data type must match the destination data type, or the combination of data types must allow implicit casting. A building error will occur if you do not use this notation.
- If the value that is assigned is STRING data, make the size of the destination STRING variable larger than that of the source string. Otherwise, an error will occur.
- For STRING variables, assignment is allowed if the size of left-hand variable is greater than the size of the text string stored in right-hand variable.
  Example:
  Assignment is allowed in the following case.
  - Variable Table:

Variable name	Data type	Size
Var1	STRING	10
Var2	STRING	20

  - User Program:
    Var2 :='ABC';

```
Var1 := Var2;
```

You cannot make assignments to union variables. You must make the assignments to individual members of the unions.

● RETURN

Overview:

The following actions occur depending on where the ST statement is used.

ST

The ST program is ended during operation and the next program is executed.

ST in a Function Inside a Function Block Instance

The function or function block is ended during operation and the next instruction after the calling instruction is executed.

Inline ST

The POU that contains inline ST with a RETURN statement is ended.

Reserved Words:

RETURN

Statement Structure:

```
RETURN;
```

Application:

Use this statement to force the current program, function, or function block to end.

● IF with One Condition

Overview:

The construct executes the specified statement when a condition is met. If the condition is not met, another statement is executed. The following expressions are used to specify whether the condition is met.

TRUE: The condition is met.

FALSE: The condition is not met.

Reserved Words:

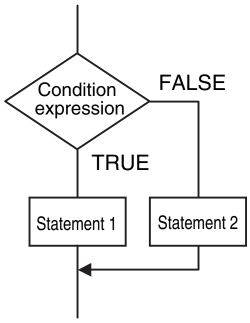
IF, THEN, (ELSE), END_IF

Note You can omit ELSE.

Construct Structure:

```
IF <condition_expression> THEN
  <statement_1>;
ELSE
  <statement_2>;
END_IF;
```

Process Flow Diagram:



Application:

Use this construct to perform one of two processes depending on evaluation of a condition (condition expression).

Description:

If <*condition_expression*> is TRUE, <*statement_1*> is executed.

If <*condition_expression*> is FALSE, <*statement_2*> is executed.

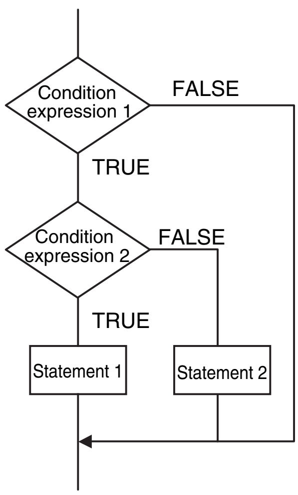
Precautions:

- IF must always be used together with END_IF.
- Write a statement that evaluates to TRUE or FALSE (for example *IF A>10*) or a BOOL variable (for example *IF A*) for the condition expression.
- You can write <*statement_1*> and <*statement_2*> on multiple lines. Separate statements with a semicolon (;).

Example: Another IF Statement before <*statement_1*>

```
IF <condition_expression_1> THEN
    IF <condition_expression_2> THEN
        <statement_1>;
    ELSE
        <statement_2>;
    END_IF;
END_IF;
```

Process Flow Diagram:

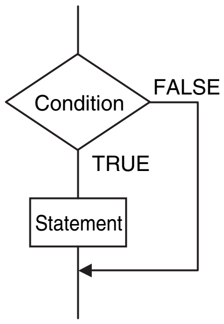


ELSE corresponds to the previous THEN statement, as shown above.

- You can execute more than one statement for both <*statement_1*> and <*statement_2*>. Separate statements with a semicolon (;).

- You can omit the ELSE statement. If it is omitted, nothing is executed when *<condition_expression>* is FALSE.

Process Flow Diagram:



Example:

Example 1: A value of 10 is assigned to variable *X* when the statement *A* > 0 is TRUE. A value of 0 is assigned to variable *X* when the statement A > 0 is FALSE.

```
IF A>0 THEN
    X:=10;
ELSE
    X:=0;
END_IF;
```

Example 2: A value of 10 is assigned to variable *X* and a value of 20 is assigned to variable Y when the statements A > 0 and B > 1 are both TRUE. A value of 0 is assigned to variable *X* and variable *Y* when the statements *A* > 0 and *B* > 1 are both FALSE.

```
IF A>0 AND B>1 THEN
    X:=10;Y:=20;
ELSE
    X:=0;Y:=0;
END_IF;
```

Example 3: A value of 10 is assigned to variable *X* when the BOOL variable *A* is TRUE. A value of 0 is assigned to variable *X* when variable *A* is FALSE.

```
IF A THEN X:=10;
ELSE X:=0;
END_IF;
```

● IF with Multiple Conditions

Overview:

The construct executes the specified statement when a condition is met. If a condition is not met but another condition is met, another statement is executed. If neither condition is met, another statement is executed.

The following expressions are used to specify whether the condition is met.

TRUE: The condition is met.

FALSE: The condition is not met.

Reserved Words:

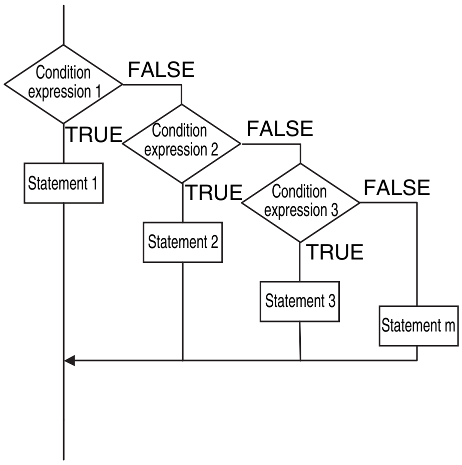
IF, THEN, ELSIF, (ELSE), END_IF

Note You can omit ELSE.

Construct Structure:

```
IF <condition_expression_1> THEN <statement_1>;
    ELSIF <condition_expression_2> THEN <statement_2>;
    ELSIF <condition_expression_3> THEN <statement_3>;
            .
            .
            .
    ELSIF <condition_expression_n> THEN <statement_n>;
ELSE<statement_m>;
END_IF;
```

Process Flow Diagram:



Application:

Use this construct to perform a process depending on evaluation of multiple conditions (condition expressions).

Description:

If *<condition_expression_1>* is TRUE, *<statement_1>* is executed.

If *<condition_expression_1>* is FALSE and *<condition_expression_2>* is TRUE, then *<statement_2>* is executed.

If *<condition_expression_2>* is FALSE and *<condition_expression_3>* is TRUE, then *<statement_3>* is executed.

.
.
.

If *<condition_expression_n>* is TRUE, *<statement_n>* is executed.

If none of the conditions is TRUE, *<statement_m>* is executed.

Precautions:

- IF must always be used together with END_IF.
- Write statements that can be TRUE or FALSE for the condition expressions. Example: IF(A>10) You can also specify BOOL variables (including functions that return a BOOL value) for the condition expressions instead of an actual expression. In that case, when the variable is TRUE, the evaluated result is TRUE and when the variable is FALSE, evaluated result is FALSE.
- You can write any of the statements on multiple lines. Separate statements with a semicolon (;).
- You can omit the ELSE statement. If it is omitted, and none of the conditions produces a match, nothing is done.

Example:

A value of 10 is assigned to variable *X* when the statement *A* > 0 is TRUE.

A value of 1 is assigned to variable *X* when the statement A > 0 is FALSE and statement B = 1 is TRUE.

A value of 2 is assigned to variable *X* when the statement A > 0 is FALSE and statement B = 2 is TRUE.

If none of the conditions is TRUE, a value of 0 is assigned to the variable *X*.

```
IF A>0 THEN X:=10;
    ELSIF B=1 THEN X:=1;
    ELSIF B=2 THEN X:=2;
ELSE X:=0;
END_IF;
```

CASE

Overview:

This construct executes a statement that corresponds to an integer set value that matches the value of an integer expression.

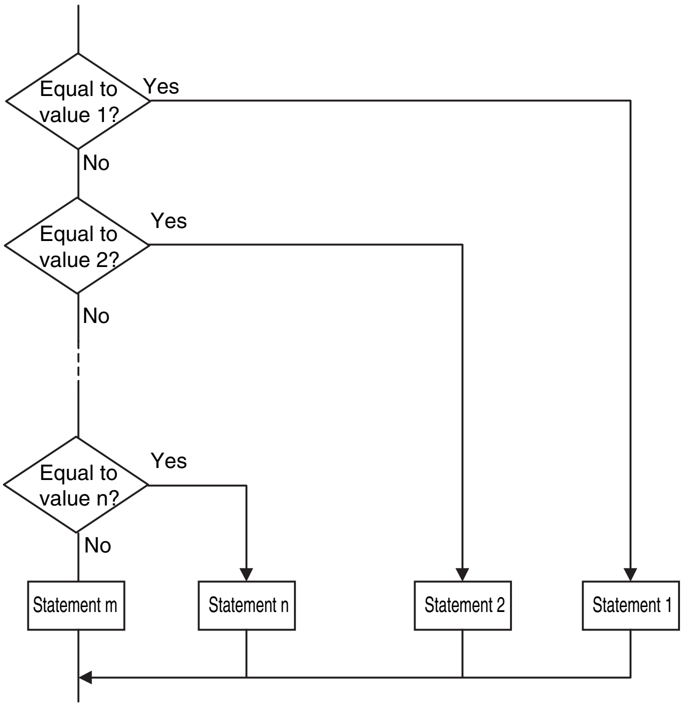
Reserved Words:

CASE

Construct Structure:

```
CASE <integer_expression> OF
    <integer_expression_value_1>:<statement_1>;
    <integer_expression_value_2>:<statement_2>;
            .
            .
            .
    <integer_expression_value_n>:<statement_n>;
ELSE<statement_m>;
END_CASE;
```

Process Flow Diagram:



Application:

Use this construct to perform different actions based on the value of an integer.

Description:

If <*integer_expression*> matches <*integer_expression_value_n*>, <*statement_n*> is executed.
If <*integer_expression*> does not match any of the integer values, <*statement_m*> is executed.

Precautions:

- CASE must always be used together with END_CASE.
- Use one of the following for the <*integer_expression*>:
  - An integer or enumeration variable (example: *abc*)
  - An integer expression (example *abc+def*)
  - A function that returns an integer value (example: *xyz()*)
- You can write any of the statements on multiple lines. Separate statements with a semicolon (;).
- To specify OR logic of multiple integers for <*integer_expression_value_n*>, separate the values with commas. To specify a continuous range of integers, separate the start integer and the end integer with two periods (..).

Example 1: You can specify a condition for a specific integer value, or the same condition for multiple integer values.

```
CASE A OF
    1: X:=1;
    2: X:=2;
    3: X:=3;
ELSE
    X:=0;
END_CASE;
```

1: X:=1;	A value of 1 is assigned to variable *X* when variable *A* is 1.
2: X:=2;	A value of 2 is assigned to variable *X* when variable *A* is 2.
3: X:=3;	A value of 3 is assigned to variable *X* when variable *A* is 3.
X:=0;	If none of the values is matched, a value of 0 is assigned to the variable *X*.

```
CASE A OF
    1: X:=1;
    2,5: X:=2;
    6..10: X:=3;
    11,12,15..20: X:=4;
ELSE
    X:=0;
END_CASE;
```

1: X:=1;	A value of 1 is assigned to variable *X* when variable *A* is 1.
2,5: X:=2;	A value of 2 is assigned to variable *X* when variable *A* is 2 or 5.
6..10: X:=3;	A value of 3 is assigned to variable *X* when variable *A* is between 6 and 10.
11,12,15..20: X:=4;	A value of 4 is assigned to variable *X* when variable *A* is 11, 12, or between 15 and 20.
X:=0;	If none of the values is matched, a value of 0 is assigned to the variable *X*.

Example 2: You can give an integer variable, integer expression, integer function return value, enumeration variable, or enumeration function return value for the <*integer_expression*>. An example is shown below.

- Example for an Integer Enumeration Variable

```
CASE ColorVar OF
    RED:
        X := 0;
    BLUE:
        X := 1;
    ELSE
        X := 2;
END_CASE;
```

* Example for an Integer Expression

```
CASE (a1 + a2) OF
    0:
        X := 0;
    1:
        X := 1;
    ELSE
        X := 2;
END_CASE;
```

* Example of an Integer Enumeration Function Return Value

```
CASE FUN( ) OF
    0:  ------------------------------  Branches depending on the return value of FUN( ).
        X := 10;
    1:
        X := 11;
    ELSE
        X := 12;
END_CASE;
```

Data Types That You Can Use in CASE Constructs

Classification	Data type		<integer_expression>
Basic data types	Integers		Supported.
	Boolean, bit string, real, duration, date, time of day, date and time, or text string data		Not supported.
Data type specifications	Array specifications	Arrays	Not supported.
		Elements	Supported for integers and enumerations only.
Derivative data types	Structures	Structures	Not supported.
		Members	Supported for integers and enumerations only.
	Unions	Unions	Not supported.
		Members	Supported for integers and enumerations only.
	Enumerations		Supported.

● FOR

Overview:

This construct repeatedly executes the same statements until a variable (called the FOR variable) changes from one value to another value.
The following expressions are used to specify whether the condition is met.

TRUE: The condition is met.

FALSE: The condition is not met.

Reserved Words:

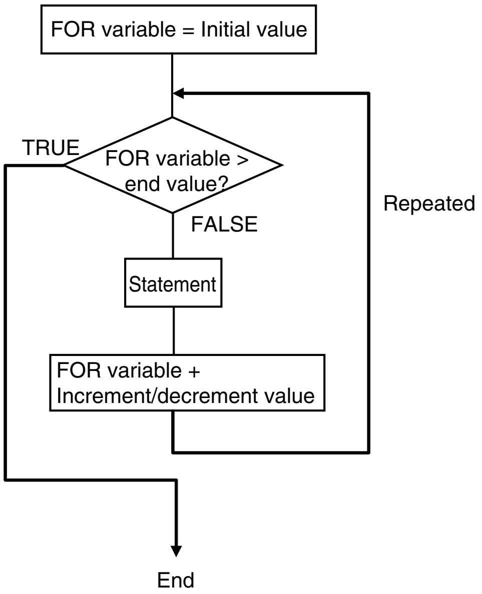
FOR, TO, (BY), DO, END_FOR

Note You can omit BY.

Construct Structure:

```
FOR <FOR_variable>:= <initial_value> TO <end_value> BY <increment/decrement> DO
    <statement>;
END_FOR;
```

Process Flow Diagram:



Application:

Use this construct when you know in advance how many times you want to repeat a process.

This type of repeat construct is particularly effective to specify each element of an array variable based on the value of a FOR variable.

Description:

A decision is made based on the evaluation of <*initial_value*>, <*end_value*>, and <*increment/decrement*>.
When <*FOR_variable*> is <*initial_value*>, <*statement*> is executed.
After execution, the value of <*increment/decrement*> is added to <*FOR_variable*> and <*statement*> is executed again if <*FOR_variable*> is less than the value of the <*end_value*>.
After execution, the value of <*increment/decrement*> is added to <*FOR_variable*> and <*statement*> is executed again if <*FOR_variable*> is less than the value of the <*end_value*>.
This process is repeated.
The loop ends when <*FOR_variable*> > <*end_value*>.
If <*increment/decrement*> is negative, the directions of the comparison symbols in the above statements are reversed.

Precautions:

- If the FOR variable is signed, <*increment/decrement*> can be a negative number.
- FOR must always be used together with END_FOR.
- The FOR variable becomes the end value plus increment/decrement after execution of the process is completed for the end value. This ends the FOR construct.

  Example: When the FOR construct is completed in the following ST statements, the value of *i* is 101.

```
FOR i:=0 TO 100 DO
    X[i]:=0;
END_FOR;
// Here, i is 101.
```

- Do not write code that directly modifies the FOR variable inside the FOR construct. Unintended operation may result.

  Example:

```
FOR i:=0 TO 100 BY 1 DO
    X[i]:=0;
    i:=i+INT#5;
END_FOR;
```

- You can write any of the statements on multiple lines. Separate statements with a semicolon (;).
- You can omit BY*<increment/decrement>*. If it is omitted, the statement is executed with an increment value of 1.
- You can specify an integer (SINT, INT, DINT, LINT, USINT, UINT, UDINT, or ULINT) variable or integer value for the *<initial_value>*,*<end_value>*, and *<increment/decrement>*. You can also specify a function that returns an integer value.

  Example 1: A value of 100 is assigned to array variable elements *SP[n]*. The FOR variable is variable n, the initial value is 0, the end value is 50, and the increment is 5.

```
FOR n := 0 TO 50 BY 5 DO
    SP[n] := 100;
END_FOR;
```

  Example 2: The total of elements *DATA[1]* through *DATA[50]* of array variable elements *DATA[n]* is calculated and the result is assigned to the variable *SUM*.

```
IF a THEN
    FOR n := 0 TO 50 BY 1 DO
        DATA[n]:= 1 ;
    END_FOR;

    FOR n := 0 TO 50 BY 1 DO
        SUM:= SUM + DATA[n] ;
    END_FOR;

    a:=FALSE;
END_IF;
```

  Example 3: The maximum and minimum values of elements *DATA[1]* through *DATA[50]* of array variable elements *DATA[n]* are found. The maximum value is assigned to the *MAX variable*, and the minimum value is to the *MIN variable*. The value of *DATA[n]* is from 0 to 1,000.

```
MAX :=0;
MIN :=1000;
FOR n :=1 TO 50 BY 1 DO
    IF DATA[n] > MAX THEN
        MAX :=DATA[n];
    END_IF;
    IF DATA[n] < MAX THEN
        MIN :=DATA[n];
    END_IF;
END_FOR;
```

- If the total execution time of the statements in the FOR construct from when the FOR variable is incremented/decremented from the initial value until it reaches the end value exceeds the task period, a Task Period Exceeded error occurs.
  - When the FOR Variable Cannot Logically Reach the End Value

    Example:

```
FOR i := 0 TO 100 BY 1 DO
    intArray[i] := i;
    i := INT#50;
END_FOR;
```

An infinite loop occurs and results in a Task Period Exceeded error.

Example:

```
FOR i := 0 TO 100 BY 0 DO
    ;
END_FOR;
```

An infinite loop occurs and results in a Task Period Exceeded error.

  - When an Overflow or Underflow Occurs Because the FOR Variable Exceeds the End Value

    Example:

```
FOR i := 0 TO 254 BY 2 DO
    INTArray[i] := i;
END_FOR;
```

Version Information

With the Sysmac Studio version 1.08 or higher, you can specify arithmetic expressions for <end_value> and <increment/decrement>.

However, the evaluation is performed for <end_value> or <increment/decrement> only before the execution of FOR loop operation. The values of <end_value> and <increment/decrement> do not change after the FOR loop operation is started.

For example, in the following case, the value of <end_value> is 10 and <increment/decrement> is 3. Even after the FOR loop operation is started and the values of variable A and C are changed, the value of <end_value> is still 10 and <increment/decrement> is still 3.

```
A := INT#1;
B := INT#2;
C := INT#10;

FOR i := 0 TO C BY A+B DO
    INTArray[i] := i;
    A := B + i;
    C := C + i;
END_FOR;
```

If an arithmetic expressions is specified for <end_value> or <increment/decrement> on the Sysmac Studio version 1.07 or lower, a building error will occur.

Data Types That You Can Use in FOR Constructs

Classification	Data type		*<initial_value>*, *<end_value>*, and *<increment/decrement>**
Basic data types	**Boolean, bit string, real, duration, date, time of day, date and time, or text string data**		Not supported.
	Integers		Supported.
Data type specifications	**Array specifications**	**Arrays**	Not supported.
		Elements	Supported for integers and enumerations only.
Derivative data types	**Structures**	**Structures**	Not supported.
		Members	Supported for integers and enumerations only.
	Unions	**Unions**	Not supported.
		Members	Supported for integers and enumerations only.
	Enumerations		Supported.

* You must use the same data type for the *<FOR_variable>*, *<end_value>* and *<increment/decrement>*. Otherwise, an error occurs when the program is built on the Sysmac Studio.

● WHILE

Overview:

This construct repeatedly executes the specified statements as long as a condition expression is TRUE.

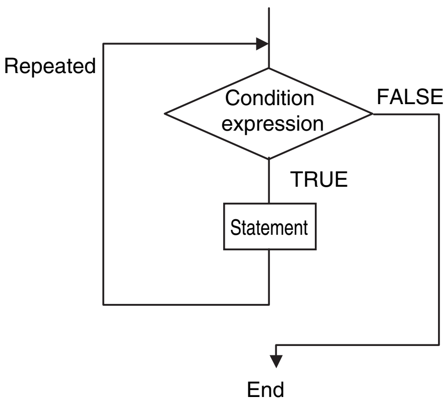
Reserved Words:

WHILE, DO, END_WHILE

Construct Structure:

```
WHILE <condition_expression> DO
    <statement>;
END_WHILE;
```

Process Flow Diagram:



Application:

Use this type of repeat construct when you do not know how many times to repeat a process (i.e., when you do not know how many times based on the condition) and you want to repeat a process for as long as a certain condition is met. You can also use this type of repeat construct to execute a process only when a condition expression is TRUE (pre-evaluation repeat construct).

Description:

The *<condition_expression>* is evaluated before *<statement>* is executed.

If *<condition_expression>* is TRUE, *<statement>* is executed. Then the *<condition_expression>* is evaluated again. This process is repeated.

If the *<condition_expression>* is FALSE, *<statement>* is not executed and the *<condition_expression>* is no longer evaluated.

Precautions:

- WHILE must always be used together with END_WHILE.
- If the *<condition_expression>* is FALSE before *<statement>* is executed, the WHILE construct is exited and *<statement>* is not executed.
- You can write *<statement_1>* and *<statement_2>* on multiple lines. Separate statements with a semicolon (;).
- You can execute more than one statement for *<statement>*. Separate statements with a semicolon (;).
- You can also specify a BOOL variable (including functions that return a BOOL value) for the condition expressions instead of an actual expression.

Example:

Example 1: The first multiple of 7 that exceeds 1,000 is calculated and assigned to variable *A*.

```
A := 0;
WHILE A <= 1000 DO
    A := A+INT#7;
END_WHILE;
```

Example 2: The value of variable *X* is doubled if *X* is less than 3,000 and the value is assigned to array variable element *DATA[1]*. Next, the value of *X* is doubled again and the value is assigned to the array variable element *DATA[2]*. This process is repeated.

```
n := 1;
X := 1;
WHILE X < 3000 DO
    X:= X*INT#10#2;
    DATA[n]:= X;
    n := n+INT#1;
END_WHILE;
```

- If you do not write correct condition expressions, the program execution time increases and may cause a Task Period Exceeded error.

Example:

```
boolVar := TRUE;
WHILE boolVar DO
intVar := intVar + INT#1;
END_WHILE;
```

● REPEAT

The following expressions are used to specify whether the condition is met.

TRUE: The condition is met.

FALSE: The condition is not met.

Overview:

This construct repeatedly executes one or more statements until a condition expression is TRUE.

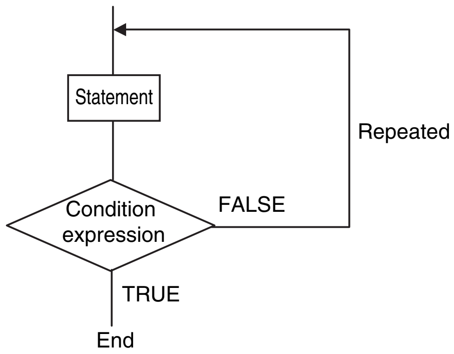
Reserved Words:

REPEAT, UNTIL, END_ REPEAT

Construct Structure:

```
REPEAT
    <statement>;
UNTIL <condition_expression>
END_REPEAT;
```

Process Flow Diagram:



Application:

Use this type of repeat construct when you do not know how many times to repeat a process (i.e., when you do not know how many times based on the condition) and you want to repeat a process for as long as a certain condition is met after processing. Use this type of repeat construct to determine repeat execution based on the result of the execution of a process (post-evaluation repeat construct).

Description:

First, <*statement*> is executed unconditionally. Then the <*condition_expression*> is evaluated.

If <*condition_expression*> is FALSE, <*statement*> is executed.

If <*condition_expression*> is TRUE, <*statement*> is not executed and the REPEAT construct is exited.

Precautions:

- REPEAT must always be used together with END_REPEAT.
- Even if the <*condition_expression*> is TRUE before <*statement*> is executed, <statement> is executed.
  In other words, <*statement*> is always executed at least one time.
- <*statement*> can contain multiple lines of code for the statement. Separate statements with a semicolon (;).
- You can also specify a BOOL variable (including functions that return a BOOL value) for the condition expressions instead of an actual expression.

Example:

Example 1: Numbers from 1 to 10 are added and the values are assigned to the variable *TOTAL*.

```
A := 1;
TOTAL := 0;
REPEAT
    TOTAL := TOTAL + A;
    A := A+INT#1;
UNTIL A>10
END_REPEAT;
```

- If you do not write correct condition expressions, the program execution time increases and may cause a Task Period Exceeded error.

  Example:

  ```
  intVar := INT#1;
  REPEAT
  intVar := intVar + INT#1;
  UNTIL intVar = INT#0
  END_REPEAT;
  ```

EXIT

Overview:

Use this statement only inside a repeat construct (FOR construct, WHILE construct, or REPEAT construct) to exit the repeat construct.

Use this statement inside an IF construct to exit from the repeat construct when a condition is met.

Reserved Words:

EXIT

Construct Structure (e.g., in an IF Construct):

```
FOR (WHILE, REPEAT) <statement>
            .
            .
            .
IF <condition_expression> THEN EXIT;
END_IF;
            .
            .
            .
END_FOR (WHILE, REPEAT);
```

Application:

Use EXIT to end a repeating process before the end condition is met.

Description (e.g., in an IF Construct):

If the <condition_expression> is TRUE, the repeat construct (FOR construct, WHILE construct, or REPEAT construct) is ended and all code inside the repeat construct after the EXIT statement is ignored.

Note 1 You can also specify a BOOL variable instead of an expression for the condition expressions.

2 Even if the *<condition_expression>* is TRUE before *<statement>* is executed, <statement> is executed.

Example:

Variable n is repeatedly incremented by 1 from 1 to 50 while the value of n is added to array variable elements *DATA[n]*. However, if *DATA[n]* exceeds 100, the repeat construct is exited.

```
IF A THEN
    DATA[3] :=98;
    FOR n := 1; TO 50 BY 1 DO
        DATA[n] := DATA[n] + n;
        IF DATA[n] > 100 THEN EXIT;
        END_IF;
    END_FOR;
    A :=FALSE;
END_IF;
```

● Function Block Calls

Overview:

This statement calls a function block.

Reserved Words: None

Statement Structure:

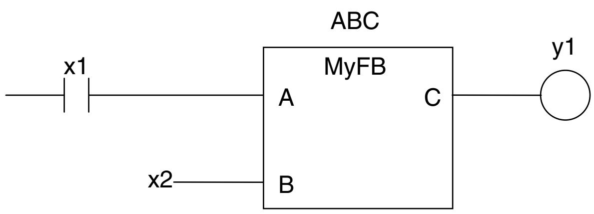
Give the argument specifications (to pass the values of the specified variables to the input variables of the called function block) and the return value specification (to specify the variable that will receive the value of the output variable of the called function block) in parenthesis after the instance name of the function block. There are two methods of writing this statement, as shown in (1) and (2) below. We recommend method 1 for program readability.

Notation Method 1:

Give both the variable names of the called function block and the parameter names of the calling POU.

ABC(A:=x1, B:=x2, C=>y1);
ABC: Function block instance name
A and B: Input or in-out variable names of called function block
x1 and *x2*: Input or in-out parameter of calling POU (can be a constant)
C: Output variable of called function block
y1: Output parameter of calling POU

- Ladder Diagram Expression



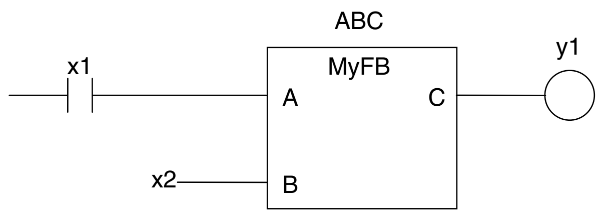
- You can give the arguments and return values in any order.
- You can omit the input variable names and input parameter names. If you omit these names, the values assigned to the input variables for the previous call are assigned to the input variables again. If this is the first time that the function block is called, the input variables are set to their initial values.
- You can omit the output variables and output parameters. If they are omitted, the value of the output variable is not assigned to anything.

Notation Method 2:

Omit the variable names of the called function block and give the parameter names of the calling POU.

ABC(x1, x2, y1);
ABC: Function block instance name
A and B: Omitted. (Input or in-out variable of called function block)
x1 and *x2*: Input or in-out parameter of calling POU (can be a constant)
C: Omitted. (Output variable of called function block or constant)
y1: Output parameter of calling POU

- Ladder Diagram Expression



- The order of parameters is based on the function block definition. The order is the same as the local variable definition for the function block, from top to bottom.

Application:

This statement calls a function block.

Example

- Programming

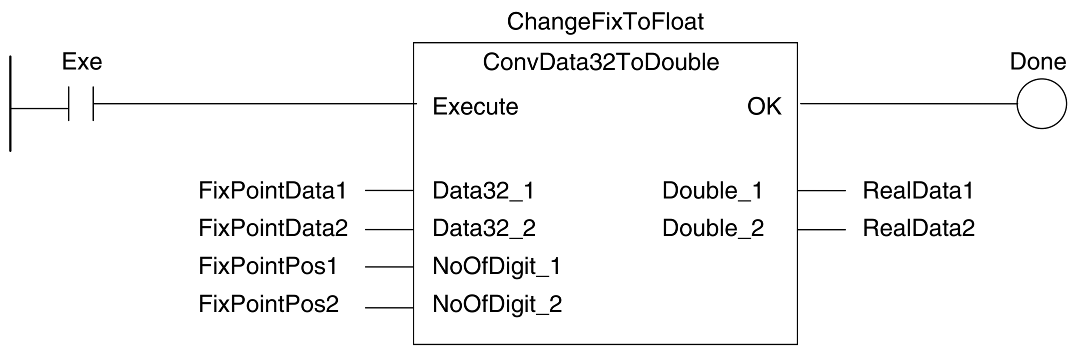
  Notation 1

  ChangeFixToFloat(Execute:=Exe,Data32_1:=FixPointData1, Data32_2:=FixPointData2, NoOfDigit_1:=FixPointPos1,
  NoOfDigit_2:=FixPointPos2,OK=>Done,Double_1=>RealData1,
  Double_2=>RealData2);

  Notation 2

  ChangeFixToFloat(Exe, FixPointData1, FixPointData2, FixPointPos1, FixPointPos2,
  Done, RealData1, RealData2);

- Ladder Diagram Expression



- Function Block Definition
  Function block name: ConvData32ToDouble
  Function Block Variables

I/O	Variable name	Data type
Input variables	Execute	BOOL
	Data32_1	DINT
	Data32_2	DINT
	NoOfDigit_1	INT
	NoOfDigit_2	INT
Output variables	OK	BOOL
	Double_1	LREAL
	Double_2	LREAL

- Program Variables

Variable name	Data type	Comments
ChangeFixToFloat	ConvData32ToDouble	Convert from fixed-point to floating-point.
Exe	BOOL	Execution trigger
FixPointData1	DINT	Decimal point position specification data 1
FixPointPos1	INT	Number of digits below decimal point 1
FixPointData2	DINT	Decimal point position specification data 2
FixPointPos2	INT	Number of digits below decimal point 2
Done	BOOL	Normal end
RealData1	LREAL	Floating-point data 1
RealData2	LREAL	Floating-point data 2

Omitting Parameters

When you call a function block, you can omit parameters that are not required. The following table shows when you can omit parameters.

POU type	Variables for the called POU	Notation pattern		Omission
		Parameters included	Examples	
FB	Given (notation method 1)	All parameters given	instance(x:=a,y:=b,z:=c);	OK
		More than one parameter given	instance(x:=a,y:=b);	
		One parameter given	instance(y:=b);	
		No parameters given	instance(x:=);	---
	Given (notation method 2)	All parameters given	instance(a,b,c);	OK
		All parameters not given	instance();	
		Only the first parameter given	instance(a);	---
		One parameter given	instance(a, ,);	
		More than one parameter given	instance(a,b);	

OK: Possible (initial used), ---: A building error will occur.

Function Calls

Overview:

This statement calls a function.

Reserved Words: None

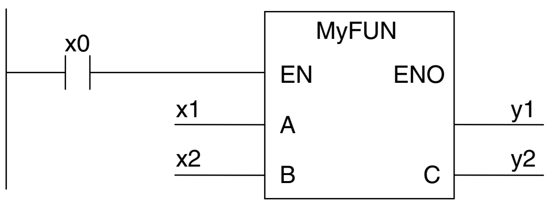
Statement Structure:

Give the output parameter to which the return value is assigned on the left side of the assignment keyword (:=). On the right side, give the argument specifications (to pass the values of the specified variables to the input variables of the called function) inside the parenthesis after the function name. There are two methods of writing this statement, as shown in (1) and (2) below.
We recommend method (1) for program readability.

Notation Method 1:

```
IF (x0=TRUE) THEN
    y1 := MyFUN(A:=x1, B:=x2, C=>y2);
END_IF;
```

- Ladder Diagram Expression



MyFUN: Function name

x0: Specifies whether to call the function.

A and B: Input variable names of the called function

x1 and x2: Input parameters of the called function

C: Output variable name of the called function

y1: Storage location for the return value from the called function

y2: Output parameters of the called function

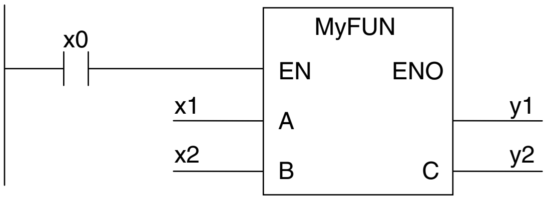
- You can give the arguments in any order.
- You can omit the input variable names and input parameter names. If they are omitted, the input variables are assigned their initial values.
- You can omit *EN* as well. If it is omitted, *EN* is assigned a value of TRUE.

Notation Method 2:

Omit the variable names of the called function and give the parameter names of the calling POU.

```
IF (x0=TRUE) THEN
    y1 := MyFUN(x1, x2, y2);
END_IF;
```

- Ladder Diagram Expression



MyFUN: Function name

x0: Specifies whether to call the function.

A and B: Input variable names of the called function

x1 and x2: Input parameters of the called function

C: Output variable name of the called function

y1: Storage location for the return value from the called function

y2: Output parameters of the called function

- The order of parameters is based on the function definition. The order is the same as the local variable definition for the function, from top to bottom.

Example:

- Programming

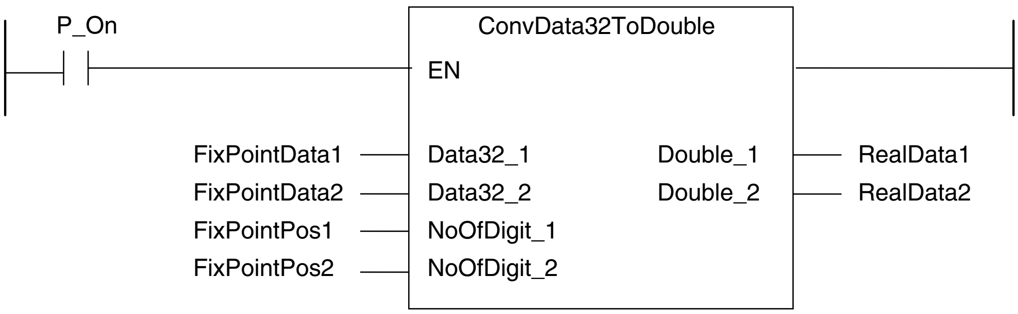
- Notation 1

  ConvData32ToDouble(Data32_1:=FixPointData1,Data32_2:=FixPointData2,
  NoOfDigit_1:=FixPointPos1, NoOfDigit_2:=FixPointPos2,
  Double_1=>RealData1, Double_2=>RealData2);

- Notation 2

  ConvData32ToDouble(FixPointData1, FixPointData2, FixPointPos1, FixPointPos2,
  RealData1, RealData2);

- Ladder Diagram Expression



- Function Definition
  Function name: ConvData32ToDouble
  Function Variables

I/O	Variable name	Data type
Input variables	Execute	BOOL
	Data32_1	DINT
	Data32_2	DINT
	NoOfDigit_1	INT
	NoOfDigit_2	INT
Output variables	Double_1	LREAL
	Double_2	LREAL
Return value	---	BOOL

- Program Variables

Variable name	Data type	Comment
ChangeFixToFloat	ConvData32ToDouble	Convert from fixed-point to floating-point.
Exe	BOOL	Execution trigger
FixPointData1	DINT	Decimal point position specification data 1
FixPointPos1	INT	Number of digits below decimal point 1
FixPointData2	DINT	Decimal point position specification data 2
FixPointPos2	INT	Number of digits below decimal point 2
Done	BOOL	Normal end
RealData	LREAL	Floating-point data 1
RealData	LREAL	Floating-point data 2

Application:

This statement calls a function.

Omitting Parameters

When you call a function, you can omit parameters that are not required. The following table shows when you can omit parameters.

POU type	Variables for the called POU	Notation pattern		Omission
		Parameters included	Example	
FUN	Given (notation method 1)	All parameters given	FUN(x:=a,y:=b,z:=c);	OK
		More than one parameter given	FUN(x:=a,y:=b);	
		One parameter given	FUN(y:=b);	
		No parameters given	FUN(x:=);	---
	Given (notation method 2)	All parameters given	FUN(a,b,c)	OK
		No parameters given	FUN();	
		Only the first parameter given	FUN(a);	---
		One parameter given	FUN(a, ,);	
		More than one parameter given	FUN(a,b);	

OK: Possible (initial used), ---: A building error will occur.

Precautions for the ST Language

Observe the following precautions when you use the ST language in the user program.

● Implicit Casts

If the data types of the operands do not match, as shown below, the data types are converted automatically according to the implicit cast rules. If the implicit cast rules are not satisfied, a building error occurs.

(1) When the data types of the operands in the expression on the right side of the assignment statement are not the same

Example:

```
A: = INT#10 + SINT#2;
```

(2) When the data types of the operands on the right and left sides of the assignment statement are not the same

Example:

```
   A: = B + C;
   /     \ /
REAL     INT
```

(3) When the data types of the operands in statement are not the same

Example:

```
    INT   LINT
      \   /
CASE A+B OF
   INT#1:
     def:=INT#10;
```

The casting rules are described for the following three cases.

Casting Rules When the Right-hand Side of an Assignment Statement Is an Arithmetic Expression

- For the right-hand operand, you can use any combination of the data types that are supported for the operator operand.
- Of the operands on the right side, the operand with the highest rank is considered the data type of the entire side. (Refer to the *Data Type Ranking Table* given below for the data type ranks.)

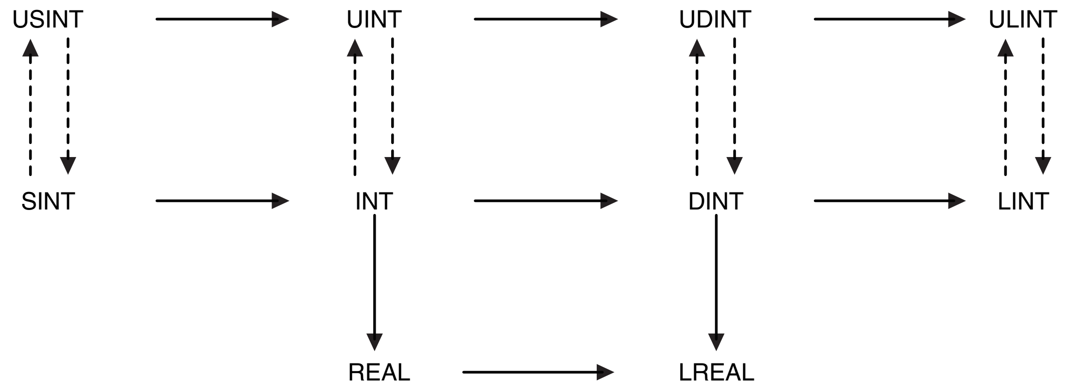
Data Type Ranking Table:

The higher the rank, the larger the range of numerical values that the data type can express.

Rank	Data type
1	SINT
2	USINT
3	INT
4	UINT
5	DINT
6	UDINT
7	LINT
8	ULINT
9	REAL
10	LREAL
11	BYTE
12	WORD
13	DWORD
14	LWORD

Casting Rules When You Assign the Right-hand Value to the Left-hand Side

In the following chart, a cast is performed if an arrow connects the data type of the source to the data type of the assignment destination. Any combination that is not connected will cause a building error.



When you assign the value, the sign and absolute value of the number do not change.

When you assign the value, the sign and absolute value of the number may change.

Example: intVar := -1; (* intVar := 16#FFFF *)
uintVar := 1;
uintVar := intVar; (* uintVar:= 16#FFFF, or –1 was assigned but the result is 65535 *)

Even if the arrow does not connect directly to a data type, you can still perform assignments for the data types. For example, SINT->USINT->UINT->UDINT->ULINT are all connected, so you can write an assignment such as ULINT:=SINT.

Precautions for Correct Use

Observe the following precautions when casting UDINT to ULINT data, DINT to LINT data, or DINT to LREAL data.

All of these are casts from 32-bit data to 64-bit data. If the result of the calculation of the right side of the assignment statement exceeds the range of 32-bit data, the correct value may not be assigned.

Example: For the following assignment statements, the result of the addition in the third statement exceeds the range of 32-bit data. An overflow will result and 0 will be assigned to *LintVar*.

```
UdintVar := UDINT#16#FFFF_FFFF;          // Upper limit of 32-bit data
DintVar   := DINT#1;                      // 1
LintVar := (UdintVar + DintVar)/DINT#2;   // (Upper limit of 32-bit data + 1)/2
```

In a case like this one, convert the data to 64-bit data before you perform the calculation. To do this for the above example, change the assignment status as shown below.

```
LintTmp1 := UDINT_TO_LINT(UDINT#16#FFFF_FFFF);  //Convert UDINT to LINT data.
LintTmp2 := DINT_TO_LINT(DINT#1);               // Convert DINT to LINT data.
LintVar    := (LintTmp1 + LintTmp2) / DINT#2;
```

Casting Rules in Expressions in Statements

The implicit cast rules for right-hand arithmetic expressions in assignment statements and for assigning the value of the right-hand side to the left-hand side also apply to expressions in statements.

Example:

```
CASE (A+B+C) OF
    Result1:
        to
    ResultN:
        to
END_CASE;
```

● Order of Execution of Functions

The order of execution of functions is not defined for functions in expressions. The order of execution of functions depends on the unit version of the CPU Unit, the version of the Sysmac Studio, and the notation. Precaution is required in cases where the results of an expression may depend on the order of execution of the functions, such as in the following cases.

- Expressions that contain more than one function that access the same global variable
- Expressions that contain a function and a variable whose value is changed by that function

Expressions That Contain More Than One Function That Access the Same Global Variable

In the following example, the order of execution of the three functions is not necessarily the same as the order of execution of the calculations, which is determined by the priority of the operators. Therefore, it is possible that the functions are executed in the following order: FUN2, FUN3, and then FUN1.

```
result := FUN1() + FUN2() * FUN3();
```

If all three of the functions in the above expression access and write the same global variable, the value of the *result* variable may change depending on the order of execution of the functions.

To ensure that the three functions are always executed in the same order, the expression is broken up. The following notation is used to execute the functions in the following order: FUN2, FUN3, and then FUN1.

```
tmp2  := FUN2();
tmp3  := FUN3();
result := FUN1() + tmp2 * tmp3;
```

Expressions That Contain a Function and a Variable Whose Value Is Changed by That Function

The following expression contains a function and a variable whose value is changed by that function.

```
result := varA + FUN4(out => varA);
```

In the above expression, the first element on the right side, variable varA, is not necessarily evaluated before FUN4 is executed. Therefore, the value of the result variable may change depending on the order of varA evaluation and FUN4 execution.

To ensure that varA evaluation and FUN4 execution always occur in the same order, the expression is broken up. The following notation is used to evaluate varA first and then execute FUN4.

```
tmp    := varA;
result := tmp + FUN4(out => varA);
```

The following notation is used to execute FUN4 first and then evaluate *varA.*

```
tmp    := FUN4(out => varA);
result := varA + tmp;
```

Calculation Precision of Expressions with Constants without Data Type Specifications

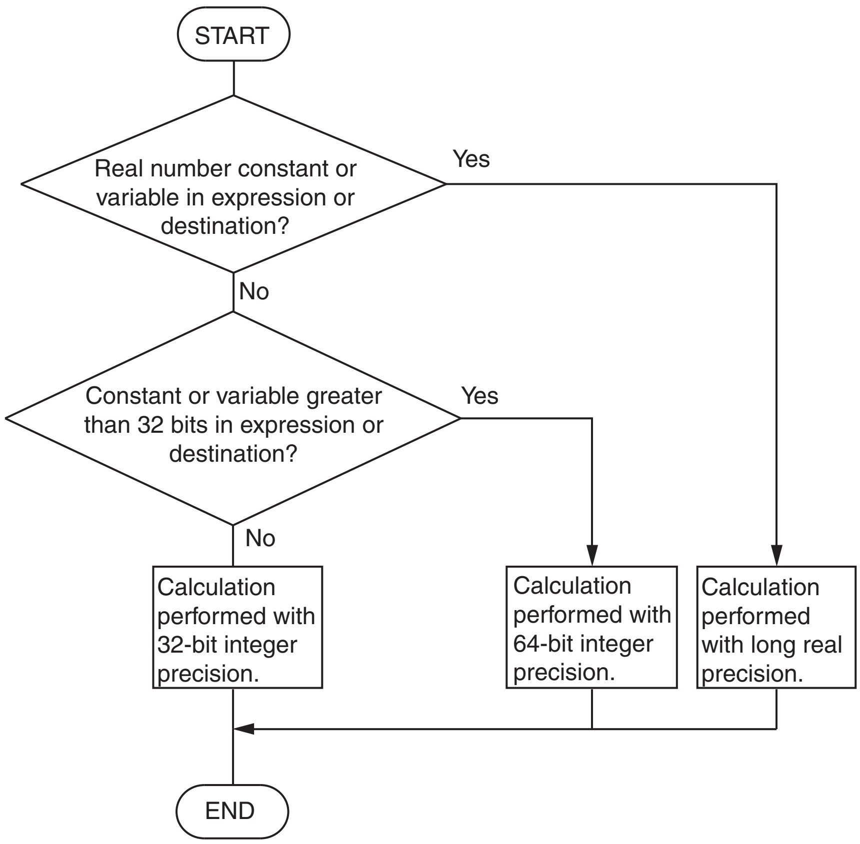
The calculation precision of an expression that contains a constant without a data type specification is automatically determined by the data types of the variables and constants that are given in the expression and destination.

Notation of Constants

If a constant is given without a decimal point, such as 100, it is processed as an integer. If a constant is given with a decimal point, such as 100.0, it is processed as a real number.

Expression Calculation Precision

The calculation precision of an expression is either 32-bit integer, 64-bit integer, or long real precision depending of the data types of the variables and constants given in the expression and destination. The following rules apply to the calculation precision of an expression.



Example: realv := 2 + 3 * 4; // The data type of the *realv* variable is REAL.

The *realv* variable is a real number, so the calculation is performed with long real precision. The calculation result is 14.0.

realv := 2 + 3.0 * 4; // The data type of the *realv* variable is REAL.

The 3.0 constant and the *realv* variable are real numbers, so the calculation is performed with long real precision. The calculation result is 14.0.

lintv := 2 + 3 * 4; // The data type of the *lintv* variable is LINT.

There is no constant or variable that is a real number, but the *lintv* variable exceeds 32 bits, so the calculation is performed with 64-bit integer precision. The calculation result is 14.

intv := 2 + 3 * 4; // The data type of the *intv* variable is INT.

There is no constant or variable that is a real number and there is no constant that exceeds 32 bits, so the calculation is performed with 32-bit integer precision. The calculation result is 14.

However, the calculation precision of division is determined only by the divisor and dividend. The rules for determining the calculation precision are the same as those in the previous flowchart.

Example: realv := 2 / 3 * 4; // The data type of the *realv* variable is REAL.

Dividing 2 by 3 does not include an integer that exceeds 32 bits for the divisor or dividend, so the calculation is performed with 32-bit integer precision. The calculation result is 0.

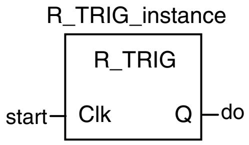
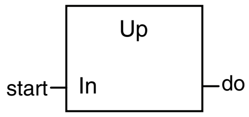
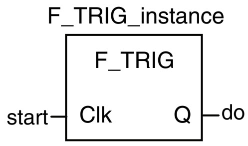
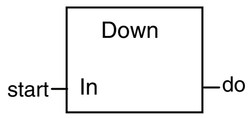
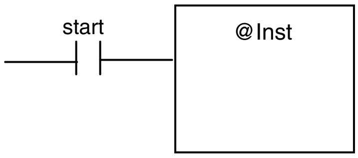
In the next step, the *realv* variable is a real number, so the calculation of 0 * 4 is performed with long real precision. The calculation result is 0.0.

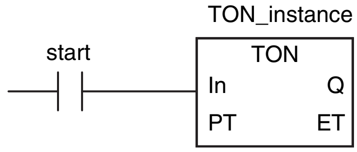
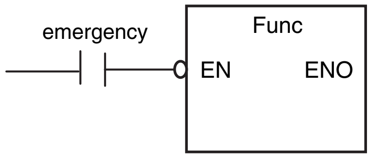
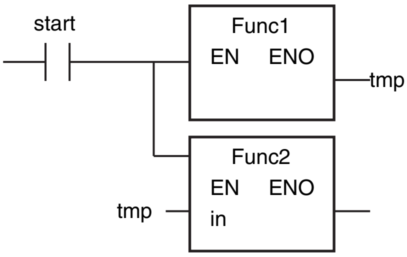
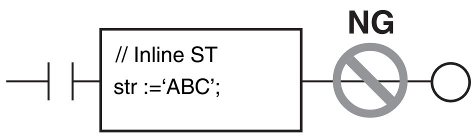
Precautions for Correct Use

The calculation precision of an expression that contains a constant without a data type specification is automatically determined by the notation of the constant and the data types of the variables that are given in the expression. Therefore, calculations may be performed with unintended precision. We recommend that you specify the data type for real numbers, such as REAL#1.0.

Differences between ST and Ladder Diagrams

The differences between ST and ladder diagrams are described below.

Item	Ladder diagram	ST (including inline ST)
Input differentiation	● **Change to TRUE** • Method 1 start do ----|↑|-----------○ • Method 2  • Method 3 	● **Change to TRUE** • **Method 1** R_TRIG_instance (Clk:=start, Q=>do); * R_TRIG_instance is an instance of the R_TRIG instruction.
	● **Change to FALSE** • Method 1 start do ----|↓|-----------○ • Method 2  • Method 3 	● **Change to FALSE** • **Method 1** F_TRIG_instance (Clk:=start, Q=>do); * F_TRIG_instance is an instance of the F_TRIG instruction.
Instruction differentiation	● **Upward Differentiation** 	● **Upward Differentiation** There is no equivalent in ST. You must create it in logic. Example: • Method 1 R_TRIG_instance (Clk:=start, Q=>do); IF (do = TRUE) THEN Inst(); END_IF; • Method 2 IF (start = TRUE) THEN IF (pre_start = FALSE) THEN Inst(); END_IF; END_IF; pre_start:=start;// Update previous value.

Item	Ladder diagram	ST (including inline ST)
Instructions that last multiple task periods	With the TON instruction, multiple cycles are required from the start of instruction execution to the end and the instruction is reset when the power flow is FALSE. Therefore, you need to declare only one instance to both execute the instruction and reset it. 	You must declare two instances, one for execution and one to reset, as shown below. IF (start = TRUE) THEN TON_instance(In:=TRUE, *omitted*); // Start timer. ELSE TON_instance(In:=FALSE, *omitted*); // Reset timer. END_IF;
Function/function block argument reversal specifications	Add a circle to indicate reversal at the intersection of the BOOL argument and the function/function block. 	Add a NOT operator to the argument. * You can add NOT operators to any BOOL variable, not just arguments. IF (NOT emergency) THEN Func(); END_IF;
Multi-stage connections		IF(start=TRUE) THEN Func2(in := Func1()); END_IF;
Post-connecting ladder instructions	You can connect only other Out instructions after an Out instruction. 	You cannot continue the ladder diagram after inline ST. 
Program divisions	You can create sections.	You cannot create sections.

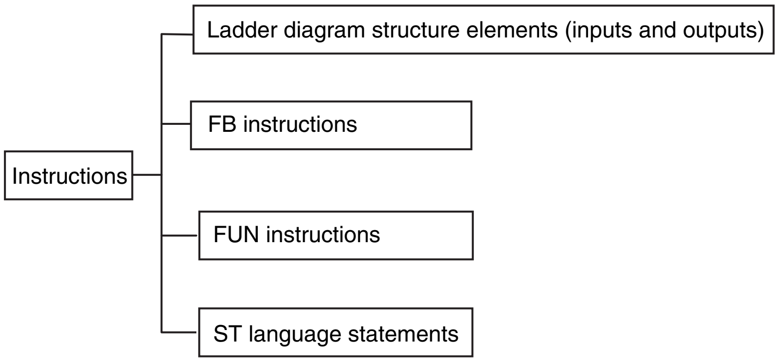
6-6 Instructions

This section describes the instructions that are pre-defined by the NJ/NX-series Controller.

For details on these instructions, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) and *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508).

6-6-1 Instructions

Instructions are the smallest unit of the processing elements that are provided by OMRON for use in POU algorithms. Instructions are classified as shown below.



Programs, user-defined functions, and user-defined function blocks consist of these instructions.

6-6-2 Basic Understanding of Instructions

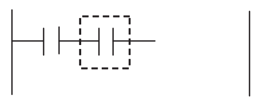
The fundamental specifications of the instructions follow the specifications of functions and function blocks.

This section describes specifications that are unique to instructions.

Ladder Diagram Structure Elements (Inputs and Outputs)

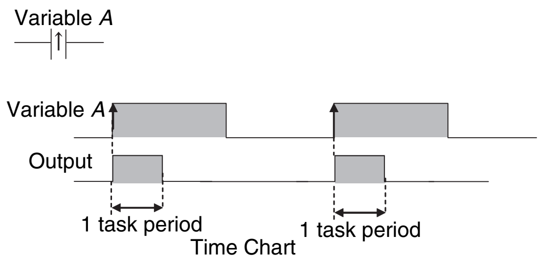
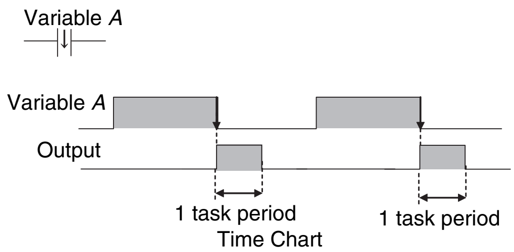
● Locations

Instructions for ladder diagram inputs and outputs have certain positions where they can be placed, as shown below.

Classification		Locations	Diagram
Input instructions	Logical start	Connected directly to the left bus bar or is at the beginning of an instruction block.	
	Intermediate instructions	Between a logical start and the output instruction.	
Output instructions		Connected directly to the right bus bar.	

● Instruction Options

Some ladder diagram instructions for inputs also detect changes to TRUE or changes to FALSE if you add an upward arrow or downward arrow to them.

Change to TRUE (↑)		The instruction reads input status, makes comparisons, tests bits, or performs other types of processing every task period and outputs the power flow when result changes from FALSE to TRUE. The output power flow changes to FALSE in the next task period (after it is TRUE for one task period).
Change to FALSE (↓)		The instruction reads input status or performs other types of processing every task period and outputs the power flow when result changes from TRUE to FALSE. The output power flow changes to FALSE in the next task period (after it is TRUE for one task period).

Function Block Instructions

● Execution Conditions

The operation of the execution condition for an FB instruction depends on the instruction.

A specific input variable for the execution condition is defined for each instruction.

Examples: *Execute* specifies a change to TRUE or a change to FALSE in the execution condition.

Enable causes the instruction to be executed each task period according to the current execution condition.

Function block instructions are unconditionally executed for as long as the POU that called them is executed.

● Instruction Options

Instruction options cannot be specified.

FUN Instructions

● Execution Conditions

All FUN instructions have *EN* inputs as execution conditions. The FUN instruction is executed each task period as long as *EN* is TRUE.

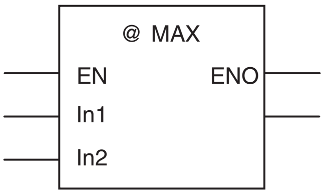
● Instruction Options

In a ladder diagram, you can add the following instruction options to specify a change to TRUE or a change to FALSE as the execution condition for that instruction. ST statements do not have options.

Instruction Options		Symbol	
Differentiation option	Change to TRUE	@	This option creates an upwardly differentiated instruction. The instruction is executed only once when *EN* changes to TRUE.

To add an instruction option, add one of the option symbols listed in the table above before the instruction.

Example:



Information That Applies to Both FB Instructions and FUN Instructions

● Condition Flags

System-defined variables that are assigned values that represent the result of instruction processing are called Condition Flags. The only Condition Flag for an NJ/NX-series Controller is the Carry Flag (P_CY).

The Carry Flag serves the following purposes.

- It shows whether the result of processing an instruction exceeds the range that can be expressed by the data type of the output variable.
- It shows whether an overflow occurred in a bit shift instruction for bit string data. For details, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

6-6-3 Instruction Errors

Instruction errors refer to the errors that occur when an instruction is executed. This section describes when an instruction error occurs, which error is detected as an instruction error, and what operation follow an instruction error, etc.

Timing When Instruction Errors Occur

The timing when instruction errors occur can be divided into the following three cases. Detectable errors and operations following to the errors differ by the timing when instruction errors occur.

- When the values of input parameters or in-out parameters are checked before instruction execution.
- When internal processing is performed during instruction execution.
- When the values of output parameters are checked after instruction execution.

Errors Detected As Instruction Errors

The followings are the errors detected as instruction errors. Different errors are detected depending on the timing when instruction errors occur.

Errors detected before or after instruction execution

The followings are the errors detected before or after instruction execution.

- Reading or writing an array variable from or to an element beyond the array range.
- Assigning a string that is longer than the defined byte length to a STRING variable.
- Assigning a string that does not end with a NULL character to a STRING variable.
- Dividing an integer variable by 0.

Errors detected during instruction execution

Errors detected during instruction execution differ by instruction. For details on errors detected in each instruction, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

Operation for Instruction Errors

The operation for the following elements differ depending on whether an instruction error occurs or not: output variable ENO, output variable Error, output variable ErrorID, system-defined variable P_PRGER, and events. The details on the operations are described below.

Output variable ENO, output variable Error, and output variable ErrorID

ENO (enable out), Error, and ErrorID (error code) are the output variables that indicate whether an error exists or not. Each instruction has different output variables. The meaning of each variable and its value on an instruction error are shown below. The values vary by the timing when an instruction error occurs.

Output variable	Data type	Meaning	Value when an instruction error occurs		
			Before instruction execution*1	During instruction execution	After instruction execution*2
ENO	BOOL	TRUE : Normal end FALSE : Error end, Execution in progress, or Not executed	FALSE	FALSE	TRUE
Error	BOOL	TRUE : Error End FALSE : Normal end, Execution in progress, or Not executed	FALSE	TRUE	FALSE
ErrorID	WORD	Error code on Error end, and WORD#16#0 on Normal end	WORD#16#0	Error code	WORD#16#0

*1 If an instruction error occurs before execution of an instruction, the instruction will not be executed. Therefore, the value of each output parameter before instruction execution will be retained.

*2 If an instruction error occurs after execution of an instruction, the instruction itself will be regarded as normally ended. Therefore, the values of output variables of the instruction will be assigned to the output parameters. Values of the output parameter to which an error occurred are retained as the one before the instruction execution.

System-defined variable P_PRGER

The system-defined variable P_PRGER is a flag that indicates the occurrence of an instruction error. If an instruction error occurs, the value will change to TRUE regardless of when the error occurred. When the instruction ends normally, the value will be retained.

For the details on P_PRGER, refer to *Instruction Error Flag* on page 6-124.

Events

When an instruction error occurs, an event is created for it. For details on events, refer to *Events for Instruction Errors* on page 6-125.

Version Information

A CPU Unit with unit version 1.02 is required to create events for instruction errors.

Output Parameters in Ladder Diagrams

The following table shows the values of output parameters when an instruction, user-defined function, or user-defined function block that is created in a ladder diagram ends normally or has an instruction error.

Condition	Type of output parameter	Value of output parameter
Normal end	Power flow output	Values are updated according to the internal algorithm.
	BOOL parameter output	
	Parameter output other than BOOL	
Instruction error	Power flow output	Set to FALSE.
	BOOL parameter output	The previous values are retained.
	Parameter output other than BOOL	

Operation When a Syntax Error Occurs in a POU Written in ST

● Errors in Assignment Statements

When an error occurs in an assignment statement written in ST, that line is not executed.

```
5   a = b / (c + d) + e * f + ABS(g);
6   x := 1;
```

For example, if a division by zero error occurs in (b/(c+d)) on line 5, execution of line 5 is cancelled (the value of a is not changed) and line 6 is executed.

This operation is the same as when the output *ENO* of a user-created function is FALSE.

```
5   a = User-created_function_block (b) + c;
6   x := 1;
```

When the *ENO* output from the user-created function is FALSE, execution of line 5 is cancelled (the value of a is not changed) and line 6 is executed.

● Errors in IF Constructs

If a syntax error occurs in ST, perform error processing for the syntax error.
When the value of (*c+d*), below, is zero, the lines between the IF and END_IF are not executed.

```
POU"AA"
   5   IF a = b / (c + d) THEN
   6      x := 1;
   7   ELSE
   8      x := 2;
   9   END_IF;
  10   y := 10;
   :   :
       IF P_PRGER = TRUE THEN
          x:= initial_value; (*Processing when an error occurs*)
          y:= initial_value;
       END_IF;
```

The user must include a safety processing for possible errors.

● Syntax Errors in ST

The following syntax errors can occur in ST.

- Exceeding the number of elements in an array.
- No parameter set for in-out variable.
- STRING assignment: When the text string size (bytes) of the left side is less than the text string length (bytes) of the right side
- Division by zero (excluding floating-point number calculations)

* When the value of a floating-point number is nonnumeric, the result of the calculation will also be nonnumeric. This is not considered an error.

● Operation for Structure Errors

The *P_PRGER* Flag changes TRUE and the following occurs.

Syntax	Error location	Operation
Assignment statement		The line is not executed.

Syntax	Error location	Operation
Control con-structs	**IF condition expression**	No statements between IF and END_IF are executed.
	CASE condition expression	No statements between CASE and END_CASE are executed.
	FOR condition expression	No statements between FOR and END_FOR are executed.
	WHILE condition expression	No statements between WHILE and END_WHILE are executed.
	REPEAT condi-tion expression	No statements between REPEAT and END_REPEAT are executed.

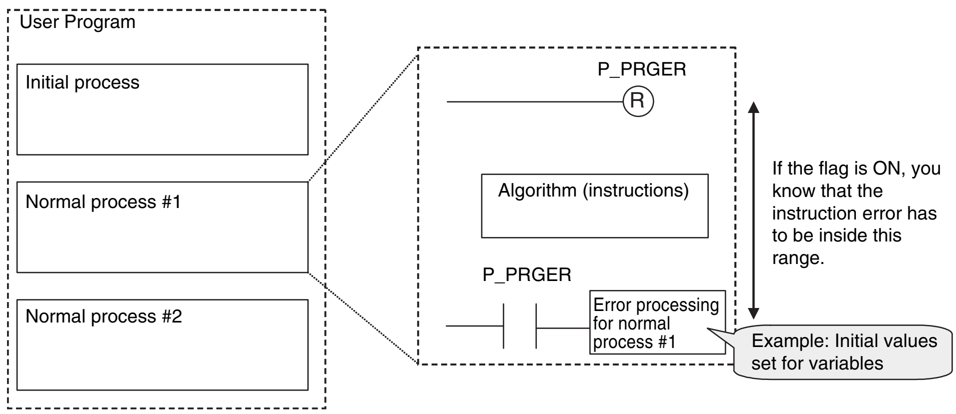
Instruction Error Flag

When an instruction error occurs in a ladder algorithm or when a syntax/function error occurs in an ST algorithm, the *P_PRGER* (Instruction Error Flag) system-defined variable changes to TRUE. The *P_PRGER* flag is a local variable for the program. This flag changes to TRUE when an instruction error occurs in the program, and remains TRUE during the next task period.

Variable name	Meaning	Function	Data type	Range of values	Initial value	Read /write
P_PRGER	Instruction Error Flag	This flag changes to and remains TRUE when an instruction error occurs. After this flag changes to TRUE, it stays TRUE until the program changes it back to FALSE.	BOOL	TRUE or FALSE	FALSE	Read /write

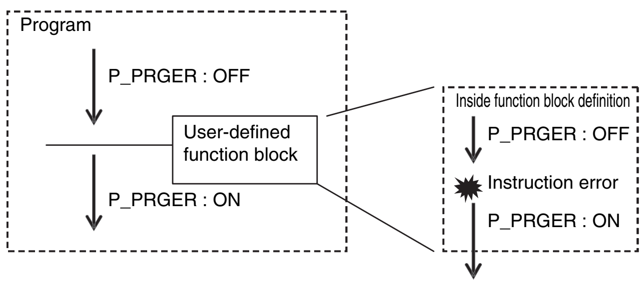
The user can write the P_PRGER Flag. You can temporarily set the value of this flag to FALSE through a user operation to determine if the error occurs within a specific range, for example. After this flag changes to TRUE, it remains TRUE until the operating mode is changed or the flag is overwritten by a program.

Example:



The *P_PRGER* Flag also changes to TRUE when an instruction error occurs inside a user-defined function block that is used by the program.

Example:



Events for Instruction Errors

When an instruction error occurs, an event is created for it. Refer to *8-6 Event Logs* for the procedure to check events. For information on the events that are created, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

Precautions for Correct Use

- To create events for instruction errors, you must select *Use* for *Event Log Settings – Instruction Error Output* on the Sysmac Studio. Refer to *4-2-2 Controller Setup* and to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504-E1-04 or later) for information on the Controller Setup.
- If you change the user program after an instruction error occurs, the information in the event log may no longer be correct.
- If an instruction with an error is executed repeatedly, an instruction error or event is created each time the instruction is executed. This may cause the event log to exceed the maximum number of events. If this occurs, older events are overwritten.

Version Information

- A CPU Unit with unit version 1.02 is required to create events for instruction errors.
- A CPU Unit with unit version 1.02 or later and Sysmac Studio version 1.03 or higher are required to specify whether to output instruction errors when they occur.

Additional Information

- If an error occurs in a motion control instruction, two events are created, one for the instruction error and one for the motion control instruction. For details on events for motion control instructions, refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508).
- Events for motion control instruction are created even if you select *Do not use* for *Event Log Settings – Instruction Error Output* in the Controller Setup on the Sysmac Studio.

6-7 Namespaces

This section provides the specifications for namespaces and the procedures to use them. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for the procedures to manipulate them.

Version Information

A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use namespaces.

6-7-1 Namespaces

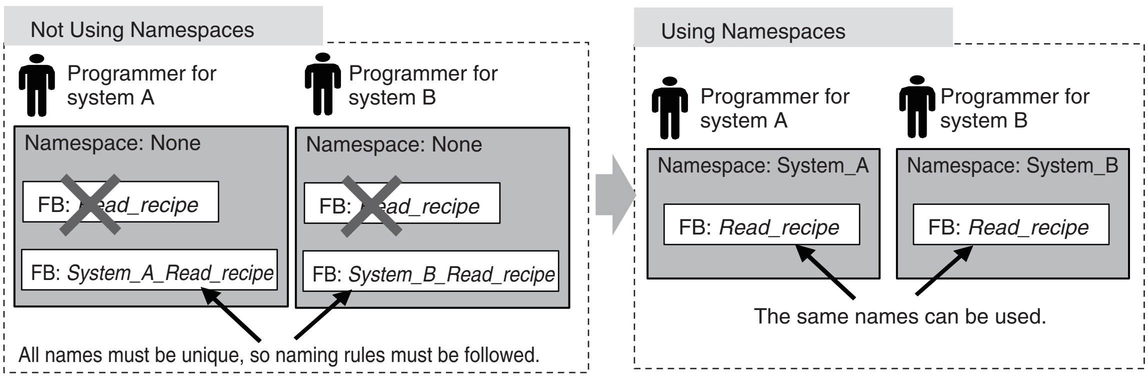
Namespaces are a system for grouping function block definitions and other entities to manage them in nested structures. They are similar to grouping files in folders to manage them in a directory structure. If you do not use namespaces, the name of each function block definition or other entity must be unique. If you use namespaces, you can use the same name more than once by setting namespaces. Using namespaces is not required.

Features of Namespaces

Namespaces provide the following features.

● Preventing Duplicated Names

As long as different namespaces are used, you can use the same name for a function block or other entity more than once. For example, assume that several systems must be programmed, and that a different programmer will program each of them. Here, it would be likely that the same names would be used for different function block definitions or other entities. If you did not use namespaces, you would have to create naming rules to prevent the duplication of names. However, if you set a different namespace for each system, programming would be possible without worrying about duplicating names with other systems.



6-7-2 Namespace Specifications

This section describes what namespaces can be used for, namespace notation, and namespace declarations.

Namespace Usage

You can use namespaces for the entities that are listed in the following table. You cannot use them for local variables.

Library object	Details
POU definitions	Function definition names and function block definition names
Data types	Structure data type names, union data type names, and enumeration data type names

Namespace Notation

Separate the levels in a namespace with backslashes (\). To use a namespace in a POU algorithm, place two backslashes (\\) at the front of the namespace.

Examples:

Location of namespace	Expression
Outside of an algorithm	*System_A\Read_recipe*
Inside of an algorithm	*System_A\Read_recipe*

● Fully Qualified Names and Short Names

The fully qualified name of an entity is the name that includes the name of the namespace. The short name of an entity is the name that does not include the name of the namespace.

In the algorithm in a POU definition, you can use the short name of any POU definition that has the same namespace as the POU definition of the algorithm.

Example:

$$\underbrace{\underbrace{\textit{System\_A}}_{\text{Name of namespace}}\backslash\underbrace{\textit{Read\_recipe}}_{\text{Short name}}}_{\text{Fully qualified name}}$$

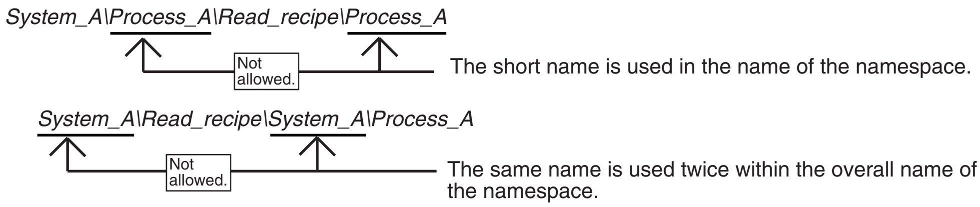
● Restrictions on Namespace Notation

- You can use the same characters as you can for variable names. For details, refer to *6-3-12 Restrictions on Variable Names and Other Program-related Names*.
- The following table gives the limits to the number of characters in the names of namespaces.

Name	Maximum size	Character encoding
Names of namespaces	93 bytes	UTF-8
Short names	127 bytes	

Precautions for Correct Use

- An error will occur when you build the program if the short name of a variable is also used in the name of the namespace.

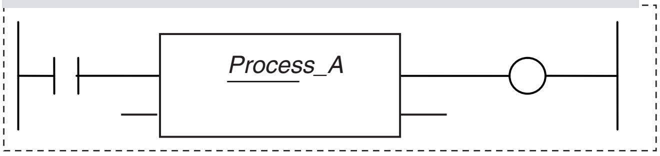


- You can use the short name of a POU definition in the algorithm of a POU definition if it is in the same namespace. However, an error will occur when you build the program if there is a POU definition or data type with the same short name at a higher level in that namespace.
  For example, assume that the following POU definitions are used. You can use the short name to call *System_A\Read_recipe\Process_A* from within the algorithm of the *Process_B* function block definition (which is in the *System_A\Read_recipe* namespace) because *Process_A* is in the same namespace.

  POU Definitions

  System_A\Read_recipe\Process_A
  System_A\Read_recipe\Process_B

  Notation in the Algorithm of the
  System_A\Read_recipe\Process_B Function Block Definition



  If, however, a *System_A* POU definition also exists at a higher level than the *System_A\Read_recipe* namespace, “Process_A” exists twice. Therefore, an error will occur when you build the program.
  In this case, you must use the fully qualified name or change the short name.

  POUPOU Definitions

  Process_A (no namespace)
  System_A\Process_A

- If any names are the same as a reserved word, an error will occur when you check the program.

Namespace Declarations

To program with namespaces, you can declare the namespaces in advance before you use them in the algorithm of a POU definition.

After you declare the namespace in the POU definition, you can use the short name of any POU definition or other entity that has the same namespace. You can also use the fully qualified name even if you declare the namespace.
In the algorithm in a POU definition, you can use the short name of any function definition or function block definition that has the same namespace as the POU definition of the algorithm even if you do not declare the namespace.

You can declare more than one namespace for the same POU definition.

● Notation Examples

Notation examples are provided below for creating a function block definition when declaring the namespaces to use in the function block definition and when not declaring the namespaces.

Examples:

In this example, the *Read_recipe* and *Calculate_upper_limit* function block definitions are used in the algorithm for the *Lifter* function block definition. Each of these function block definitions is in a different namespace. In the *Lifter* function block definition, only the *System_C* namespace is declared. The fully qualified name must be given for the *Read_recipe* function block, which is not in the *System_C* namespace. The short name can be given for the *Calculate_upper_limit* function block, which is in the *System_C* namespace.

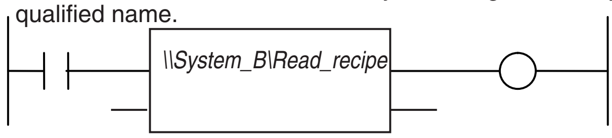
Namespace	Entity	Short name
System_A	Function block definition	*Lifter*
System_B	Function block definition	*Read_recipe*
System_C	Function block definition	*Calculate_upper_limit*

The following notation is used in the namespace declaration for the *Lifter* function block definition.

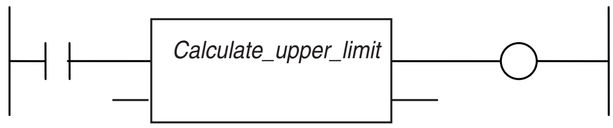
System_C

Notation for the Algorithm of the Lifter Function Block Definition

- When Namespace Is Not Declared
  The *System_B* namespace of the *Read_recipe* function block definition is not declared, so you must give the fully qualified name.



- When Namespace Is Declared
  The *System_C* namespace of the *Calculate_upper_limit* function block definition is declared, so you can give the short name.



● Restrictions of Declarations

- You can use short names only in the algorithm of a POU definition.

Precautions for Correct Use

- An error is detected during the program check in the following cases.
  - If a namespace that does not exist is declared
  - If you declare more than one namespace for one POU definition, and a POU definition, data type, or other entity with the same name exists in two or more namespaces
- An error will occur when you build the program if the same name is used as follows for different POU definitions or data types.
  - If the same name is used for the namespace of a POU definition and at a higher level in the namespace
  - If the same name is used in a declared namespace
  - If the same name is used without a namespace

Namespace	Entity	Short name		
System_A\Lifter	Function block definition	Process_A	Not allowed.	The name is used in the namespace of the POU definition.
System_A	Function block definition	Process_A	Not allowed.	The name is used at a higher level than the namespace of the POU definition.
System_B	Function block definition	Process_B		
		Process_A	Not allowed.	The name is used in a declared namespace.
None	Function block definition	Process_A	Not allowed.	The name is used without a namespace.

The following notation is used in the namespace declaration for the *Process_A* function block definition.

System_B

Additional Information

You cannot set a namespace for a program name. However, you can declare namespaces for objects that are used in the algorithm of the program.

6-7-3 Procedure for Using Namespaces

Use the Sysmac Studio to set the namespaces and then declare them. Perform steps 1 and 2 when you create data types or when you create function definitions, function block definitions, or other objects. Declare a namespace with step 3 to use an object for which a namespace is set.

1. In the Data Type Editor, set the namespace for the data type.
2. Set the namespace in the properties of the function definition or function block definition.
3. In the Ladder Editor or ST Editor, declare the namespace in the properties of the function definition or function block definition.
4. Use the data types, function definitions, and function block definitions in the user program.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504-E1-03 or later) for specific procedures.

6-8 Libraries

This section describes the specifications of libraries. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504-E1-03 or later) for specific procedures.

Version Information

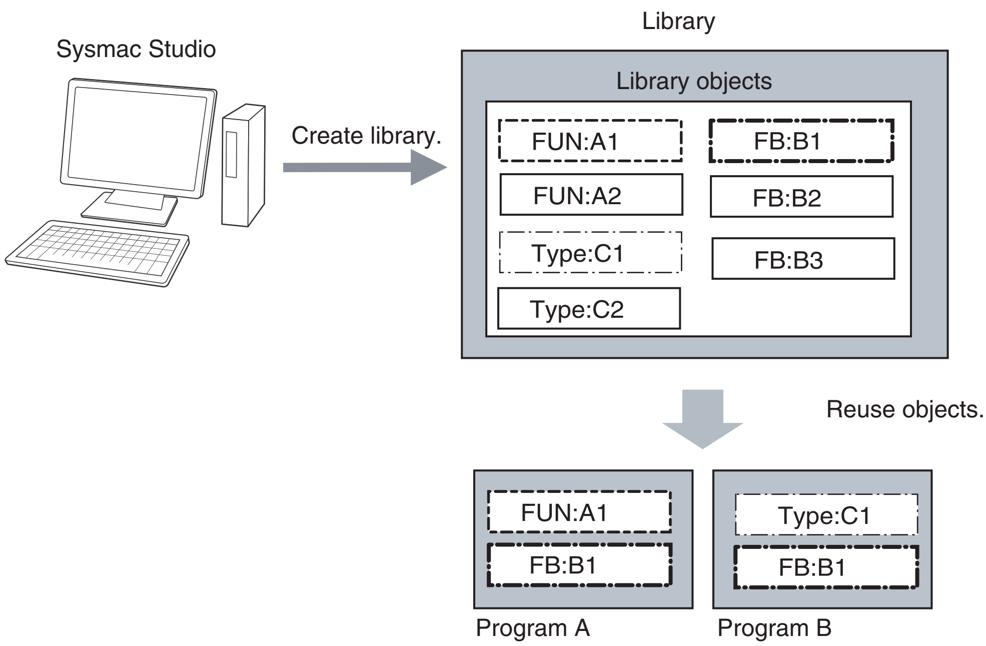
A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use libraries.

6-8-1 Introduction to Libraries

A library contains POU definitions and data types in a form that allows you to reuse them as objects in programming. The objects in a library are called library objects.

The NJ/NX-series CPU Units allow you to create and use libraries.

The following figure illustrates the use of library objects. Here, program A uses FUN:A1 and FB:B1 from the objects in the library, and program B uses Type:C1 and FB:B1.



6-8-2 Specifications of Libraries

This section describes the library settings and synchronization.

Library Settings

The following settings are supported for libraries.

Setting	Description
Name	The name of the library.
Version	The version of the library.
Author	The creator of the library. (optional)
Creation date	The date that the library was created.
Update date	The date that the library was last updated.
Comment	A comment on the library. (optional)
Company name	The name of the company that created the library. (optional)
ID	A unique ID that is used to access the library. The ID is generated automatically. You cannot change it.
Display/hide source	You can specify whether to display or hide the source.*
Attached files	You can attach one or more files.

* If data protection is set for a library object, a password is required to display the source code.

● Selecting Library Objects

You can select the objects to include in a library. You can also access other libraries to create library objects. When you do, you can select whether to include the library data from the accessed library.

Additional Information

When you select to include accessed library data, the accessing library is created so that it contains a copy of the accessed library data. This means that only one library file is required. However, if there is more than one accessing library, you must change each one of them to make any changes. When you select not to include accessed library data, the accessing library is created without the accessed library data. This means that there will be two library files, the accessing file and the accessed file. However, even if there is more than one accessing library, you need to change only the accessed library to make changes.

Library Synchronization

You can download a library to a Controller, upload a library from a Controller, or verify a Controller library against one on the computer.

Additional Information

- If you transfer a project for which transferring the source program is disabled from the Sysmac Studio to a Controller that contains libraries for which the source is displayed, the source data for the library is not transferred.
- The libraries in the Controller are deleted for the Clear All Memory operation.

6-8-3 Library Object Specifications

This section describes the library objects that can be created and the settings for the library objects.

Applicable Library Objects

You can handle the following entities as library objects.

Library objects	Details
POU definitions	Functions and function blocks
Data types*	Structure data types, union data types, and enumeration data types

* Data types are always included in the library object selections on the Sysmac Studio.

Library Object Settings

You can set the following for each library object.

Property	Definition
Name	The name of the library object.
Namespace	The namespace of the library object.
Version*	The version of the library object.
Author*	The creator of the library object. (optional)
Creation date*	The date that the library object was created.
Update date*	The date that the object library was last updated.
Comment	A comment on the library object. (optional)

* These items can be set only for functions and function blocks. They are set in the POU definition properties on the Sysmac Studio.

6-8-4 Procedure to Use Libraries

Use the following procedures to create and use libraries.

Procedure to Create Libraries

Create a project to use as the library. Use the following procedure to create and save a library.

1 Create a library project.

When you create the project, select a library project as the project type in the Project Window.

2 Create library objects.

In the library project, create the required POU definitions and data types, and then check them to make sure that they operate correctly.

3 Set the properties of the library.

Set the properties of the library project, including selecting the library objects, hiding/displaying source code, and attached files.

4 Save the project as a library file.

Save the project in a library file in the Create Library File Dialog Box.

Additional Information

- You can change an existing project to a library project as long as the only device that is registered in the project is a Controller. Simply change the project type in the project properties to a library project.
- You can create data that cannot be used as library objects in a library project. However, you cannot select any of this data as library objects.
- We recommend that you use namespaces for names of the functions, function block definitions, and data types that you create as library objects to prevent duplicating names with other libraries. For details on namespaces, refer to *6-7 Namespaces*.

Procedure to Use Libraries

You can read objects that are created in libraries into a project to use them in the user program. Use the following procedure to use libraries.

1 Specify the library.

Specify the library file to access in the Library Reference Dialog Box of the project in which to use the library objects.

2 Use the library objects in programming.

Use the library objects from the library that you read in the project. Use the library objects in the same way as any other functions, function block definitions, or data types.

6-9 Programming Precautions

This section describes precautions for developing a user program.

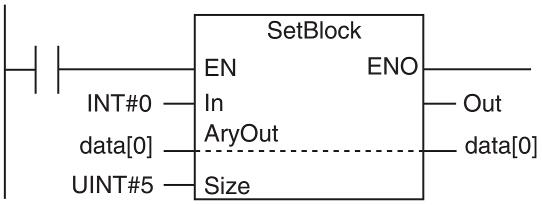
6-9-1 Array Specifications for Input Variables, Output Variables, In-Out Variables

Some instructions handle array variables.

Example:

Data Type Table

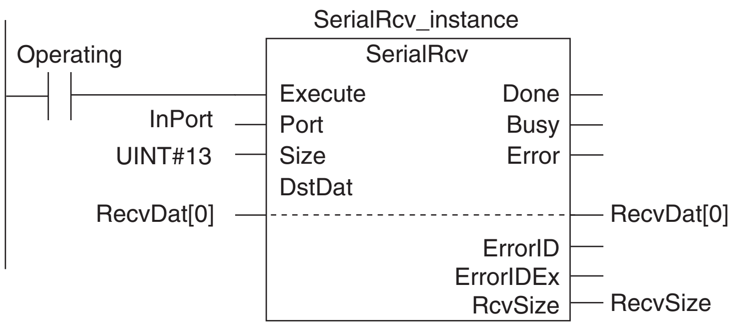
Variable name	Data type
data	ARRAY[0..4] OF INT



6-9-2 Structure Variables for Input Variables, Output Variables, In-Out Variables

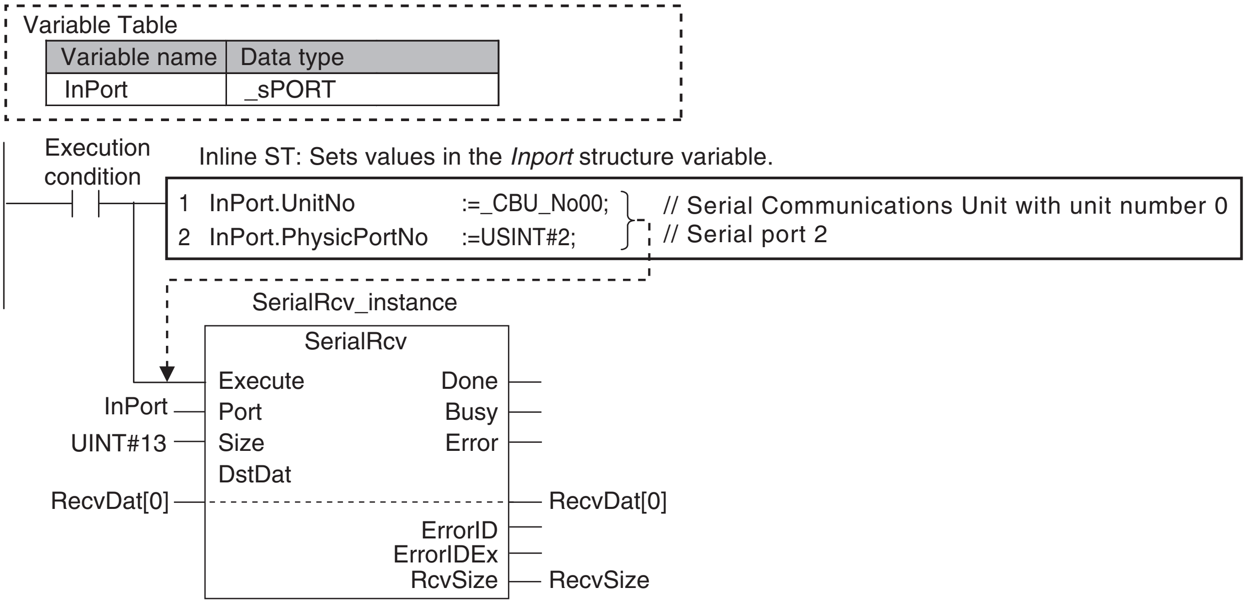
Some instructions have structure variables for input, output, or in-out variables.

Example:



In this case, you must create a structure variable for the input, output, and in-out parameters, then use the MOVE instruction to set the values.

Example:



6-9-3 Master Control

Introduction

Master control is used to make output FALSE for all processing between the MC (Master Control Start) instruction and the MCR (Master Control End) instruction. Master control is useful to control the execution conditions of a relatively long series of instructions.

Refer to information on the MC and MCR instructions in the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for details.

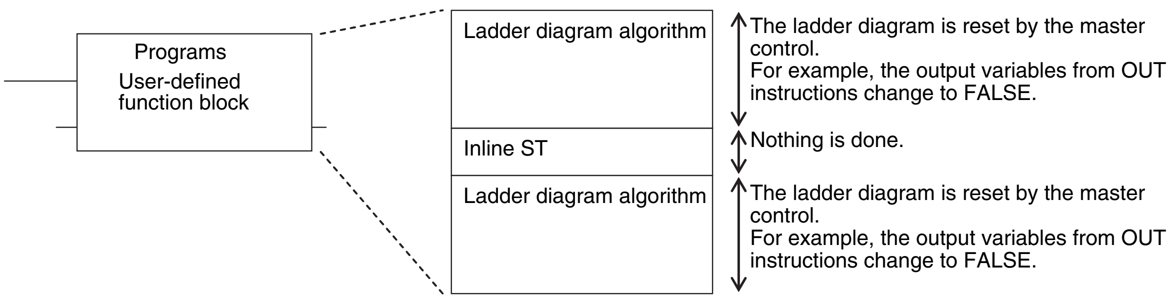
Master Control Programming Languages

You can use master control in ladder diagrams.

You cannot use master control with ST. You also cannot use master control for inline ST inside a ladder diagram.

Example:

Inside a Master Control Region:



Operation of Instructions That Are Reset in a Master Control Region

Refer to information on the MC and MCR instructions in the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for the operation of other instructions in the master control region when master control is reset.

7

Checking Operation and Actual Operation

This section describes the items and procedures for checking the operation of an NJ/NX-series Controller, including offline debugging procedures.

7-1 Overview of Steps in Checking Operation and Actual Operation

The shaded steps in the overall procedure that is shown below are related to the checking operation and actual operation. In *Step 2-4. Offline Debugging*, a simulation is used to check operation without going online with the Controller. In *Step 5. Checking Operation and Starting Operation on the Actual System*, you go online with the Controller to check the operation of the physical Controller. When checking operation is completed, you start actual operation.

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series Controller* for the overall procedure.

Step 1. Software Design

- Step 1-1 Designing I/O and Processing
- Step 1-2 Designing Tasks
- Step 1-3 Designing Programs

Step 2. Software Setups and Programming

- Step 2-1 Slave and Unit Configurations
- Step 2-2 Controller Setup
- Step 2-3 Programming
- Step 2-4 Offline Debugging

Step 3. Mounting and Setting Hardware

Step 4. Wiring

Step 5. Checking Operation and Starting Operation on the Actual System

7-2 Offline Debugging

This section describes how to use simulation to debug operation offline. You can simulate the operation of an NJ/NX-series Controller on a computer to check the operation of the user program with only the computer. There are also debugging operations that can be used during simulation that are not supported on the physical Controller. This makes user program development and debugging more efficient.

7-2-1 Features of Simulation

In the following way, simulation is more effective than going online with the Controller to debug operation.

- You can use breakpoints, step execution, pausing, and other functions to check program logic.
- You can select only specific programs to simulate to check only those programs.
- You can change the simulation execution speed to check operation at a slower speed than for actual operation.
- You can use the Task Execution Time Monitor to estimate the task execution times.
- You can use debugging programs to manipulate inputs from outside the Controller.

7-2-2 Simulation Execution

You can do the following for simulations.

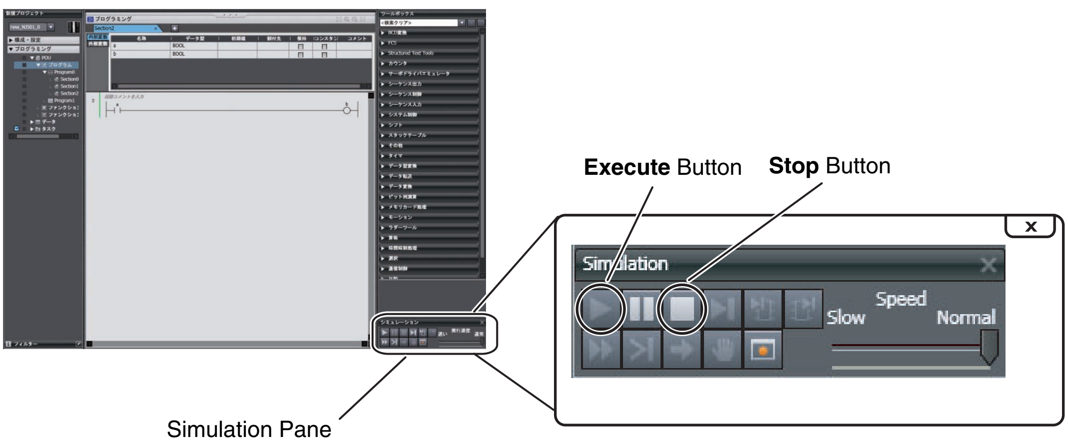
- Start and stop the Simulator.
- Check the logic of programs.
- Estimate task execution times.
- Use online debugging functions.

Starting and Stopping the Simulator

You perform simulations by starting the Simulator from the Simulation Pane of the Sysmac Studio. After you complete checking operation with the simulation, you stop the Simulator. The following procedure shows how to start and stop simulations.

1 Select ***Simulation Pane*** from the View Menu of the Sysmac Studio.

The Simulation Pane is displayed on the lower right of the window.



2 Click the **Execute** Button in the Simulation Pane.

The user program is transferred to the Simulator and the simulation starts. When a simulation starts, the Editors and other parts of the Sysmac Studio window will enter the same state as when the Sysmac Studio is online with the Controller.

3 After you complete checking operation, click the **Stop** Button in the Simulation Pane to stop the Simulator.

Checking the Logic of Programs

You can use simulation debugging to stop the operation of the Simulator or to execute a program one step at a time to check the validity of the program logic. You can perform the following operations with the buttons in the Simulation Pane.

Operation	Description of operation
Breakpoints	Use a breakpoint to specify a location in a program and pause program execution at that location.
Step execution	Use step execution to execute one line of an ST program or one instruction in a ladder diagram program and then pause the Simulator.
Continuous step execution	Use continuous step execution to continually perform step execution at a specified interval.
Pausing	Use pausing to pause execution of the simulation.
Step-in execution	Use step-in execution to perform step execution of source code inside a function or function block.
Step-out execution	Use step-out execution to execute the current function or function block to the end.
One-period execution	Use one-period execution to execute the current task for one period. Execution pauses at the beginning of the program in the next period.
Conditional breakpoints	Use conditional breakpoints to pause the execution of a program at a breakpoint when the specified stopping condition is met.

Estimating Task Execution Times

If you execute the Simulator in Execution Time Estimation Mode, the estimated task execution time from when task execution starts until it stops is displayed on the Task Execution Time Monitor Display. The average and maximum estimated task execution times are displayed. Refer to *Task Execution Timeout* on page 5-84 for the Task Execution Time Monitor Display.

Precautions for Correct Use

The estimated task execution times are not necessarily the same as the actual task execution times on the physical Controller. Depending on the user program, I/O configuration, and whether communications are used, the execution times on the physical control may exceed the estimated maximum value. Use them only as guidelines in task design. Always confirm the task execution times while connected to the physical Controller to study the designs and before starting actual system operation.

Online Debugging Functions

With the Simulator, you can use some of the functions for debugging that are supported when you are online with the Controller. The following table shows the differences between online debugging with the Controller or offline debugging with the Simulator. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details on the differences in debugging operations for the Controller and for the Simulator.

Debugging function	Controller	Simulator
Monitoring	Supported.	Supported.
Monitoring in a Watch Tab Page	Supported.	Supported.
Monitoring in the I/O Map	Supported.	Supported.
Differential monitoring	Supported.	Supported.
Controlling BOOL variables	Supported.	Supported.
Forced Refreshing (TRUE/FALSE/Cancel)	Supported.	Supported.
Changing present values of data	Supported.	Supported.
Clearing all memory	Supported.	Not supported.
Cross-reference pop-ups	Supported.	Supported.
Online editing	Supported.	Supported.
Monitoring Controller status	Supported.	Not supported.
Monitoring task execution status	Supported.	Supported.
Monitoring axis status (MC Monitor Table)	Supported.	Supported.
Changing the operating mode	Supported.	Not supported.
Resetting the Controller	Supported.	Not supported.
Data tracing	Supported.	Supported.
Setting triggers	Supported.	Supported.
Setting variables to sample	Supported.	Supported.
Starting and stopping tracing	Supported.	Supported.
Displaying trace results	Supported.	Supported.
Exporting trace results	Supported.	Supported.
Creating 3D equipment models	Supported.	Supported.
Displaying digital and analog charts	Supported.	Supported.
Displaying 3D axis paths	Supported.	Supported.
Monitoring task execution times	Supported.	Supported.
Estimating execution processing times	Not supported.	Supported.
Debugging with program simulations	Not supported.	Supported.
Setting simulation programs	Not supported.	Supported.
Changing the simulation speed	Not supported.	Supported.
Setting breakpoints	Not supported.	Supported.
Step execution	Not supported.	Supported.
Troubleshooting	Supported.	Supported.
Monitoring error information	Supported.	Supported.
Displaying error logs	Supported.	Supported.
Setting event tables	Supported.	Supported.
Monitoring user memory usage	Supported.	Supported.
Clock Information Settings	Supported.	Not supported.
Releasing access rights	Supported.	Not supported.

7-2-3 Setting Up Simulations

You set the following for simulations.

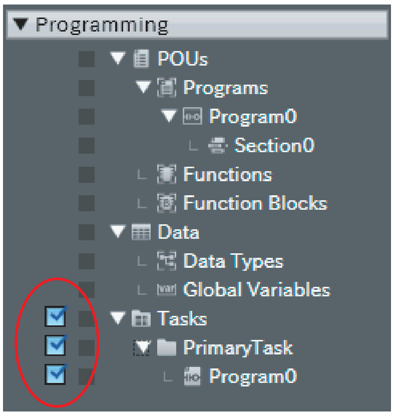
- Setting simulation programs
- Setting debug programs
- Setting the simulation speed

Setting Simulation Programs

You can set the task or programs to simulate. You can choose to simulate some or all of the programs in the user program. The following procedure shows how to set the simulation programs.

1 Display the Simulation Pane.

Check boxes are displayed to the left of the programs that are listed under **Tasks** in the Multiview Explorer project to designate programs for simulation.



2 Select the check boxes for the tasks or programs to simulate.

Setting Debug Programs

A debugging program is used to check operation with offline debugging. The debugging program contains instructions to perform virtual input processing on inputs received from outside of the Controller, force user-defined errors, and perform other such debugging tasks. You can execute debugging programs only on the Simulator.

The following procedure shows how to create debugging programs.

1 Right-click **Programs** under **Programming** – **POU** in the Multiview Explorer and select ***Add for Debugging*** – ***Multipart Ladder*** or ***Add for Debugging*** – ***Structured Text*** from the menu.

A debug program is created.

2 Enter the test program code into the debugging program that you just created.

3 Assign the debugging program to a task.

You can also change a normal program that is already completed into a debug program in the same way.

1 Right-click a program under **Programming** – **POU** – **Programs** in the Multiview Explorer and select ***SettingsForDebugging*** – ***Enable***.

Setting the Simulation Speed

You can use the Simulation Speed Slider in the Simulation Pane to change the simulation speed from 0.1x to 1x. You can change simulation speed while a simulation is in progress or when it is stopped. Use this to display the execution of the Simulator more slowly than for actual operation.

7-3 Checking Operation on the Actual System and Actual Operation

This section describes the procedures from checking operation on the actual system to starting actual operation.

7-3-1 Procedures

The procedures from checking operation on the actual system to starting actual operation are given below.

Step 1. Going Online from the Sysmac Studio and Downloading the Project	Reference
1. Turn ON the power supply to the Controller. 2. Place the Sysmac Studio online with the Controller. 3. Download the project (i.e., the user program, Unit configuration, and other settings) from the Sysmac Studio.	*Sysmac Studio Version 1 Operation Manual* (Cat. No. W504)

Step 2. Checking Operation on the Controller	Reference
1. In PROGRAM mode, check the I/O wiring by performing forced-refreshing with user-specified values from the I/O Map or Ladder Editor. 2. For motion control, use the MC Test Run operations to perform the following: check the wiring, jog to check the motor rotation directions, perform relative positioning to check the travel distances (e.g., for electronic gear settings), and check homing operation. 3. Change the Controller to RUN mode and check the operation of the user program.	*7-3-3 Checking I/O Wiring* *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507) *7-3-5 Checking the Operation of the User Program*

Step 3. Starting Actual Controller Operation	Reference
1. Confirm that operation is performed as designed and then start actual operation.	---

Additional Information

Use the synchronization function to download the project from the Sysmac Studio to the Controller. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for information on the synchronization function.

7-3-2 Downloading the Project

Use the following procedure to download the project from the Sysmac Studio to the physical Controller.

1 Go online with the Controller, and then select ***Synchronization*** from the Controller Menu.

The data on the computer and the data in the physical Controller are compared automatically.

2 Click the **Transfer to Controller** Button.

7-3-3 Checking I/O Wiring

Check the I/O wiring by using forced refreshing from the Watch Tab Page of the Sysmac Studio. You can write values to I/O for Units or slaves to check the results to test the I/O wiring. Refer to *8-5-1 Forced Refreshing* for information on forced refreshing.

7-3-4 MC Test Run

The MC Test Run function is used mainly to perform the following operations from the Sysmac Studio without a user program.

- Checking wiring: You can monitor Servo Drive connector I/O signals and Servo Drive status.
- Checking the operation and direction of the motor: You can turn ON the Servo and jog axes.
- Checking electronic gear settings: You can perform relative positioning, and check and change travel distances.
- Checking homing: You can check the homing operation.

Connect online to the Controller from the Sysmac Studio and perform the MC Test Run on the MC Test Run Tab Page.

For details, refer to the *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

Use the following procedure.

1 After you complete the necessary wiring, connect the Sysmac Studio online to the Controller.

2 Create axes, assign the axes, and set the following axis parameters.

Axis parameter settings required for an MC Test Run: Unit of Display, Command Pulses Per Motor Rotation, Travel Distance Per Motor Rotation, Maximum Velocity, Maximum Jog Velocity, Maximum Acceleration Rate, Maximum Deceleration Rate, Software Limit Function Selection, Software Limits, and Count Mode

3 Open the MC Test Run Tab Page and perform the following.

Example:

- Monitoring and checking wiring
- Jogging to check the direction of the motor
- Check travel distances for relative positioning (electronic gear settings).
- Confirming the homing operation

7-3-5 Checking the Operation of the User Program

To check the operation of the user program on the actual system, change the operating mode of the CPU Unit to RUN mode. You can use the following to check operation.

- Checking the operation of the user program
- Correcting the user program with online editing
- Checking the operation of the user program with data tracing

Checking the Operation of the User Program

You can perform the following to check the operation of the user program.

1. Monitor the execution status of the user program.
2. Check the operation by changing the status of program inputs and program outputs, and the values of variables.

Monitoring the Execution Status of the User Program

You can monitor the TRUE/FALSE status of program inputs and outputs and the present values of variables in the Controller. You can monitor the status on the Ladder Editor, Watch Tab Page, or I/O Map of the Sysmac Studio.

Checking the Operation by Changing the Status of Program Inputs and Program Outputs, and the Values of Variables

You can change the TRUE/FALSE status of program inputs and outputs and the present values of variables in the user program to see if the user program operates as designed. Use forced refreshing to change the status of program inputs and program outputs. Use one of the methods to change the present values of variables. Refer to *8-5-1 Forced Refreshing* and *8-5-2 Changing Present Values* for details.

Correcting the User Program with Online Editing

You can use online editing to correct a user program that you determined needs to be corrected while checking operation. You can use online editing to change a user program without stopping the operation of the CPU Unit. Refer to *8-5-3 Online Editing* for details.

Checking Operation with Data Tracing

You can use data tracing to check when program inputs and program outputs are changed to TRUE or FALSE and to check changes in the values of variables. Refer to *8-5-4 Data Tracing* for details.

7-3-6 Starting Actual Operation

Change the operating mode to RUN mode to start actual operation. Check the user program, data, and parameter settings sufficiently for proper execution before you use them for actual operation.

8

CPU Unit Functions

This section describes the functionality provided by the CPU Unit.

8-1 Data Management, Clock, and Operating Functions

This section describes the data management, clock, and operating functions.

8-1-1 Clearing All Memory

You can initialize the user program, Controller Configurations and Setup, and variables in the CPU Unit to the defaults from the Sysmac Studio. This is called the Clear All Memory operation.

Precautions for Correct Use

- The Clear All Memory operation can be performed only in PROGRAM mode.
- You cannot execute the Clear All Memory operation when write protection of the CPU Unit is set in the security functions.
- Do not turn OFF the power supply to the Controller during the Clear All Memory operation.

After you clear the memory, the CPU Unit operates in the same way as immediately after you create the system configuration with the CPU Unit in the factory default condition.

The absolute encoder home offset is not cleared.

● Operations from the Sysmac Studio

Connect the Sysmac Studio to the CPU Unit online, and select ***Clear All Memory*** from the Controller Menu.

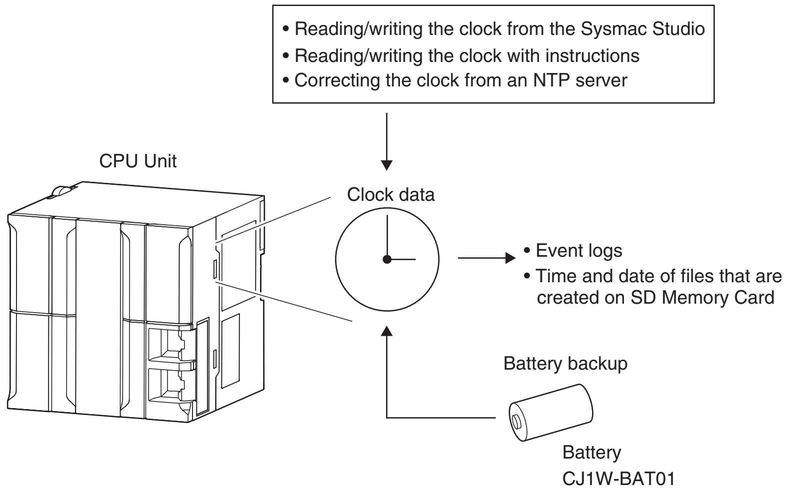
8-1-2 Clock

Introduction

A clock (RTC) is built into the CPU Unit. The clock data from this clock is used for timestamps in the event logs and for the time and date of files that are created on the SD Memory Card.

The following functions are supported.

- Reading/writing the clock from the Sysmac Studio
- Reading/writing the clock with instructions
- Reading the clock from system-defined variables (Writing is not possible.)
- Correcting the clock from an NTP Server



Precautions for Correct Use

The clock data is retained by the Battery when the power is turned OFF. The clock data is not correct when the power is turned ON. You can reset the clock data from an NTP server over an EtherNet/IP network after the power is turned ON.

Clock Data Range

- NX-series CPU Units:1970-01-01 to 2069-12-31 (January 1, 1970 to December 31, 2069).
- NJ-series CPU Units:1970-01-01 to 2106-02-06 (January 1, 1970 to February 6, 2106).

Setting the Time Zone and the Local Time

Before you use the CPU Unit for the first time, set the time zone and local time in the clock data. You can set the time zone and local time from the Sysmac Studio in the Controller Clock Dialog Box.
The clock data that is read by the EtherCAT slaves and CJ-series CPU Units from the CPU Unit and the clock data that is set are the local times in the time zone.

Additional Information

When a Battery is not mounted or when the Battery voltage is low, the time zone setting is retained, but the clock data is not retained and will not be correct.

Setting the Clock Data

Use one of the following methods.

Changing Clock Data from the Sysmac Studio

You can use the Sysmac Studio to synchronize the clock data of the built-in clock with the clock on the computer.

Changing Clock Data with Instructions

You can use the SetTime instruction to set the clock data.

Changing the Clock Data from an NTP Server

You can use an NTP server on EtherNet/IP to set the clock data.

Correcting the Clock from an NTP Server

Application

In a network system, the clock data must be shared by the entire system. NTP is supported to enable easy time synchronization.

Specifications

An NTP client is provided.

Refer to the *NJ/NX-series Built-in EtherNet/IP User's Manual* (Cat. No. W506) for details.

Reading the Clock Data

If the clock data is incorrect, the incorrect value is read.

Reading the Clock Data from Instructions

You can use the GetTime instruction to read the clock data from the user program.

Reading the Clock from System-defined Variables (Writing Is Not Possible)

You can use the following system-defined variable to read the clock data.
_CurrentTime (System Time)

Sysmac Studio Procedure

You can select ***Controller Clock*** from the Controller Menu of the Sysmac Studio to display the clock data.

Logging

When you change the clock data, an event is recorded in the event log. However, nothing is recorded in the event log if the time is corrected for the NTP.

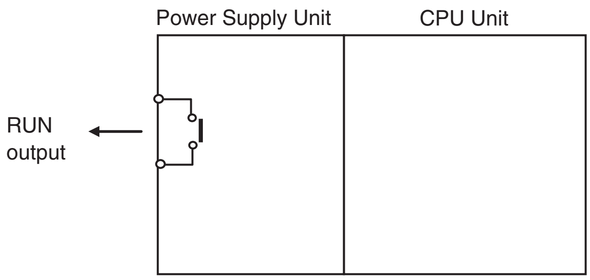
Related System-defined Variables

Variable name	Meaning	Description	Data type	R/W
_CurrentTime	System Time	This variable contains the CPU Unit's internal clock data.	DATE_AND_TIME	R

8-1-3 RUN Output

Introduction

The RUN output on NX-PA9001, NX-PD7001, or NJ-P□3001 Power Supply Unit is ON while the CPU Unit is operating.



The RUN output operates as shown in the following table.

Status	Operation
During RUN mode	ON
Startup state **(until RUN mode is entered according to the Startup Mode setting).**	OFF
During PROGRAM mode	
When a major fault level Controller error occurs	

The ratings of the RUN output on NX-PA9001, NX-PD7001, or NJ-P□3001 Power Supply Unit are as follows:

Item	Description
Contact form	SPST-NO
Switching capacity	2 A at 250 VAC for resistive load 0.5 A at 120 VAC for inductive load 2 A at 24 VDC for resistive load

Application

You can use the RUN output for the following purposes:

- Obtain a signal to notify the host that the CPU Unit is functioning normally and is currently operating.
- Synchronize the completion of startup of more than one CPU Unit
- Release interlocks when the CPU Unit starts operation.

Precautions for Safe Use

It takes up to approximately 10 to 20 s to enter RUN mode after the power is turned ON. The outputs during this time behave according to the slave or Unit specifications. Use the RUN output on the Power Supply Unit, for example, to implement fail-safe circuits so that external devices do not operate incorrectly.

8-2 Management Functions for CJ-series Units

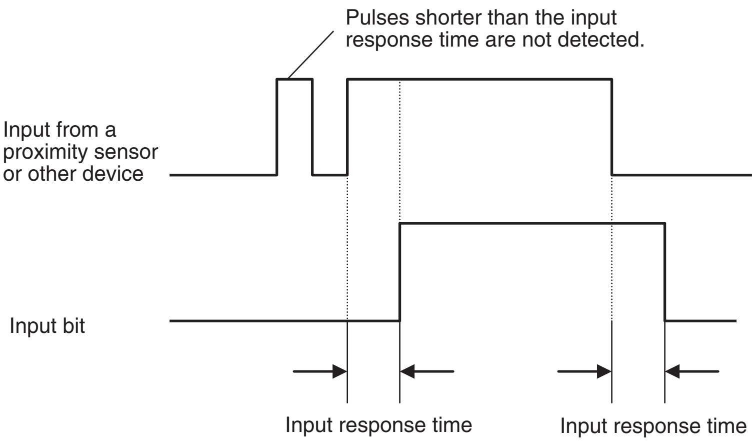
This section describes the management functions used for Units in the Controller.

You can use CJ-series Units only with NJ-series CPU Units.

8-2-1 Basic I/O Units

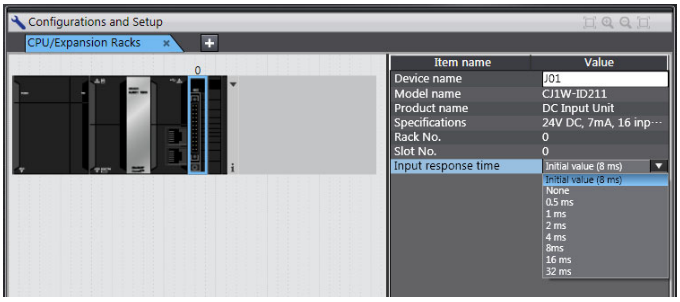
Introduction

You can increase the input response time to reduce chattering and the effects of external noise. You can decrease the input response time to enable detection of shorter input pulses. Do not set the ON response time or OFF response time to less than the refresh time.



Setting Methods

From the Multiview Explorer of the Sysmac Studio, double-click **CPU/Expansion Racks** under **Configurations and Setup**. Then select the input response times in the Unit information for the Basic I/O Units.



You must do either of the following to enable the settings.

- Cycle the power supply to the Controller.
- Reset the Controller (the entire CPU Unit) from the Sysmac Studio.

Related System-defined Variables

The set values for the input response times of the Basic Input Units are output to the following system-defined variable.

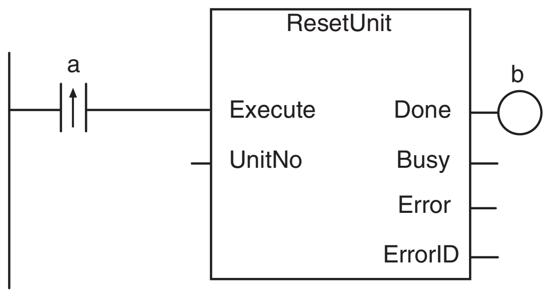
Variable name	Meaning	Description	Data type	R/W
_CJB_InRespTm	Basic Input Unit Input Response Times	Contains the response times of the Basic I/O Units in 0.1-ms increments.	ARRAY[0..3, 0..9]OF UINT	R

8-2-2 Special Units

Restarting Special Units

You can restart a Special Unit (Special I/O Unit or CPU Bus Unit) to enable values that are set for it. If you restart a Special Unit, you do not have to cycle the power supply to the Controller. Execute the following ResetUnit (Restart Unit) instruction to restart Special Units.

Instruction name	Instruction	Description
Restart Unit	ResetUnit	Restarts the CPU Bus Unit or Special I/O Unit.



The ResetUnit instruction restarts a Special Unit across multiple task periods when execution condition *a* changes to TRUE. If the restart ends normally, the output variable *Done* (normal end) changes to TRUE and variable *b* therefore changes to TRUE.

If Special Unit settings are changed in any of the following ways, you must restart the Special Unit or cycle the power supply to the Controller.

- Editing from the Special Unit Settings Tab Page of the Sysmac Studio
- Editing from the I/O Map or Watch Tab Page
- Setting the user program

Related System-defined Variables

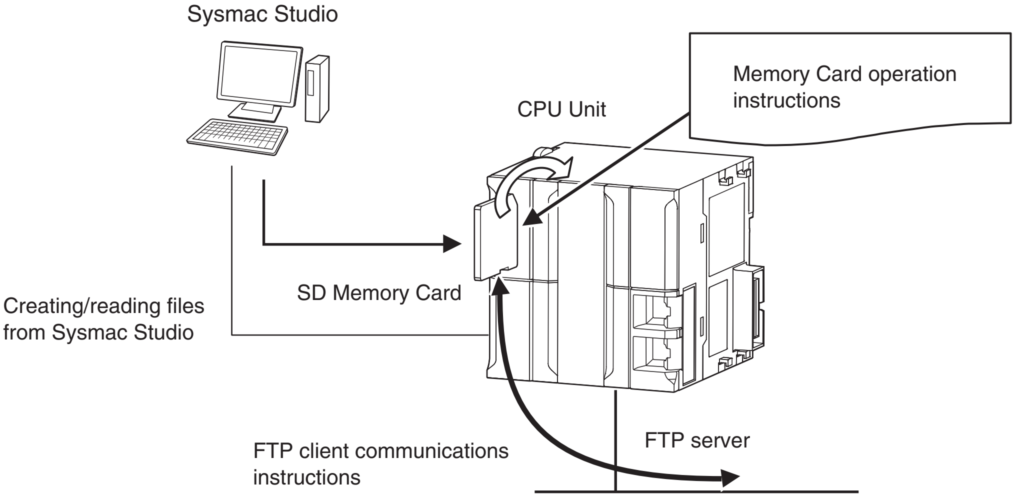
Variable name	Meaning	Description	Data type	R/W
_CJB_CBU00 InitSta to _CJB_CBU15 InitSta	CPU Bus Unit Initializing Flags	The corresponding variable is TRUE during initialization of the CPU Bus Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units.	BOOL	R
_CJB_SIO00InitSta to _CJB_SIO95InitSta	Special I/O Unit Initializing Flags	The corresponding variable is TRUE during initialization of the Special I/O Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units.	BOOL	R
_CJB_CBU00 Restart to _CJB_CBU15 Restart	CPU Bus Unit Restart Bits	The CPU Bus Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the CPU Bus Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Bit to TRUE with an instruction, the restart process begins from refresh processing in the next task period after the instruction is executed.	BOOL	RW
_CJB_SIO00 Restart to _CJB_SIO95 Restart	Special I/O Unit Restart Bits	The Special I/O Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the Special I/O Unit is restarted.) The numbers in the variable names indicate the unit numbers of the applicable Units. If you change the Restart Bit to TRUE with an instruction, the restart process begins from refresh processing in the next task period.	BOOL	RW

8-3 SD Memory Card Operations

This section describes the functions that you can use for SD Memory Cards.

8-3-1 SD Memory Card Operations

The NJ/NX-series CPU Unit supports the following functions for SD Memory Cards.



Function	Introduction
SD Memory Card operation instructions	You can access SD Memory Cards from instructions in the user program.
FTP client communications instructions*1	You can use these instructions to transfer files via FTP from the CPU Unit to computers or Controllers at Ethernet nodes.
FTP server	You can use FTP commands from an FTP client on the Intranet to read and write large files in the SD Memory Card through EtherNet/IP.
File operations from the Sysmac Studio	You can perform file operations from the Sysmac Studio for the SD Memory Card inserted in the CPU Unit. You can perform file operations for Controller files in the SD Memory Card and save standard document files on the computer.
SD Memory Card life expiration detection	Notification of the expiration of the life of the SD Memory Card is provided in a system-defined variable and event log.
SD Memory Card backup and automatic transfer from SD Memory Card	You use the SD Memory Card inserted in the CPU Unit to backup, restore, and verify user programs and data in the Controller. Refer to *9-2 SD Memory Card Backups* and *9-4 Automatic Transfers from SD Memory Cards* for details.

*1 A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the FTP client communications instructions.

8-3-2 Specifications of Supported SD Memory Cards, Folders, and Files

SD Memory Card Specifications

The NJ/NX-series Controllers support both SD cards and SDHC cards. However, operation was confirmed only for the OMRON SD Memory Card given in the following table. Correct operation may not be possible if you use any other SD or SDHC card.

Model	Card type	Capacity [GB]	Formatting	Number of overwrites	Write protection
HMC-SD291	SD card	2	FAT16	100,000 overwrites	You can write-protect the SD Memory Card with a hardware switch on the Card.
HMC-SD491	SDHC card	4	FAT32		

The system-defined variable *_Card1Err* (SD Memory Card Error Flag) changes to TRUE (observation level) in the following cases.

- When there is a format error

If an error occurs, the SD PWR indicator on the front of the CPU Unit goes out, and accessing the SD Memory Card will not be possible.

Folder and File Specifications

● Character Restrictions

Object named by user	Usable characters	Reserved words	Multibyte character compatibility	Case sensitivity	Maximum size (without NULL)
Volume label	0 to 9, A to Z, and a to z, as well as % - _ @ ! ‘ ( ) ~= # & + ^ [ ] { } , . ; and single-byte kana*1	CON, PRN, AUX, CLOCK$, NUL, COM0, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT0, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9	Not supported.*2	Case insensitive	11 bytes
Directory name	0 to 9, A to Z, and a to z, as well as $ % ’ - _ @ ! ‘ ( ) ~= # & + ^ [ ] { } , . ; and single-byte kana				65 bytes
File name					65 bytes

*1 You cannot begin volume label names with a space.

*2 Even if the computer supports multibyte characters (e.g., for Japanese), you cannot use them in the CPU Unit.

● Subdirectory Levels

You can create up to 5 levels (example: f1/f2/f3/f4/f5/abc.txt)

Maximum Number of Stored Files

The number of files that you can store on an SD Memory Card depends on the directory level in which you store the files. The maximum number of files for each is given in the following table. However, the values in the table assume that 8.3 filename is used. If you use long file names, the maximum number of stored files is less than the value given in the following table.

Directory level	Format	Maximum number of stored files
Root directory	FAT16	511
	FAT32	65,533
Subdirectory	FAT16, FAT32	65,533

Maximum Size of One File

The maximum size of any one file is 2,147,483,647 bytes (2 GB –1 byte).

8-3-3 SD Memory Card Operation Instructions

You can perform various operations on the SD Memory Card by using the following instructions.

Instruction name	Instruction	Description
Read Variable from File	**FileReadVar**	The FileReadVar instruction reads the contents of a binary file on the SD Memory Card and writes it to the specified variable. You can specify array and structure variables.
Write Variable to File	**FileWriteVar**	The FileWriteVar instruction writes the value of a specified variable to a binary file in the SD Memory Card. You can specify array and structure variables. If the directory specified for the file name does not exist, it is created.
Open File	**FileOpen**	The FileOpen instruction opens the specified file.
Close File	**FileClose**	The FileClose instruction closes the specified file.
Seek File	**FileSeek**	The FileSeek instruction sets a file position indicator in the specified file.
Read File	**FileRead**	The FileRead instruction reads the data from the specified file.
Write File	**FileWrite**	The FileWrite instruction writes data to the specified file.
Get Text String	**FileGets**	The FileGets instruction reads a text string of one line from the specified file.
Put Text String	**FilePuts**	The FilePuts instruction writes a text string of one line to the specified file.
Delete File	**FileRemove**	The FileRemove instruction deletes the specified file from the SD Memory Card.
Change File Name	**FileRename**	The FileRename instruction changes the name of the specified file or directory.
Copy File	**FileCopy**	The FileCopy instruction copies the specified file to a different file.
Create Directory	**DirCreate**	The DirCreate instruction creates a directory in the SD Memory Card.
Delete Directory	**DirRemove**	The DirRemove instruction deletes a directory from the SD Memory Card.

8-3-4 FTP Client Communications Instructions

FTP client communications instructions are used to transfer files via FTP from an NJ/NX-series CPU Unit to computers or Controllers at Ethernet nodes.

The files on the SD Memory Card are read and written when the following instructions are executed.

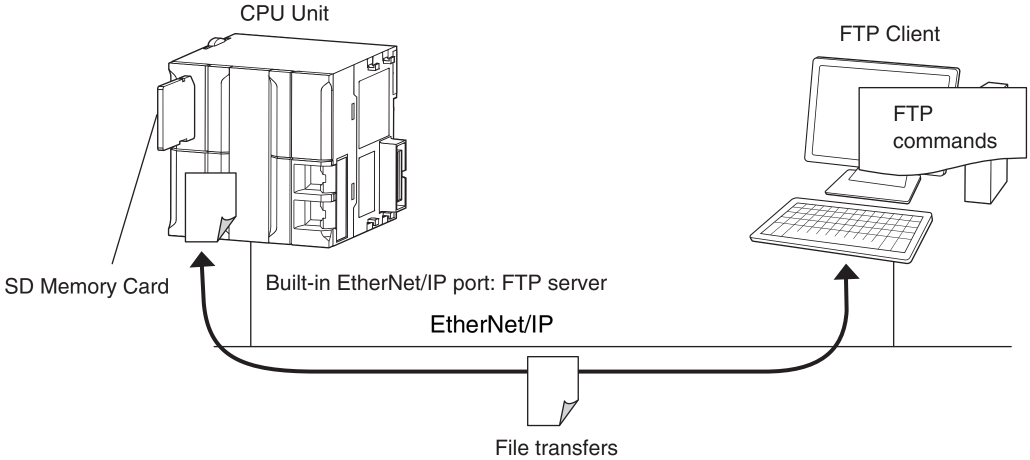
Instruction name	Instruction	Description
Put File onto FTP Server	**FTPPutFile**	The FTPPutFile instruction uploads one or more files in the FTP client's SD Memory Card to the FTP server.
Get File from FTP Server	**FTPGetFile**	The FTPGetFile instruction downloads one or more files from the FTP server to the FTP client's SD Memory Card.

Version Information

A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the FTP client communcations instructions.

8-3-5 FTP Server

You can read and write files on the SD Memory Card via EtherNet/IP by sending FTP commands to the built-in EtherNet/IP port from an FTP client.
Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP User's Manual* (Cat. No. W506) for details.



8-3-6 File Operations from the Sysmac Studio

You can perform file operations from the Sysmac Studio for the SD Memory Card inserted in the CPU Unit. In addition to Controller files, you can also store document files or other files on the SD Memory Card.

8-3-7 SD Memory Card Life Expiration Detection

You can determine the remaining life of the SD Memory Card before the Card becomes physically deteriorated.
You can determine the remaining life of the SD Memory Card with the following functions.

- System-defined variable *_Card1Deteriorated* (SD Memory Card Life Warning Flag)
- SD Memory Card Life Exceeded (Observation) record in the event log

The life of the SD Memory Card is checked when the power is turned ON and periodically while the SD Memory Card is inserted.
When the end of the life of the SD Memory Card is detected, save the data on the SD Memory Card and replace the SD Memory Card.

8-3-8 List of System-defined Variables Related to SD Memory Cards

The following system-defined variables show the status of the SD Memory Card.

Variable name	Meaning	Description	Data type
_Card1Ready	SD Memory Card Ready Flag	TRUE when the SD Memory Card is recognized. It is FALSE when an SD Memory Card is not recognized. TRUE: The Card can be used. FALSE: The Card cannot be used.	BOOL
_Card1Protect	SD Memory Card Write Protected Flag	TRUE when the SD Memory Card is write-protected. TRUE: Write protected. FALSE: Not write protected.	BOOL
_Card1Err	SD Memory Card Error Flag	TRUE when an unusable SD Memory Card is inserted or a format error occurs. TRUE: There is an error FALSE: There is no error	BOOL
_Card1Access	SD Memory Card Access Flag*1	TRUE during SD Memory Card access. TRUE: Card is being accessed. FALSE: Card is not being accessed.	BOOL
_Card1Deteriorated	SD Memory Card Life Warning Flag	TRUE when the end of the life of the SD Memory Card is detected. TRUE: The end of the life of the Card is detected. FALSE: The end of the life of the Card was not detected.	BOOL
_Card1PowerFail	SD Memory Card Power Interruption Flag*2	TRUE when the power supply to the Controller was interrupted during access to the SD Memory Card. TRUE: Power was interrupted during SD Memory Card access. FALSE: Operation is normal.	BOOL

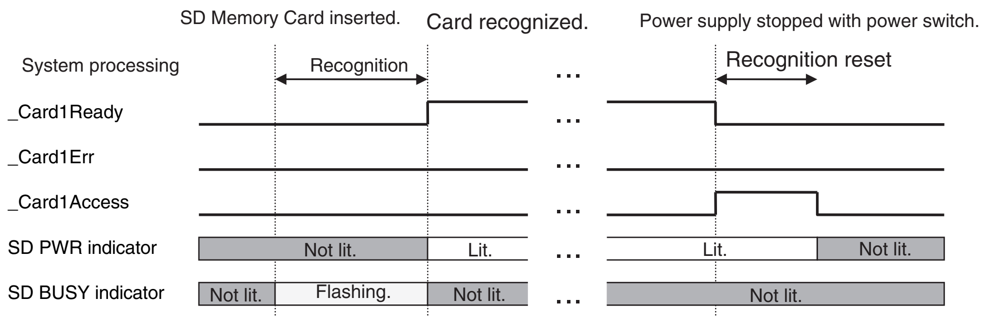
*1 Precaution When Using SD Memory Card Access Flag (*_Card1Access*)
The SD Memory Card Access Flag is intended for use in notifying external devices. The status of access to the SD Memory Card is not updated in realtime. Because of this, do not use the flag in the user program. Because the status of access to the SD Memory Card is not shown in realtime, it may cause unexpected Controller operation if you use it in the user program.

*2 Precautions When Using the SD Memory Card Power Interruption Flag (*_Card1PowerFail*)
If the SD Memory Card Power Interruption Flag is TRUE, check to see if the correct file is in the SD Memory Card and to see if the SD Memory Card operates properly. If the correct file is missing or the SD Memory Card does not operate properly, download the correct file to the SD Memory Card again. Cycle the power supply to the Controller or reset the Controller, and then see if the SD Memory Card operates properly. When you are finished, change SD Memory Card Power Interruption Flag to FALSE. (*_Card1PowerFail* does not change to FALSE automatically.)

Note Refer to *9-2 SD Memory Card Backups* for the system-defined variables that are used with the SD Memory Card backup function.

Additional Information

SD Memory Card Recognition and Unmounting Timing Chart



8-3-9 SD Memory Card Self-diagnostic Functions

You can perform self-diagnosis on the inserted SD Memory Card when the power supply is turned ON.

You can select whether to perform self-diagnosis when the power is turned ON in the Operation Settings of the Controller Setup under the Configurations and Setup from the Sysmac Studio as shown below.

- File system check
- Check equivalent to CHKDSK
- Restoration attempt when check fails

Access point	Setting group	Setting	Description	Set values
Operation Settings, Operation Settings Tab, Basic Settings	SD Memory Card Settings	Memory Card Diagnosis at Startup *1	Sets whether to execute self-diagnosis (file system check and restoration) on the inserted SD Memory Card when the power is turned ON.	Do not check. Check.

*1 Self-diagnosis is not executed if write protection is set on the SD Memory Card itself.

Results of Self-diagnosis

Case	Indicators			Error type	Correction	Remarks
	RUN	SD PWR	SD BUSY			
Self-diagnosis in progress	Flashing.	Not lit.	Lit.	---	---	---
1. When self-diagnosis found no problems	---	Lit.	Not lit.	Normal	None	---
2. The format of the SD Memory Card is not correct.	---	Not lit.	Not lit.	Observation	Use the Sysmac Studio to format the SD Memory Card.	---
3. An error was detected during the file system check and the file system was automatically restored.	---	Not lit.	Flashes during restore operation. Not lit after restore operation is completed.	Observation	Use file operations in the Sysmac Studio or insert the SD Memory Card into the computer to check whether any files were deleted by the restore operation.	If a corrupted file is detected, an attempt is made to restore the file.
4. The SD Memory Card failed.	---	Not lit.	Not lit.	Observation	Replace the SD Memory Card.	---

Precautions for Correct Use

If the recovery function is activated at startup, time is required to enter RUN mode. During that time, outputs will be OFF and external communications are not performed. Use the RUN output on the Power Supply Unit, for example, to implement fail-safe circuits so that external devices do not operate incorrectly.

Precautions for Safe Use

Never interrupt the power supply to the Controller during SD Memory Card access. That includes when SD Memory Card self-diagnosis at startup is enabled. An attempt is made by the SD Memory Card restoration function to restore any corrupted files. If the restoration fails, these files may be deleted automatically at startup.

8-3-10 Exclusive Control of File Access in SD Memory Cards

Access to files on the SD Memory Card is possible with the following methods.

(1) FTP server

(2) SD Memory Card operation instructions

(3) FTP client communications instructions

(4) File operations from the Sysmac Studio

However, if the same file on the SD Memory Card is accessed from different sources, unintended operations such as reading a file while it is being written or writing a file while it is being read may occur.

It is necessary to perform exclusive controls in order to prevent multiple accesses to the same file.

The following table shows the combinations of operations that require exclusive controls.

When you use a combination of operations that requires exclusive controls, execute the later processing only after checking that the first processing is finished.

<table>
<tr><th colspan="3" rowspan="3"></th><th colspan="6">First access</th></tr>
<tr><th colspan="2">Instructions*1</th><th colspan="2">File Operations from the Sysmac Studio</th><th colspan="2">FTP server</th></tr>
<tr><th>Reading</th><th>Writing</th><th>Reading</th><th>Writing</th><th>Reading</th><th>Writing</th></tr>
<tr><th rowspan="6">Later access</th><th rowspan="2">Instructions</th><th>Reading</th><td colspan="2" rowspan="2">Exclusive control is performed automatically, and an error occurs for the instruction that is executed later.</td><td>Exclusive control is not required.</td><td rowspan="2">Perform exclusive control.</td><td>Exclusive control is not required.</td><td rowspan="4">Perform exclusive control.</td></tr>
<tr><th>Writing</th><td>Perform exclusive control.</td><td>Perform exclusive control.</td></tr>
<tr><th rowspan="2">File operations from the Sysmac Studio</th><th>Reading</th><td>Exclusive control is not required.</td><td>Perform exclusive control.</td><td>---</td><td>---</td><td>Exclusive control is not required.</td></tr>
<tr><th>Writing</th><td colspan="2">Perform exclusive control.</td><td>---</td><td>---</td><td>Perform exclusive control.</td></tr>
<tr><th rowspan="2">FTP server</th><th>Reading</th><td>Exclusive control is not required.</td><td>Perform exclusive control.</td><td>Exclusive control is not required.</td><td>Perform exclusive control.</td><td>---</td><td>---</td></tr>
<tr><th>Writing</th><td colspan="4">Perform exclusive control.</td><td>---</td><td>---</td></tr>
</table>

*1 The instructions include the SD Memory Card operation instructions and the FTP client communications instructions.

8-4 Security

This section describes the security functions that are supported by the NJ/NX-series Controller.

To protect your assets, you can use security functions to protect the user program and data in the Controller. To prevent incorrect operation, you can use security functions to restrict operations on the Sysmac Studio.

The NJ/NX-series Controller supports the following security functions.

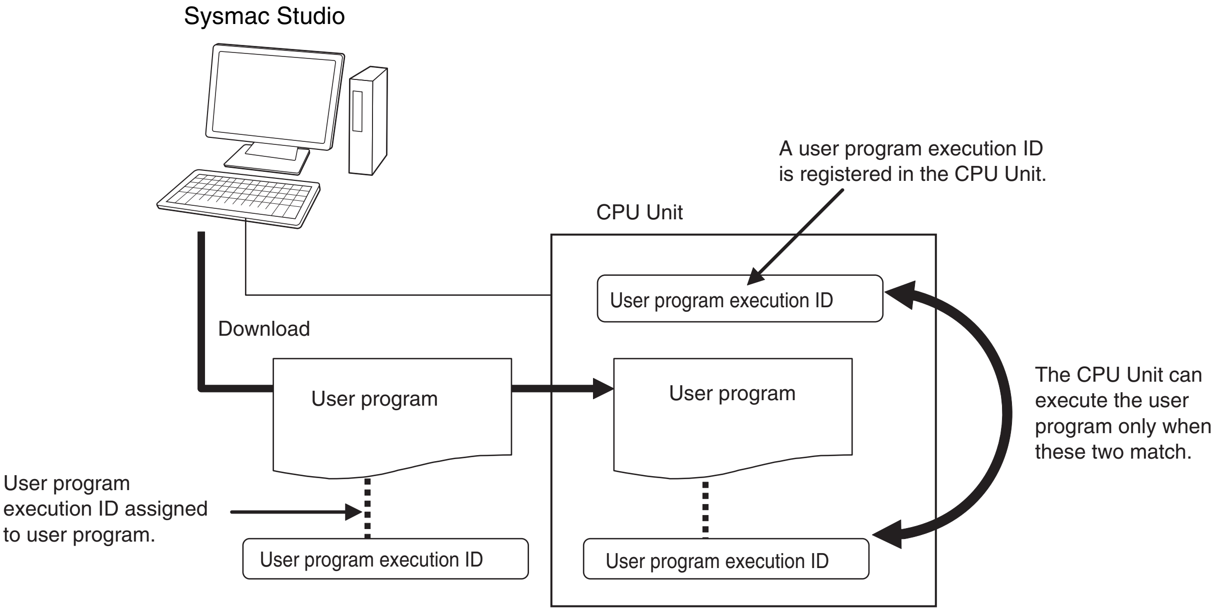
Application	Security function	Outline of function	Reference
Prevention of the theft of assets	Authentication of user program execution IDs	This ensures that a user program cannot be operated on another CPU Unit even if the user program is copied.	8-20
	User program transfer with no restoration information	You can transfer the user program to the CPU Unit without the source code. This prevents anyone from displaying the user program on another computer even if they upload it.	8-23
	Overall project file protection	You can place a password on a project file to protect your assets.	8-24
	Data protection*	You can place protection on part of the data in a project file to protect your assets.	8-25
Prevention of incorrect operation	Operation authority verification	You can set operation authorities to restrict the operations that can be performed on the CPU Unit from the Sysmac Studio.	8-27
	CPU Unit write protection	You can prevent rewriting data in the Controller from the Sysmac Studio.	8-29
Prevention of incorrect connections	CPU Unit names	You can check to see if the CPU Unit name and serial ID on the computer and in the Controller are the same to prevent going online with the wrong Controller.	8-30

* A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required.

8-4-1 Authentication of User Program Execution IDs

Introduction

You can set a specific ID (called a user program execution ID) in the CPU Unit in advance. If you do, you can execute only a user program with the same ID.

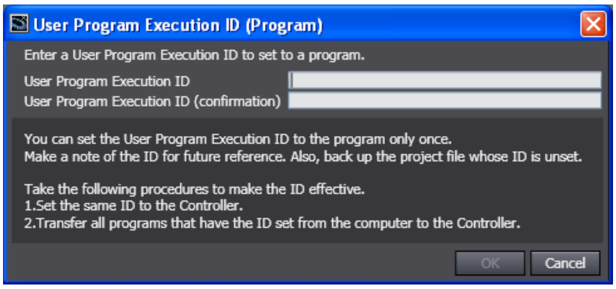


You can therefore prevent different CPU Units (hardware) from executing a user program.

In contrast to the protection function, you can still display and edit the user program even if a user program execution ID is set.

Operating Procedure

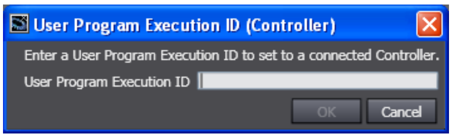
1. Always backup the project files before you assign a user program execution ID.
2. Assign the user program execution ID to the user program offline from the Sysmac Studio.



Precautions for Correct Use

After you assign a user program execution ID to a user program, you cannot change or delete the ID. To use a different ID, read the project file without an ID that was backed up in step 1, above, and assign another user program execution ID. To delete the ID, use the project file without an ID that was backed up in step 1, above.

3 Connect the Sysmac Studio online and register the user program execution ID that was set in step 2 in the CPU Unit.



The registration of the user program execution ID in the CPU Unit is recorded in the event log. At this time, the user program execution ID in the CPU Unit is overwritten even if it is already registered.

4 Transfer the user program with the same user program execution ID to the CPU Unit.

If the user program execution ID in the user program does not match the user program execution ID in the CPU Unit or if one of them does not have an ID, an ID Verification Error (major fault level Controller error) occurs when you attempt to change to RUN mode and the CPU Unit will not operate.

Precautions for Correct Use

After you assign a user program execution ID to the CPU Unit, you cannot read or delete the ID. To delete the ID from the CPU Unit, perform the Clear All Memory operation on the CPU Unit.

● Operation When an ID Verification Error Occurs

When the User Program Execution ID in the CPU Unit Is Incorrect or Not Registered:

Connect online to the CPU Unit from the Sysmac Studio and perform the following steps.

1 Overwrite or register the correct user program execution ID in the CPU Unit.

2 Cycle the power supply to the Controller, or reset the CPU Unit from the Sysmac Studio.

When the User Program Execution ID Is Not Assigned to the User Program or Is Incorrect

1 Read the backed up project file from the Sysmac Studio, and assign the correct user program execution ID.

2 Connect the Sysmac Studio to the CPU Unit online and transfer the user program.

3 Cycle the power supply to the Controller, or reset the Controller from the Sysmac Studio.

- **Other Situations**

To Delete the User Program Execution ID Assigned to the User Program:

Read the backed up project file in the Sysmac Studio.

To Delete the User Program Execution ID from the CPU Unit:

Connect the Sysmac Studio to the CPU Unit online and perform the Clear All Memory operation.

To Check the User Program Execution ID Assigned to the User Program:

For security, the user program execution ID that is assigned to the user program cannot be checked from the Sysmac Studio. Read the backed up project file in the Sysmac Studio and set the user program execution ID again.

To Check the User Program Execution ID in the CPU Unit:

For security, the user program execution ID that is set in the CPU Unit cannot be checked from the Sysmac Studio. Perform the Clear All Memory operation and register the correct user program execution ID.

Specifications

- **User Program Execution ID Verification Specifications**

Timing of Verification

At startup, the CPU Unit compares the user program execution ID that is registered in the CPU Unit with the user program execution ID that is assigned to the user program.

Verification Conditions

The conditions for verifications are given in the following table.

"A" and "B" indicate the IDs.

User program execution ID that is registered in the CPU Unit	User program execution ID that is assigned to the user program	Error	Operation
A	A	None	Possible
None	None		
None	A	ID Verification Error	Not possible.
A	None		
A	B		

Operation When the IDs Do Not Match

When the IDs do not match, an ID Verification Error (major fault level Controller error) occurs, and the CPU Unit does not operate. However, to reset the error you must cycle the power supply to the Controller or reset the Controller from the Sysmac Studio.

- **User Program Execution ID Character Specifications**

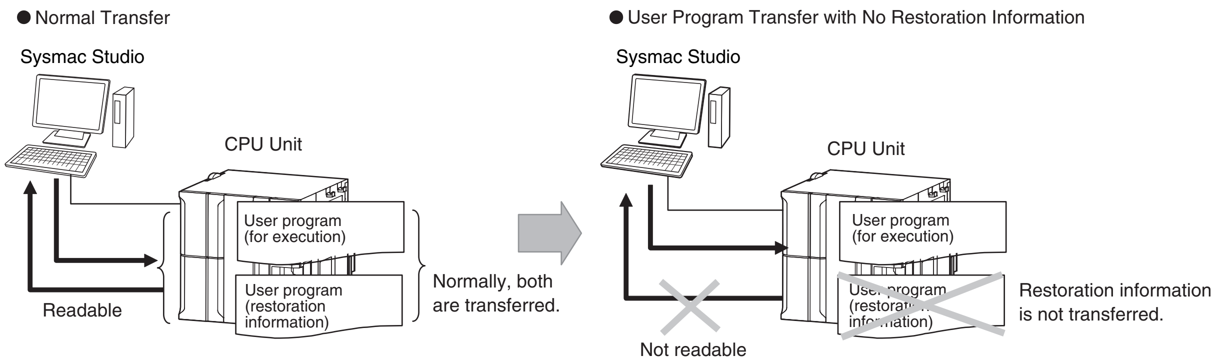
Usable characters	Case sensitivity	Maximum size (without NULL)
0 to 9, A to Z, and a to z	Case sensitive	8 to 32 characters

8-4-2 User Program Transfer with No Restoration Information

You can transfer the user program to the CPU Unit without the source code. This prevents anyone from displaying the user program on another computer even if they upload it.

Introduction

Normally, when you transfer the user program from the Sysmac Studio to the CPU Unit, information is transferred to restore it.
This function does not transfer information for restoration. That makes it impossible to read the user program.



This function is used to prevent theft of user program data when on-site maintenance of the user program is not required.

Operating Procedure

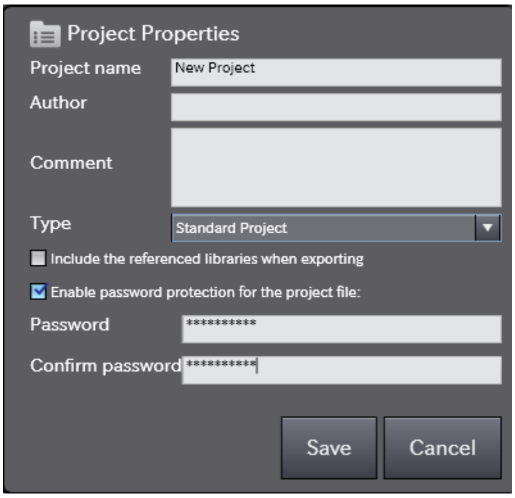
When you transfer the user program to the CPU Unit, select the *Do not transfer program source* Check Box in the Synchronization Window of the Sysmac Studio and then click the **Transfer to Controller** Button.

8-4-3 Overall Project File Protection

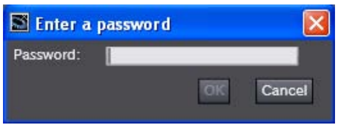
You can place a password on a project file to protect your assets.

Operating Procedure

This section describes how to set a password for a project. When you use ***Save As*** to save the project file, select the *Enable password protection for the project file* Check Box to enable setting a password.



Use the following procedure to open a project for which a password is set. If you try to open or import a project file for which a password is set, the Enter a Password Dialog Box is displayed.



Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific procedures.

8-4-4 Data Protection

You can place protection on part of the data in a project file to protect your assets.

Version Information

A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use data protection.

Introduction

You can place protection on part of the data in a project file to restrict access to that data. You can select any of three levels of access restrictions when you set protection. Protection must be temporarily cleared to access the restricted data. The length of time for which protection is cleared depends on the operation that you use.

Protected Data

Protection can be set for the following data. Only change protection can be set for cam profiles.

- Ladder diagrams (applies to programs, functions, and function blocks)
- ST (applies to programs, functions, and function blocks)
- Cam profiles

Levels of Access Restrictions

You can select one of the following levels of access restrictions.

- Prohibiting copying, displaying, and changing the data
- Prohibiting displaying and changing the data
- Prohibiting changing the data

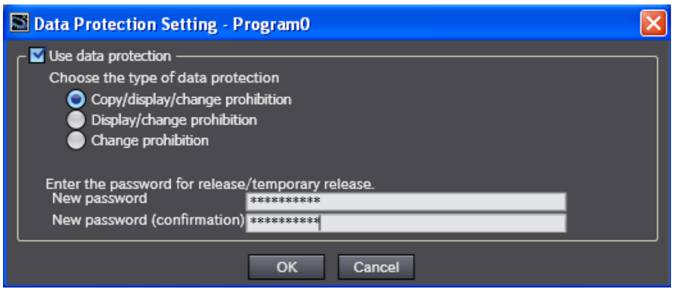
The following table shows the data access methods, restrictions for each restriction level, and the length of time that protection is cleared.

Data access operation	Levels of access restrictions			Length of time that protection is cleared
	Prohibiting copying, displaying and changing the data	Prohibiting displaying and changing the data	Prohibiting changing the data	
Displaying the data	Restricted.	Restricted.	Not restricted.	While the project is open
Printing the data	Restricted.	Restricted.	Not restricted.	
Changing the data	Restricted.	Restricted.	Restricted.	
Copying the data	Restricted.	Not restricted.	Not restricted.	Protection must be temporarily cleared for each operation
Displaying basic comparison results	Not restricted.	Not restricted.	Not restricted.	This operation is not restricted.
Displaying detailed comparison results	Restricted.	Restricted.	Not restricted.	While the project is open
Jumping from an event log or cross-reference to the data	Restricted.	Restricted.	Not restricted.	
Registering objects in a library	Restricted.	Not restricted.	Not restricted.	Not possible to clear protection temporarily.

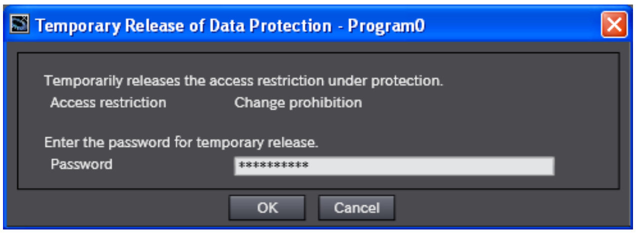
Operating Procedure

This function is used when the Sysmac Studio is offline. The settings are saved in the project file. When you use the synchronization function of the Sysmac Studio to transfer the project, the data protection settings in the data in the computer or Controller are transferred to Controller or computer.

Select ***Security*** – ***Set/Release Data Protection*** from the Controller Menu of the Sysmac Studio to set protection.



Select ***Security*** – ***Temporarily Change Prohibition of Data Protection*** from the Controller Menu of the Sysmac Studio to temporarily clear protection.



Select ***Security*** – ***Finish Temporary Change Prohibition of Data Protection*** from the Controller Menu of the Sysmac Studio to end temporary change protection.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific procedures.

8-4-5 Operation Authority Verification

Introduction

Online operations are restricted by operation rights to prevent damage to equipment or injuries that may be caused by operating mistakes. Examples are shown below.

- I/O Monitor: Writing, forced refreshing, etc.
- Controller operations: Changing the operating mode, online editing, MC Test Run, etc.

You can register passwords for operation authority for each CPU Unit in the Sysmac Studio. If a correct password is entered when an online connection is made to a Controller, the online operations for the operation authority category for the password that was entered will be allowed.

The Administrator sets a password for each operation authority. Users are notified of the operation authority name and password according to their skills.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific operating procedures for operation authorities.

Operation

For operation authority verification, select ***Security*** – ***Setting of Operation Authority*** from the Controller Menu on the Sysmac Studio.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific procedures.

Specifications

● Types of Operation Authorities

You can use the following five operation authorities on the Sysmac Studio. They are given in descending order of authority.

English name	Password
Administrator	Required.
Designer*1	Optional*2 Whether a password is required is determined by the default operation authority that is set in the Setting of Operation Authority Dialog Box. The default operation authority is used when a password is not input.
Maintainer*1	
Operator*1	
Observer*1	Not required.

*1 A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required.

*2 Whether a password is required is determined by the default operation authority that is set in the Setting of Operation Authority Dialog Box. A password must be entered to perform operations that require an operation authority that is higher than the default operation authority. A password is not required to perform operations that require an operation authority that is equal to or lower than the default operation authority.

● Examples of Online Operations for Operation Rights

Examples of the online operations that are allowed for each operation authority are given below. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details.

OK: Operation possible, VR: Verification required for each operation, NP: Operation not possible

Status monitor (example)	Adminis-trator	Designer	Main-tainer	Operator	Observer
Monitoring errors for troubleshooting	OK	OK	OK	OK	OK

I/O monitor operations (examples)	Adminis-trator	Designer	Main-tainer	Operator	Observer
I/O monitor: Reading	OK	OK	OK	OK	NP
I/O monitor: Writing	OK	OK	OK	VR	NP
Controlling BOOL variables	OK	OK	OK	VR	NP
Forced refreshing	OK	OK	OK	NP	NP

Controller operations (examples)	Adminis-trator	Designer	Main-tainer	Operator	Observer
RUN mode/PROGRAM mode	OK	OK	VR	NP	NP
Online editing	OK	OK	VR	NP	NP
Resetting the Controller	OK	OK	NP	NP	NP
Resetting errors (troubleshooting)	OK	OK	OK	VR	NP
Starting or restarting an MC Test Run	OK	OK	VR	NP	NP
User program execution IDs for Controllers	OK	NP	NP	NP	NP
CPU Unit write-protection	OK	OK	OK	NP	NP

● Password Specifications

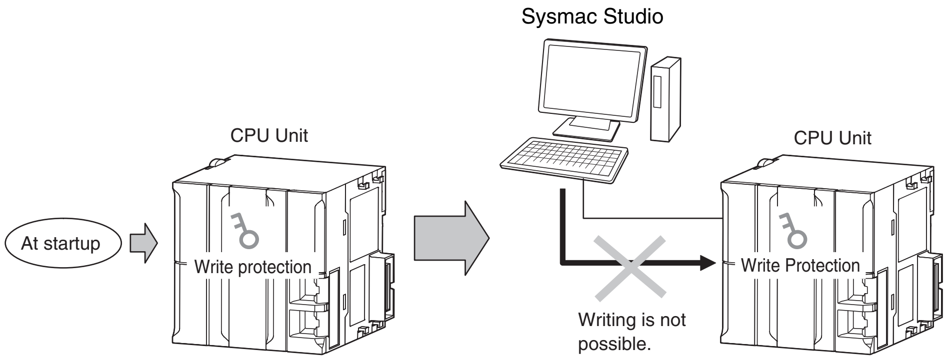
Item	Description
Valid number of characters	8 to 32
Applicable characters	Single-byte alphanumeric characters (case sensitive)

8-4-6 CPU Unit Write Protection

This function disables the ability to write data to CPU Units to protect user program assets and prevent misuse.

● Controller Write Protection at Startup

This setting automatically enables write protection when you turn ON the power supply to the Controller.

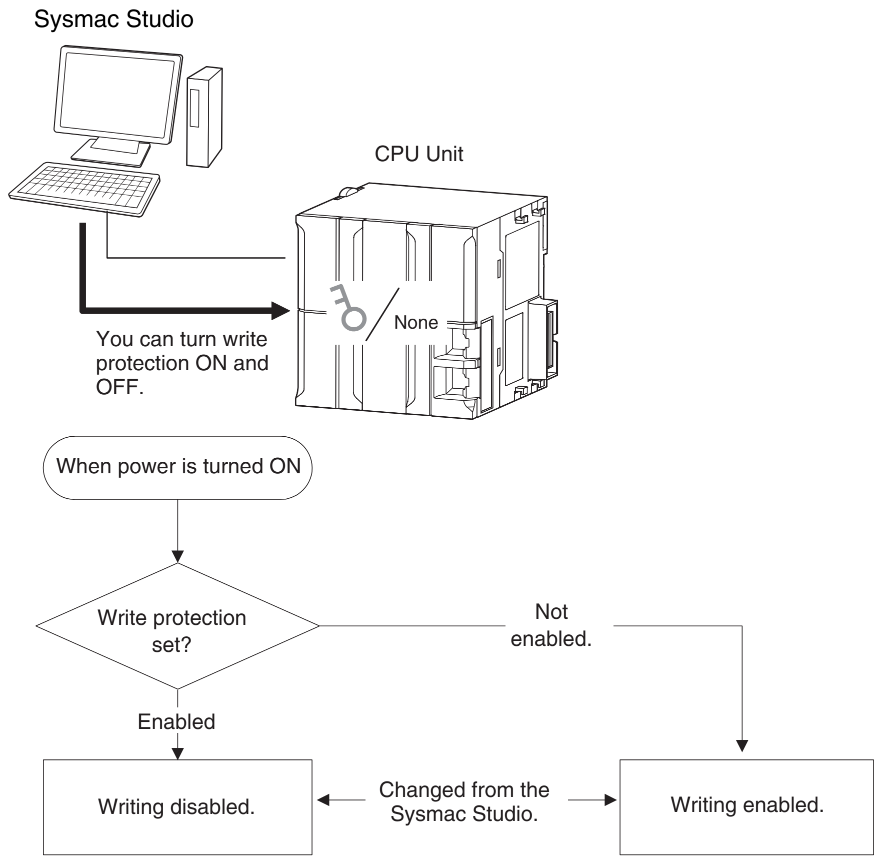


Set whether to automatically enable write protection when the power supply is turned ON in the **Operation Settings** under **Configurations and Setup** – **Controller Setup** of the Sysmac Studio.

Access point	Setting group	Setting	Description	Set values
Operation Settings, Operation Settings Tab, Basic Settings	Security Settings	Write Protection at Startup	Sets whether to enable write protection.	Do not use. Use.

● Setting and Removing Write Protection from the Sysmac Studio

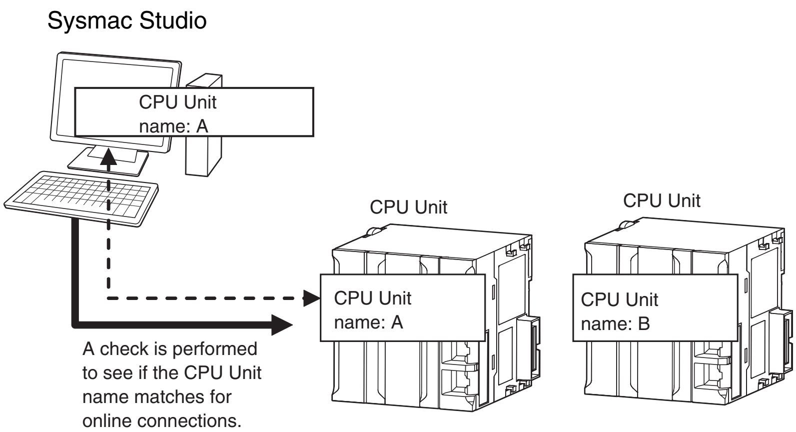
In the Sysmac Studio, go online and select ***Security*** – ***CPU Unit Write Protection*** from the Controller Menu to toggle write protection.



8-4-7 CPU Unit Names and Serial IDs

Introduction

Register a CPU Unit name in the CPU Unit. When going online to a CPU Unit from the Sysmac Studio, the CPU Unit name in the project is compared to the name of the CPU Unit being connected to. This helps prevent incorrect connections to the CPU Unit from the Sysmac Studio. It is particularly effective for operations performed over an EtherNet/IP network.

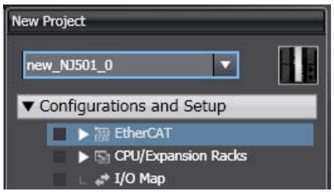


In addition to the CPU Unit name, it is also possible to use serial ID identification based on the CPU Unit production information (optional).

Setting Methods

1 Set the CPU Unit name when you create a project on the Sysmac Studio.

The CPU Unit name is displayed as shown below.



To change the name, right-click the Controller icon and select ***Rename***.

2 When you first connect to the CPU Unit online, the Sysmac Studio prompts you to store the CPU Unit name in the CPU Unit.

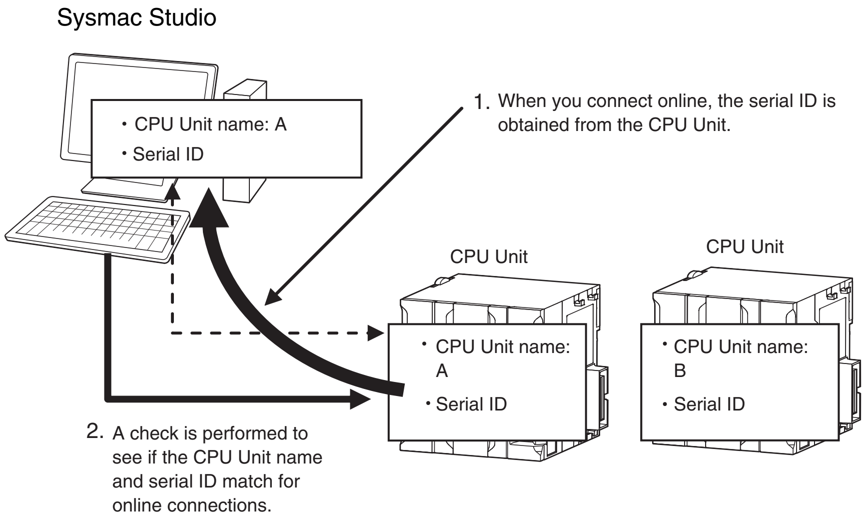
3 After that, when you connect to the CPU Unit online, the Sysmac Studio refers to the CPU Unit name in the project and the CPU Unit name of the CPU Unit you connect to. A warning dialog box is shown if they do not match, and you are asked whether to continue to connect.

Additional Information

You can name EtherNet/IP ports in the Network Configurator.

Serial IDs

When the Sysmac Studio first connects online, you can obtain the serial ID from the CPU Unit's production information and store it in the project. After that, when the Sysmac Studio connects online, both the CPU Unit name and the serial ID are compared. This enables stricter verification of the CPU Unit.



8-5 Debugging

This section describes debugging.

The NJ/NX-series Controller provides the following debugging operations.

- Forced refreshing
- Changing present values
- Online editing
- Data tracing
- Differential monitoring

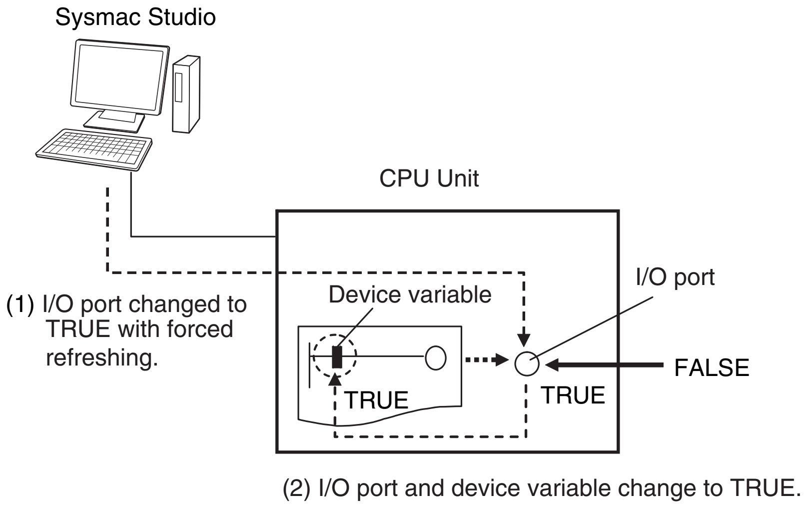
8-5-1 Forced Refreshing

Description

Forced refreshing allows the user to refresh external inputs and outputs with user-specified values from the Sysmac Studio to debug the system. Forced refreshing is executed not for the specified device variables, but for the I/O ports that are assigned to the device variables. The state that is specified with forced refreshing is retained until forced refreshing is cleared from the Sysmac Studio. (Refer to *Holding/Clearing Forced Refreshing* on page 8-36 for information how forced refreshing is retained or cleared according to changes in CPU Unit status.) All forced refreshing is cleared when a fatal error occurs, when a Clear All Memory operation is performed, when the operating mode is changed, when power is interrupted, or when the project is downloaded.

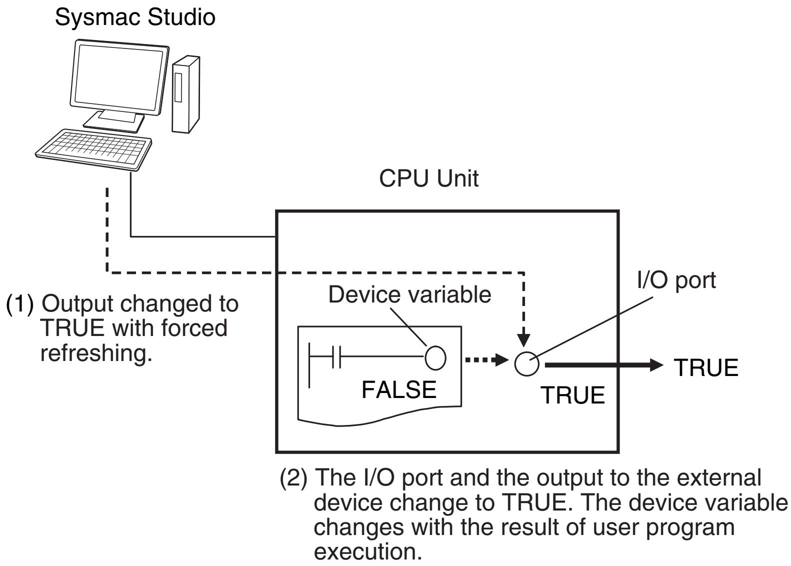
● Inputs

The I/O port and device variable change to the status that is specified with forced refreshing regardless of the status of the external input.



Outputs

The I/O port and the output to the external device change to the status that is specified with forced refreshing. In the user program, the status of the device variable that is assigned to the I/O port will not necessarily be the status that was specified with forced refreshing. It will change with the results of user program execution.



Applicable Areas

You can execute forced refreshing for the following I/O ports and memory used for CJ-series Units.

- I/O ports for EtherCAT slaves
- I/O ports for CJ-series Basic I/O Units
- I/O ports for CJ-series Special Units
- I/O bits for DeviceNet or CompoNet slaves that are specified for AT specifications from variables

If you execute forced refreshing from the Ladder Editor or the Watch Tab Page, the status of the I/O port or memory element for a CJ-series Unit will change via the variable.

Note You can use memory for CJ-series Units only with NJ-series CPU Units.

Number of Simultaneous I/O for Forced Refreshing

The number of variables that you can refresh with forced refreshing is listed below.

- CJ-series Units: 64 points total
- EtherCAT slaves: 64 points total

The number of external I/O points are given for the above limits. For example, if more than one variable is assigned the same external I/O point as the AT specifications, it is counted as only one point.

Application

Inputs

- To apply a simulated input signal to debug the user program
- To create a status that would occur only when a failure occurs (e.g., two exclusive bits turning ON or OFF at the same time)

Outputs

- To turn outputs ON and OFF to check wiring
- To intentionally turn OFF an output you do not want to operate regardless of results of user program execution

Operating Procedure

Operations can be performed from the following panes.

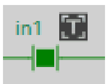
- Program Panes (Ladder diagram language)
- I/O Map
- Watch Tab Page

Procedure for Forced Refreshing from Ladder Editor

1. Select ***Monitor*** from the Controller Menu. The monitor turns ON.
2. Double-click the ladder program, ladder function, or ladder function block under **Programming** in the Multiview Explorer.

   The rungs are displayed on the Ladder Editor in monitor status.
3. Right-click the input or output and select ***Forced Refreshing*** – ***TRUE***. The input or output is forced to TRUE. Right-click the input or output and select ***Forced Refreshing*** – ***FALSE***. The input or output is forced to FALSE.
4. The input or output in the Ladder Editor changes to TRUE or FALSE and the execution condition changes accordingly.

   A mark that indicates that the input or output has forced status is displayed as shown below.



Ladder diagram

The TRUE or FALSE mark for forced status indicates the status that was specified for forced refreshing. It does not indicate the current value of the input or output.

Forced status mark	Operation
	TRUE specified with forced refreshing
	FALSE specified with forced refreshing

Additional Information

If there are other variables that are assigned the same memory address as one that is specified as the AT specification of a variable for which forced refreshing is specified, the forced status mark is displayed for all of the variables with that AT specification.

Affect of Operating Modes and Power Interruptions

Operating Modes for Forced Refreshing

You can execute forced refreshing in either PROGRAM mode or RUN mode. Forced refreshing is not possible while there is a major fault level Controller error.

● Status of Forced Refreshing during Operating Mode Changes or Power Interruptions

By default, the forced refreshing is cleared when the operating mode changes between RUN mode and PROGRAM mode and when the power is interrupted.

Holding/Clearing Forced Refreshing

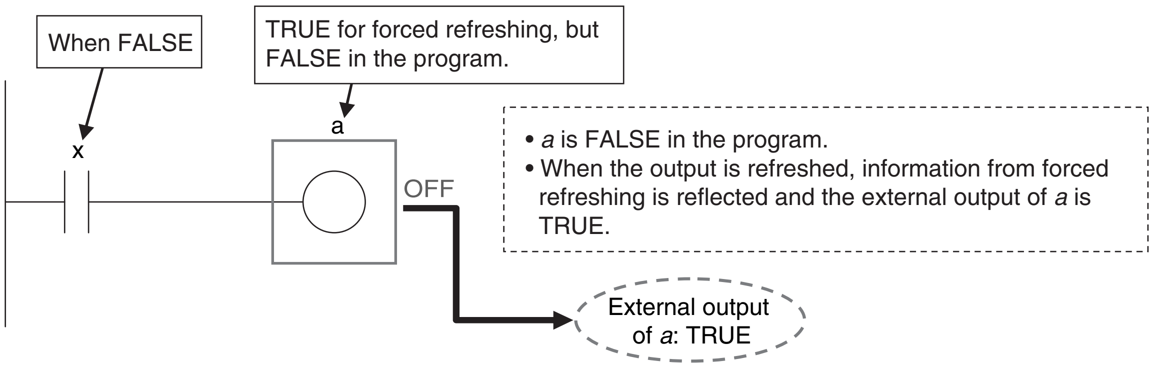
Forced refreshing is retained and cleared according to changes in the status of the CPU Unit as shown below.

Change in status		Forced refreshing status
When power is turned ON		Cleared
When operating mode changes	**RUN to PROGRAM mode** **PROGRAM to RUN mode**	Cleared
After downloading		Cleared
When a major fault level Controller error occurs		Cleared
During online editing		Retained

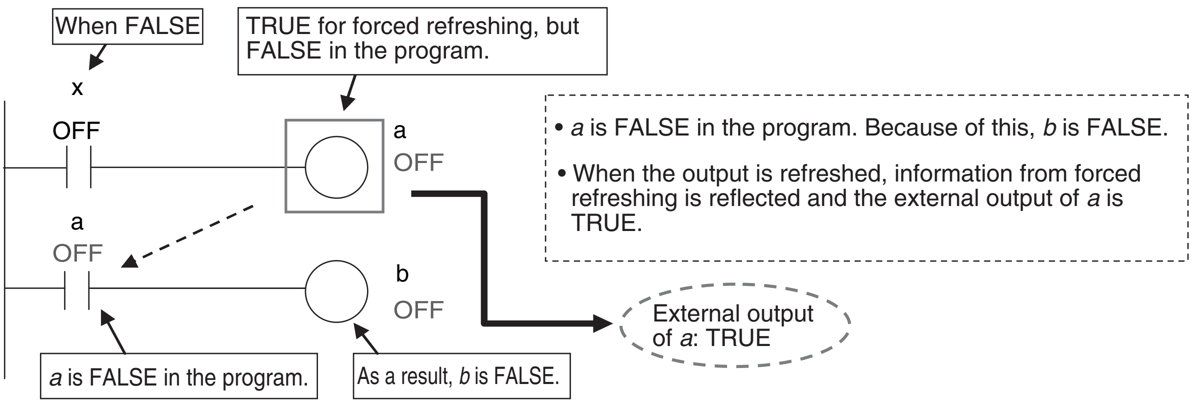
Programming Precautions for Forced Refreshing

If forced refreshing is set in the user program, the status of variables for which forced refreshing is specified are overwritten by the user program. Therefore, the status that is specified for forced refreshing is not maintained in the user program. However, refreshing to external devices uses the values that were specified for forced refreshing, and not the status of the variables in the user program. If forced refreshing is used in a program, the values of variables in the program may be different from the status of the external outputs.

Example: When *a* Is Refreshed to TRUE with Forced Refreshing



When There Is Another Input that is Controlled by the Forced Input



Precautions for Safe Use

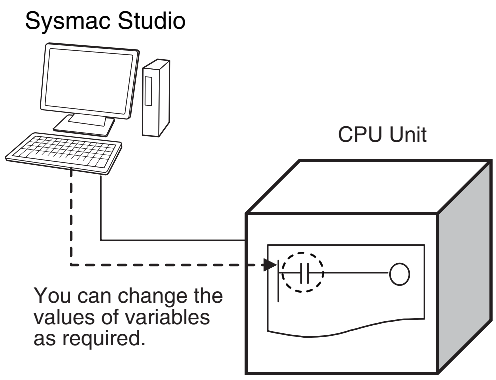
- Confirm that no adverse effect will occur in the system before you use forced refreshing.
- Forced refreshing ignores the results of user program execution and refreshes I/O with the specified values. If forced refreshing is used for inputs for which I/O refreshing is not supported, the inputs will first take the specified values, but they will then be overwritten by the user program.

Depending on the difference in the forced status, the control system may operate unexpectedly.

8-5-2 Changing Present Values

Description

You can change the present values of variables that are used in the user program and settings and you can change program inputs and outputs to TRUE or FALSE. This allows you to check the operation of the user program and settings.



Precautions for Safe Use

Always confirm the safety of the system before you change the present value of a variable.

Application

● Changing Program Inputs and Outputs to TRUE or FALSE

You can change the value of any BOOL variable to TRUE or FALSE. The specified value is then overwritten by the execution results of the user program. If the operating mode is changed or the power supply is cycled, the initial value is restored. You can control BOOL variables in the Ladder Editor, Watch Tab Page, or I/O Map.

● Changing the Values of Other Variables

You can change the present values of user-defined variables, system-defined variables, and device variables as required. You can do this on a Watch Tab Page.

Precautions for Safe Use

Always confirm the safety of the system before you change the present value of a variable.

Operating Procedure

Operations can be performed from the following panes to change the present values. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details on the operating procedures on the panes.

- Program panes (ladder diagrams and ST)
- I/O Map
- Watch Tab Page

Precautions on Changing the Status of Outputs Assigned to External Devices by Changing Present Values

Observe the following precautions when you change the status of an output that is assigned to an I/O port of a CJ-series Basic Output Unit or EtherCAT output slave by changing a present value.

● Changing Present Values in the I/O Map in RUN Mode

Any value of an I/O port that is changed in the I/O Map is then overwritten by the execution results of the user program. The value that was specified by changing the present value is not output to the external device. To change the value of an I/O port and output that value to an external device, use forced refreshing.

● Changing Present Values in a Watch Tab Page in PROGRAM Mode

The value that was specified in a Watch Tab Page by changing the present value of a device variable* that is defined as an external or local variable is not output to the external device. To output a specified value to an external device, do one of the following:

- Use forced refreshing.
- Change the present value in a Watch Tab Page of a device variable* that is defined as a global variable.

* The devices variables must be assigned to an I/O port of a CJ-series Basic Output Unit or EtherCAT output slave. This also applies to a global variable with an AT specification to an output bit that is assigned to a CJ-series Basic Output Unit.

● Precaution When Directly Writing to I/O Memory Addresses Assigned to Output Bits for CJ-series Basic Output Units

Any value that is written to an I/O memory address that corresponds to an output bit that is assigned to a CJ-series Basic Output Unit through a tag data link will be overwritten by the execution results of the user program. The value that is written directly to the I/O memory address from the tag data link will therefore not be output to the external device.

Note You can use memory for CJ-series Units only with NJ-series CPU Units.

8-5-3 Online Editing

This section introduces online editing. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details.

Introduction

The online editing function is used to add to or change part of a program in the CPU Unit directly from the Sysmac Studio.

You can select any of the following to perform online editing.

- POUs (programs, functions, and function blocks)
  For a ladder diagram program, select a section.
- Global variables

Application

You can use online editing to change a user program without stopping the operation of the CPU Unit.

Sysmac Studio Operations

● Performing Online Editing

1. Select the item to edit online.
2. Select ***Online Edit*** from the Project Menu.
3. Make the required changes.
4. Select ***Online Edit*** – ***Transfer*** from the Project Menu.
5. Check the results.
6. The user program will begin operation after online editing.

⚠ Caution

Execute online editing only after confirming that no adverse effects will occur if the I/O timing is disrupted. If you perform online editing, the task execution time may exceed the task period, I/O may not be refreshed with external devices, input signals may not be read, and output timing may be changed.

Precautions for Correct Use

- The differentiation status of differentiated instructions in a program that is edited online is initialized.
- When online editing changes are applied, the execution times of the tasks are extended. Set the task period appropriately so that you do not cause a Task Period Exceeded error due to online editing.
- If the power supply to the Controller is interrupted when online edits are being saved,* a major fault level Controller error (User Program/Controller Configurations and Setup Transfer Error, Incorrect User Program/Controller Configurations and Setup, or Non-volatile Memory Restored or Formatted) occurs. If one of these errors occurs, download the user program again.
- Do not execute the MC_SaveCamTable instruction while online edits are being saved.* Otherwise the online edits may not be saved correctly.

* Online edits are saved from when you click the **Yes** Button in the confirmation dialog box until the Online Editing Pane closes. However, with a CPU Unit with a unit version of 1.04 or later and Sysmac Studio version 1.05 or higher, saving continues until the dialog box that indicates saving data to built-in non-volatile memory (which is displayed after the confirmation dialog box) closes.

8-5-4 Data Tracing

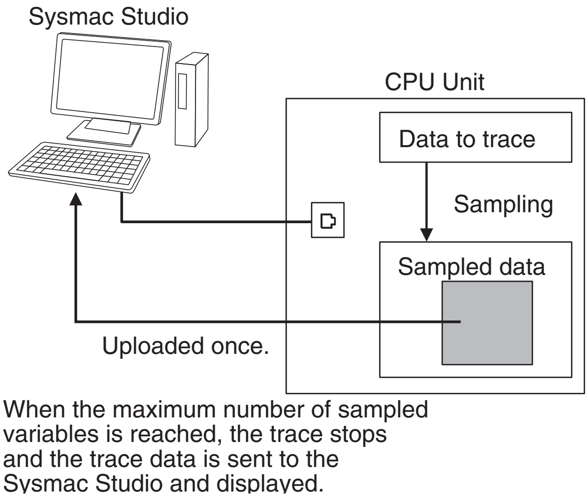
You can use data tracing to sample variables without any additional programming. You can read and check the data from the Sysmac Studio, and save the data to a file. This is used to start up, operate, and maintain devices.
This section introduces data tracing. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific operating procedures.

The two tracing methods are described below.

● Triggered Tracing

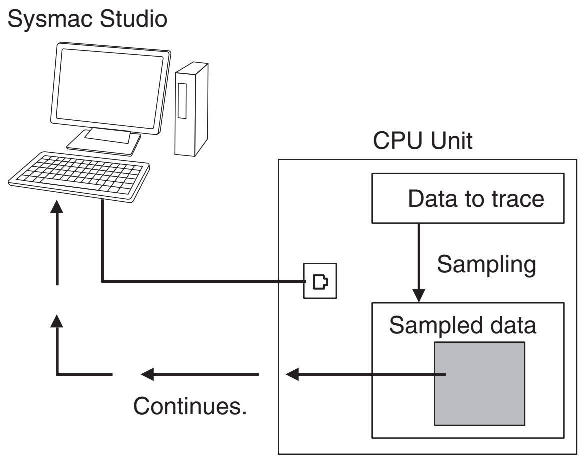
Trigger conditions are set to record data before and after an event. Sampling stops automatically when the maximum number of sampled variables is reached. Even if the Sysmac Studio is not online, you can trace data when trigger conditions are met and then upload the data after placing the Sysmac Studio online.

- You can check the flow of the program based on the status of changes in the present values of variables.
- You can use the data to investigate the cause of unexpected changes in the values of variables.



● Continuous Tracing

Sampling starts without any trigger and continues on even after 10,000 samples are collected. Sample data is transferred to the computer as it is collected and saved to a file. When the display buffer is full, the data is automatically saved to a CSV file. You can use this to store trace results data for a long tracing period in multiple CSV files.



Data Tracing Specifications

The following table gives the specifications of data tracing.

Item		Description
Types of data traces	Single triggered trace	Set a trigger condition to start sampling. Data from before and after the condition is met is saved.
	Continuous tracing	Sample data is transferred to a computer as it is collected and saved to a file.
Setting of timing of sampling	Period of specified task	Specify a task. The period of that task is set as the sampling period.
	Specified fixed interval	The time you enter is set as the sampling period. However, the time you enter is rounded off to an integer multiple of the primary periodic task.
	Trace sampling instruction	With this method, sampling is performed whenever the TraceSamp instruction is executed in the user program.
Setting sampled data[*1]	Maximum number of targets	NX701-□□□□: 192 variables NJ501-□□□□: 192 variables NJ301-□□□□: 48 variables NJ101-□□□□: 48 variables
	Data types	Basic data types except for text strings Arrays (specify the element) Enumerations Members of structures and unions
Maximum number of records		10,000 samples per variable
Setting trigger positions		The trigger position is set in respect to the overall trace time or quantity.
Setting triggers	Condition data types	Basic data types except for times, durations, dates, and text strings Arrays (specify the element), structures (specify the member), and unions (specify the member)
	Condition expression[*2]	Tracing is started when one of the following conditions is met. BOOL: TRUE or FALSE Non-BOOL: Equals (=), Greater than (>), Greater than or equal (≥), Less Than (<), Less than or equal (≤), Not equal (≠)[*3]
	Commands from Sysmac Studio	Tracing starts when the **Trigger TRUE** Button is clicked.
	Data Trace Trigger instruction	Tracing starts when the TraceTrig instruction is executed.
	Evaluation timing	• If something other than the TraceSamp instruction is used to set the timing of sampling, the trigger is evaluated only once in the specified task period. • If the TraceSamp instruction is used to set the timing of sampling, the trigger is evaluated whenever the instruction is executed.
	Delay	A slider is used to set the percentage of sampling before and after the trigger condition is met. (Example: 20%/80%)
Starting a trace	Commands from Sysmac Studio	Tracing is started when the **Execute** Button is clicked on the Sysmac Studio.
	Starting tracing at start of operation	Tracing can be started when operation of the Controller starts (i.e., when the operating mode is changed from PROGRAM mode to RUN mode).

Item		Description
Stopping a trace	**Triggered traces**	• Tracing stops when the maximum value of 10,000 samples is reached. • Tracing is stopped when the **Stop** Button is clicked on the Sysmac Studio.
	Continuous traces	• If stopping tracing is set as the operation to perform when the maximum number of samples is reached, tracing stops when the maximum number of samples or maximum amount of time is reached. • Tracing is stopped when the **Stop** Button is clicked on the Sysmac Studio.
Setting continuous tracing	**Maximum data storage period**	You can set the maximum amount of time to save continuous trace data.
	Maximum data storage size	You can set the maximum total size of all files saved during continuous tracing.
	Data items per file	You can set the number of samples to save in each file during a continuous trace.
	File save location	You can specify the location to create files to save data during a continuous trace.
	File name prefix	You can specify a prefix to automatically add to the beginning of the file names.
	Setting of operation when limit is reached	You can specify the operation to perform when the storage time period or size limit is reached. • Stopping the trace • Deleting the oldest files and continuing
Displaying trace results	**Graph display**	You can display a graph where the X axis represents time and the Y axis represents the value of the variable. You can display both BOOL variables and other variables on the same graph.
	Table display	You can display the maximum value, minimum value, average value, and value at the specified time for each variable in a table.
	3D Motion Monitor Display Mode	You can position a virtual composition model in 3D space and display the composition motion based on the command positions and actual positions of the motion axes.
Exporting trace data	**Exporting to CSV files**	You can save the trace results and all settings other than the trace number to a CSV file.
Number of data traces that can be executed simultaneously[*4]		NX701-□□□□: 4 traces NJ501-□□□□: 4 traces NJ301-□□□□: 2 traces NJ101-□□□□: 2 traces
Importing trace data		You can display CSV format trace results on top of the current graph.
Saving		You can save the trace results in the project along with the trace settings.
Printing		You can print graphs. The Sysmac Studio's printing functionality is used.

*1 You cannot perform data traces for the *EN*, *ENO*, *P_off*, *P_on*, *P_CY*, *P_First_RunMode*, *P_First_Run* and *P_PRGER* system-defined variables, in-out variables that are used in function block instances, and variables in functions.

*2 Data tracing will not start at the data trace starting point even if the trigger condition is met.

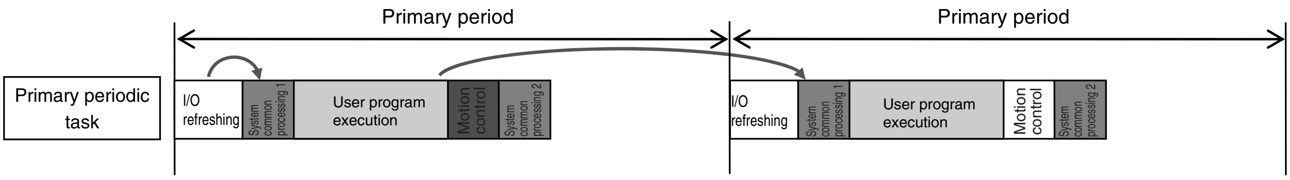
*3 Combinations of multiple condition expressions are not permitted. Also, the valid range for comparison constants is determined by the valid range of the literal expressions for the variable type on the left side of the condition expression.

*4 Trace numbers 0 to 3 are set for the NX701 and NJ501. Trace numbers 0 and 1 are set for the NJ301 and NJ101. These numbers are used to execute instructions and to access system-defined variables.

Data Trace Operation

Processing for data traces (sampling and trigger detection) are performed in System Common Processing 1, between I/O refreshing and user program execution.

Example: If sampling is specified in the primary periodic task, data tracing is executed in System Common Processing 1, as shown in the following diagram.



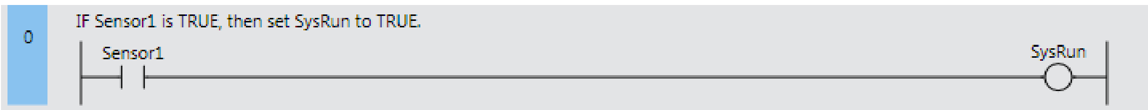
Display examples for data trace operations and execution results is given below for sampling in a specified task period.

Additional Information

I/O refreshing, user program execution, and motion control processing are all executed in the same task period. For data tracing, user program execution and motion control processing for the current task period and I/O refreshing for the next task period are displayed at the same time. The timing charts in the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) are based on the task periods, so the display are not the same as those for data tracing.

Example 1:

In this example, the *SysRun* variable is changed to TRUE in the user program when the *Sensor1* variable (assigned to the sensor input signal) changes to TRUE.

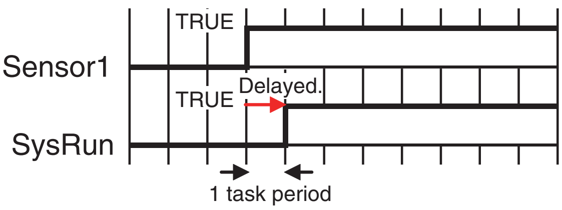


The data trace operations and display of the execution results are given below.

1. In data trace processing in System Common Processing 1, TRUE is obtained for *Sensor1*.
2. *SysRun* is changed to TRUE in the user program.
3. In data trace processing in System Common Processing 1 in the next primary period, TRUE is obtained for *SysRun*.

Therefore, in the data trace display, *SysRun* is shown as TRUE one task period after *Sensor1*.

Data Trace Display

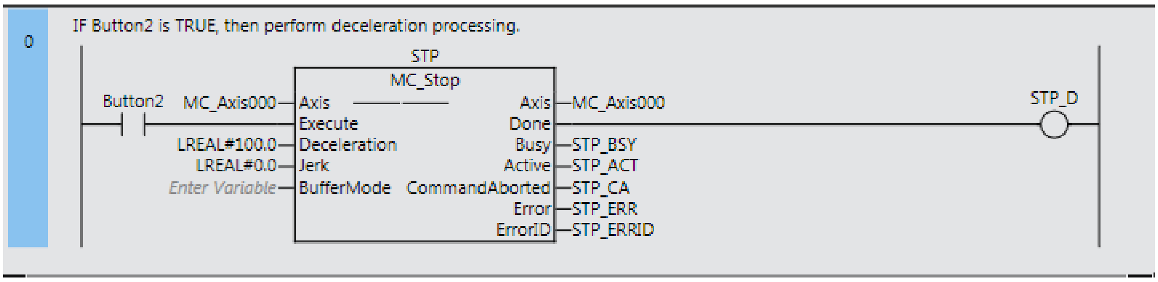


Additional Information

If the values of variables change during user program execution, the changes in the values and changes for output processing for I/O refreshing are changed in the same task period.

Example 2:

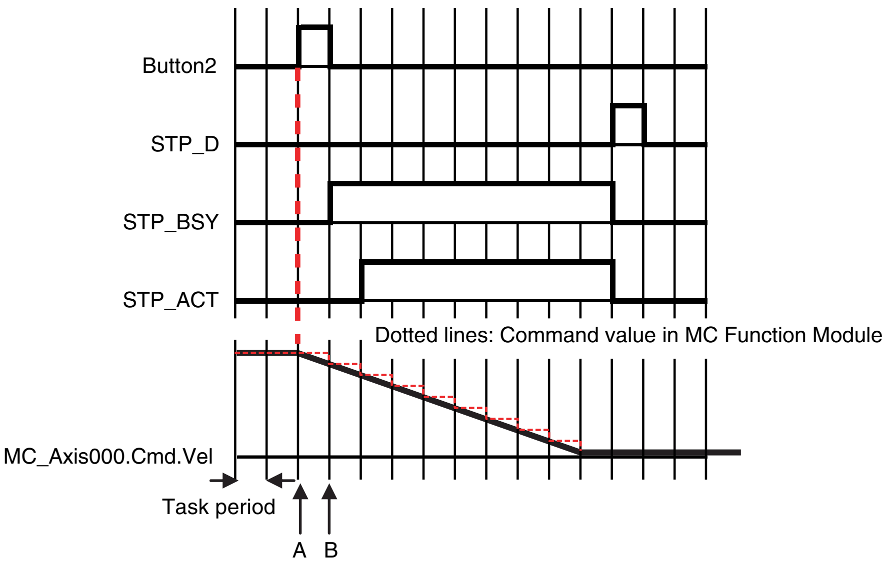
When the *Button2* variable (assigned to an input signal from a pushbutton) changes to TRUE during velocity control, the user program in this example decelerates axis 0 (*MC_Axis000*) to a stop.



The data trace operations and display of the execution results are given below.

1. In data trace processing in System Common Processing 1, TRUE is obtained for *Button2*.
2. *STP_BSY* is changed to TRUE in the user program and the Motion Control Function Module performs deceleration processing.
3. In data trace processing in System Common Processing 1 in the next primary period, TRUE is obtained for *STP_BSY* and the status of the motion variable is obtained.
4. *STP_ACT* is changed to TRUE in the user program.
5. In data trace processing in System Common Processing 1 in the next primary period, TRUE is obtained for *STP_ACT*.

The command value in the MC Function Module starts changing (B in the following diagram) when *STP_BSY* changes to TRUE in the user program and the Motion Control Function Module starts to perform deceleration processing. The command value changes stepwise in synchronization with the primary periodic task. The data trace, however, interpolates the values to connect the values for the previous and current periods. Therefore, the display shows that the command value for the Command Velocity motion control variable (*MC_Axis000.Cmd.Vel*) changes one period early, i.e., when *Button2* changes to TRUE (A in the following figure). The display also shows that *STP_BSY* changes to TRUE one period after deceleration starts and then *STP_ACT* changes to TRUE after another period.



Additional Information

For function blocks that contain motion control instructions, the values of input parameters are passed to the input variables when execution of the function block starts, and the values of the output variables are passed to the output parameters when execution of the function block ends. (Refer to *Variable Designations for Function Blocks* on page 6-11.) On the data trace displays, input parameters and input variable, and output parameters and output variables, change in the same task period.

Related System-defined Variables

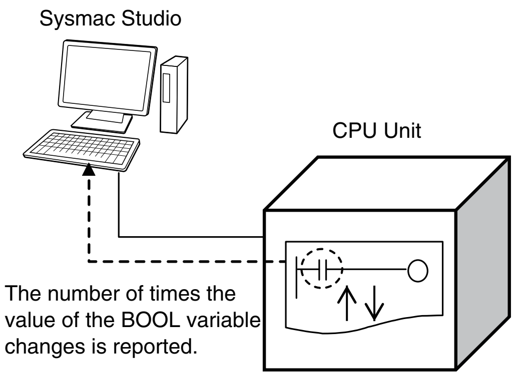
Variable name	Member	Meaning	Description	Data type	R/W
_PLC_TraceSta[0..3]*				_sTRACE_STA	R
	.IsStart	Trace Busy Flag	TRUE when a trace starts.	BOOL	R
	.IsComplete	Trace Completed Flag	TRUE when a trace is completed. Changes to FALSE when the next trace starts.	BOOL	R
	.IsTrigger	Trace Trigger Monitor Flag	TRUE when the trigger condition is met. Changes to FALSE when the next trace starts.	BOOL	R
	.ParamErr	Trace Parameter Error Flag	Changes to TRUE when a trace starts if there is an error in the trace settings. FALSE when the settings are normal.	BOOL	R

* These numbers correspond to the data trace numbers 0 to 3.

Note You cannot use these system-defined variables in the user program. Use the GetTraceStatus instruction to read the status of data tracing from the user program.

8-5-5 Differential Monitoring

Differential monitoring reports the number of times the value of the specified BOOL variable matches the specified condition. The specified condition is evaluated for a match in every task period of the primary periodic task (called the primary period). Differential monitoring provides a running total of the number of times the condition is matched.



Version Information

A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to use differential monitoring.

Application

You can use differential monitoring to check or count the number of times an external input signal turns ON or OFF or a input in the user program changes to TRUE or FALSE. This is useful during system commissioning and for troubleshooting operation failures during production.

Specifications of Differential Monitoring

The specifications of differential monitoring are given in the following table.

Item		Specification
Differential monitoring condition	Number of variables	8 max.
	Specified variables	• BOOL • Element of BOOL array • BOOL member of structure or union
	Condition expression	• Change to TRUE • Change to FALSE
Conditional match evaluation	Timing	Once every primary period
	Match count	The number of times the specified variable matches the condition is counted.
Starting and stopping	Start condition	Command from Sysmac Studio
	Stop condition	• Command from Sysmac Studio • Occurrence of a major fault level Controller error • User program download • Clear All Memory operation • Disconnecting online connection to Sysmac Studio
Operating modes		• RUN mode • PROGRAM mode

Differential Monitoring Conditions

The variables and the changes that you can monitor with differential monitoring are called differential monitoring conditions. The specifications for the differential monitoring conditions are described below.

● Number of Variables

You can specify a maximum of eight variables. This means differential monitoring can detect the numbers of times conditions are met for eight variables in parallel.

● Specified Variables

The data types of the variables that you can specify for differential monitoring are given below.

- BOOL
- Elements of BOOL arrays
- BOOL members of structures or unions

You cannot specify an array, structure, or union.

The types of variables that you can specify are listed below.

Type of variables		Specification
System-defined variables		Possible.*1
Semi-user-defined variables		Possible.
User-defined variables	Global variables	Possible.
	Variables used in a program	Possible.
	Variables used in a function block	Possible.*2
	Variables used in a function	Not possible.

*1 The following variables cannot be used:
EN, ENO, P_Off, P_CY, P_First_RunMode, P_First_Run, and P_PRGER.

*2 In-out variables cannot be used.

Condition Expressions

The condition of the change in the variable to detect is called the condition expression. There are two types of condition expressions that you can select from. You specify a condition expression for each variable.

- Change to TRUE
- Change to FALSE

Precautions for Correct Use

For example, we will assume the condition expression was set to a change to TRUE. Even if the value of the specified variable is TRUE when differential monitoring is started, the status of the value of the variable is not detected as a change to TRUE. The value of the variable must first change to FALSE and then to TRUE to be considered as a change to TRUE.

When Conditions Are Evaluated and the Match Count

The condition for the specified variable is evaluated every primary period. The value of the variable from the previous evaluation is compared with the value of the variable for the current evaluation. If the value of the variable that matches the specified condition has changed, the count is incremented.

- The number of times a condition match occurs is counted separately for each variable.
- The count values are reset to zero when differential monitoring is started.
- The count value for just one variable cannot be reset to zero.

Precautions for Correct Use

- Even if the value changes to match the condition expression more than one time within the same primary period, the count will be incremented only once in each primary period.
- If the values of the variable are the same at the time of the previous and current evaluations, the condition is not considered to be a match, even if the value changed between evaluations.

Start Condition and Stop Condition

Use the Sysmac Studio to start differential monitoring.

Normally, use the Sysmac Studio to stop differential monitoring. Differential monitoring will stop automatically at the following times.

- When a major fault level Controller error occurs
- When the user program is downloaded
- When the Clear All Memory operation is performed
- When an online connection to the Sysmac Studio is broken

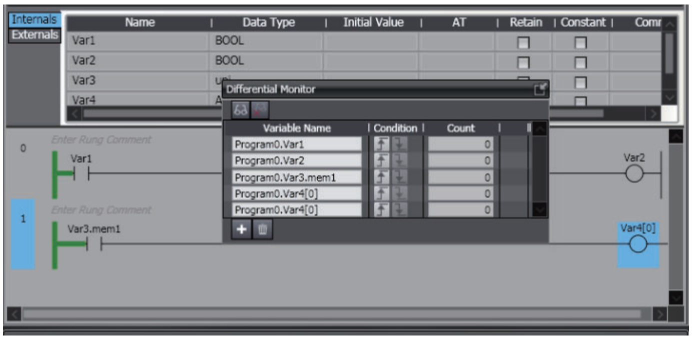
Procedure

Use the following procedures to control differential monitoring. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details.

1 Select ***Differential Monitor*** from the View Menu on the Sysmac Studio.

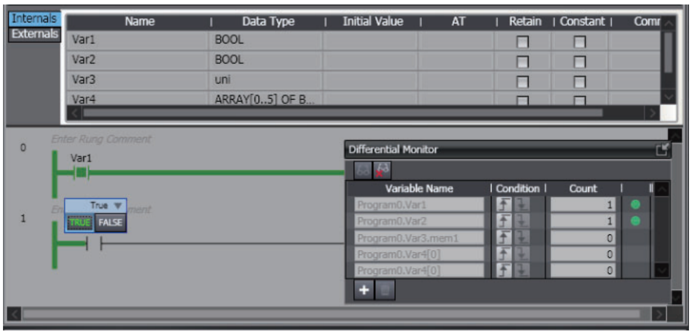
2 Right-click a variable that can be specified for differential monitoring and select ***Add Differential Monitor.***

3 Set the differential monitoring condition expression for each variable in the Differential Monitor Window.



4 Execute the user program.

The number of times that the condition is met for each variable is displayed in the Differential Monitor Window.



Precautions for Differential Monitoring

Observe the following precautions when you use differential monitoring.

● Loss of Communications with Sysmac Studio While Differential Monitoring Is in Progress

Let's assume that communications with the Sysmac Studio were cut off during differential monitoring because the communications cable was disconnected or because the Sysmac Studio ended due to an error. In such cases, the CPU Unit will continue execution of differential monitoring. To restart execution of differential monitoring, you must resume communications with the Sysmac Studio and stop differential monitoring.

● Simultaneous Execution of Differential Monitoring

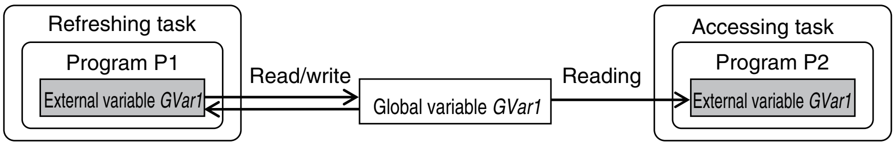
You cannot run differential monitoring from more than one copy of the Sysmac Studio running on the same computer or from the Sysmac Studio running on different computers.

● Specifying Global Variables and External Variables

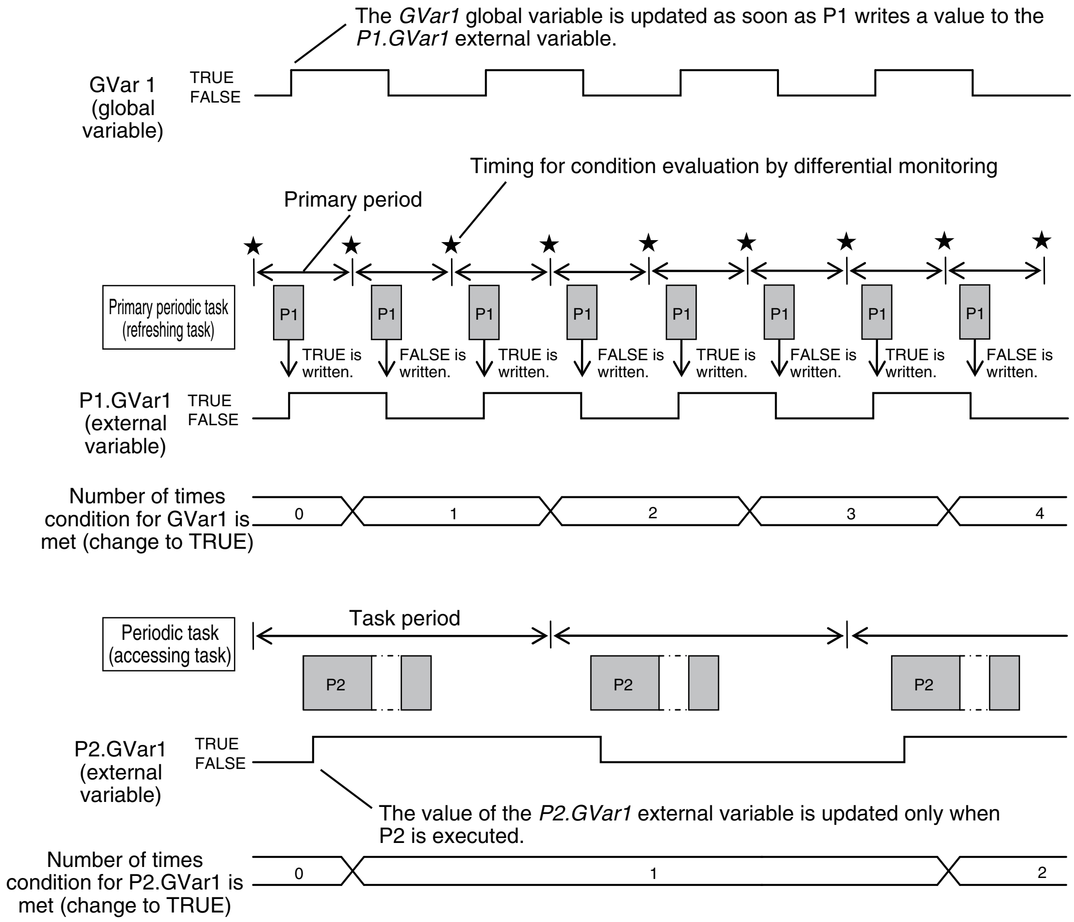
You can specify global variables or external variables (which specify global variables in POUs) for differential monitoring. Keep in mind that the values of global variables and external variables are updated at different times. A global variable is updated as soon as the value is written. An external variable, however, is updated only when the CPU Unit executes the POU in which that external variable is declared. The following figure shows this. In this example, the following two variables are monitored.

Variable	Type of variable	POU that executes the read/write
GVar1	Global variable	The P1 program that is assigned to the primary periodic task
P2.GVar1	This is an external variable that is declared in the P2 program and points to *GVar1*.	The P2 program that is assigned to the periodic task

The *GVar1* global variable is read and written by the P1 program that is assigned to the primary periodic task. Therefore, it will be updated in the primary period as long as the program writes to it every period. The *P2.GVar1* external variable, however, is updated only when the CPU Unit executes the P2 program that is assigned to the periodic task. This means the external variable is updated only in the task period of the periodic task. Because the task period of the periodic task is longer than the primary period, the count for *P2.GVar1* is updated fewer times than the count for *GVar1*.



Refreshing task
Program P1
External variable GVar1
Read/write
Global variable GVar1
Reading
Accessing task
Program P2
External variable GVar1



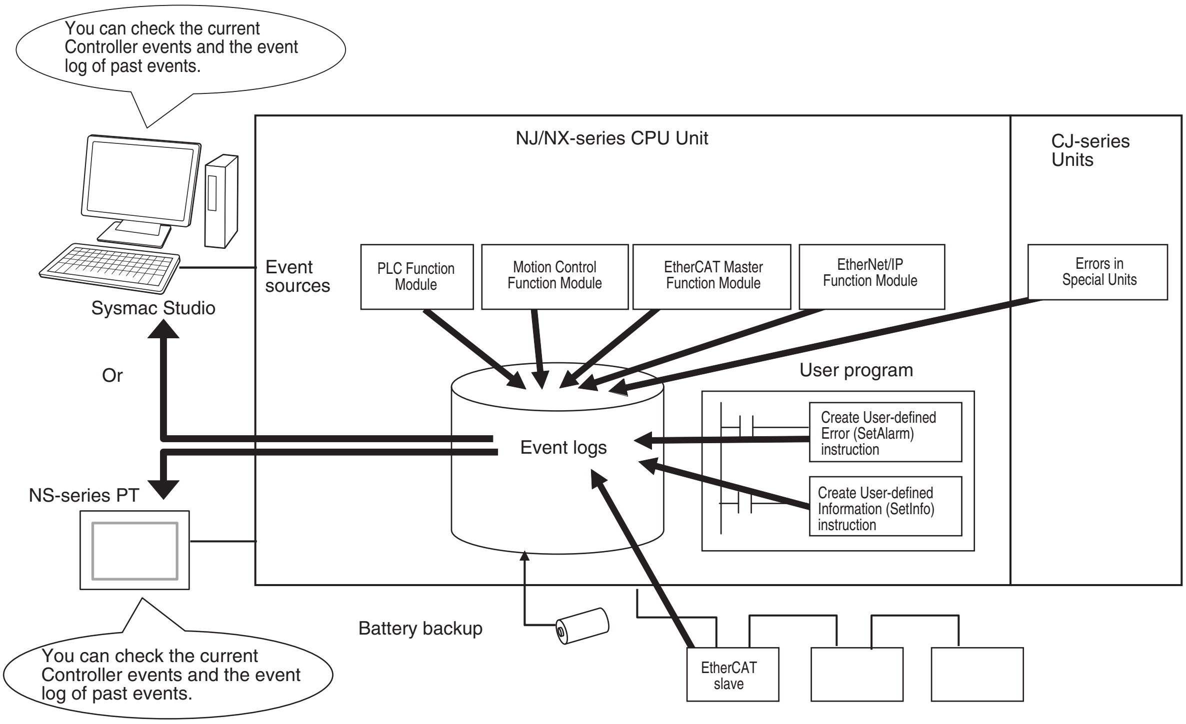
The GVar1 global variable is updated as soon as P1 writes a value to the P1.GVar1 external variable.
GVar 1 (global variable)
TRUE
FALSE
Timing for condition evaluation by differential monitoring
Primary period
Primary periodic task (refreshing task)
P1
TRUE is written.
FALSE is written.
P1.GVar1 (external variable)
Number of times condition for GVar1 is met (change to TRUE)
0
1
2
3
4
Periodic task (accessing task)
Task period
P2
P2.GVar1 (external variable)
The value of the P2.GVar1 external variable is updated only when P2 is executed.
Number of times condition for P2.GVar1 is met (change to TRUE)
0
1
2

8-6 Event Logs

This section describes the event logs.

8-6-1 Introduction

The event logs contain records of events,* such as errors, status changes, and user-defined events, that occurred in the NJ/NX-series Controller.



* Here, events are unscheduled events that occur on the Controller, such as errors. “Event” refers to an error or to information that does not indicate an error but for which the user must be notified by the Controller or for a user definition. There are two types and four classifications of events.

- Controller events
  Controller errors
  Controller information
- User-defined events
  User-defined errors
  User-defined Information

To use an NS-series PT to check events, connect the PT to the built-in EtherNet/IP port on the CPU Unit.

Precautions for Correct Use

- You can use CJ-series Units only with NJ-series CPU Units.
- Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for details on the PT's Troubleshooter.

Features

Event logs have the following features.

- In addition to error records, various records are recorded for events such as the time the power supply is turned ON or OFF, and the time when operation is started.
- You can check these records based on the time. You can therefore use them to isolate the causes of errors when problems occur.

Types of Events

Events are classified as shown below.

System-defined Events (Controller Events)

The Controller automatically detects these events. Controller events include events for the function modules in the CPU Unit, CJ-series Units, and EtherCAT slaves. The different types of system-defined events are as follows:

- Controller errors
- Controller information

User-defined Events

These are events that occur in applications that the user developed. You can execute instructions to create the following types of events.

- User-defined errors
- User-defined information

You can read the event logs from the Sysmac Studio or from an HMI.

8-6-2 Detailed Information on Event Logs

Event Sources

This information identifies where an event occurred in the Controller. The event sources are given below for Controller events and user-defined events.

Sources of Controller Events

Controller events occur in the function modules in the CPU Unit.

For some function modules, there is more detailed information about the event source. This information is called the detailed event source.

The following are Controller events.

Event source	Source details
PLC Function Module	I/O bus master or CJ-series Unit *
Motion Control Function Module	Common, axis, or axes group
EtherCAT Master Function Module	Communications port, EtherCAT master, or EtherCAT slave
EtherNet/IP Function Module	Communications port/communications port 1/communications port 2, CIP/CIP1/CIP2, FTP, NTP, or SNMP

* The source details information does not show information from the error histories from within CJ-series CPU Special Units or EtherCAT slaves. Read the error histories from the appropriate Support Software.

Sources of User-defined Events

User-defined events occur in the PLC Function Module.

Category

This information displays the category of event log. It is used to access error logs from the Sysmac Studio or an HMI.

Event type	Event log category	Description
Controller events	System log	The Controller automatically detects and records these events. CJ-series Unit errors are also included.
	Access log	This is a record of events that have affect Controller operation due to user actions.
User-defined events	User event log	This is a log of events that are defined by the user.

Number of Records

Each event log can contain the following number of records. If the number of events exceeds the number of records permitted, the CPU Unit overwrites the oldest events.

Event type	Event log category	Maximum number of records
Controller events	System log	NX701-□□□□: 1,024 events, NJ501-□□□□: 1,024 events, NJ301-□□□□: 512 events, NJ101-□□□□: 512 events
	Access log	
User-defined events	User event log	

Retaining Events during Power Interruptions

The NJ/NX-series CPU Unit uses a Battery to retain the event logs when the power is interrupted.

Precautions for Correct Use

The event logs are retained by Battery. They are not retained when there is no Battery.

Periodically export event logs as required.

Event Codes

Event codes are assigned to Controller events by the system in advance according to the type of event. Event codes are assigned to user-defined events by the user. Controller event codes are 8-digit hexadecimal values. You can use the Get Error Status instruction to read the error codes of current errors. You can assign a decimal number from 1 to 60,000 as the event code for a user-defined event.

Event Levels

Each event has an event level that indicates its level. The event level depends on the type of event. Levels are defined separately for Controller events and user-defined events.

● Controller Events

Controller events are classified into five levels according to the degree of the effect that the events have on control, as shown in the following table.

No.	Level		Classification
1	High	Controller errors	Major fault level
2	▲		Partial fault level
3			Minor fault level
4	▼		Observation level
5	Low	Controller information	Information level

Errors with a higher level have a greater impact on the functions that the Controller provides, and are more difficult to recover from.

When an event in one of these levels occurs, the Sysmac Studio or an HMI will display the error.

● User-defined Events

User-defined events are classified into the following levels. These levels are defined by the NJ/NX-series System.

The event levels are defined for user-defined events.

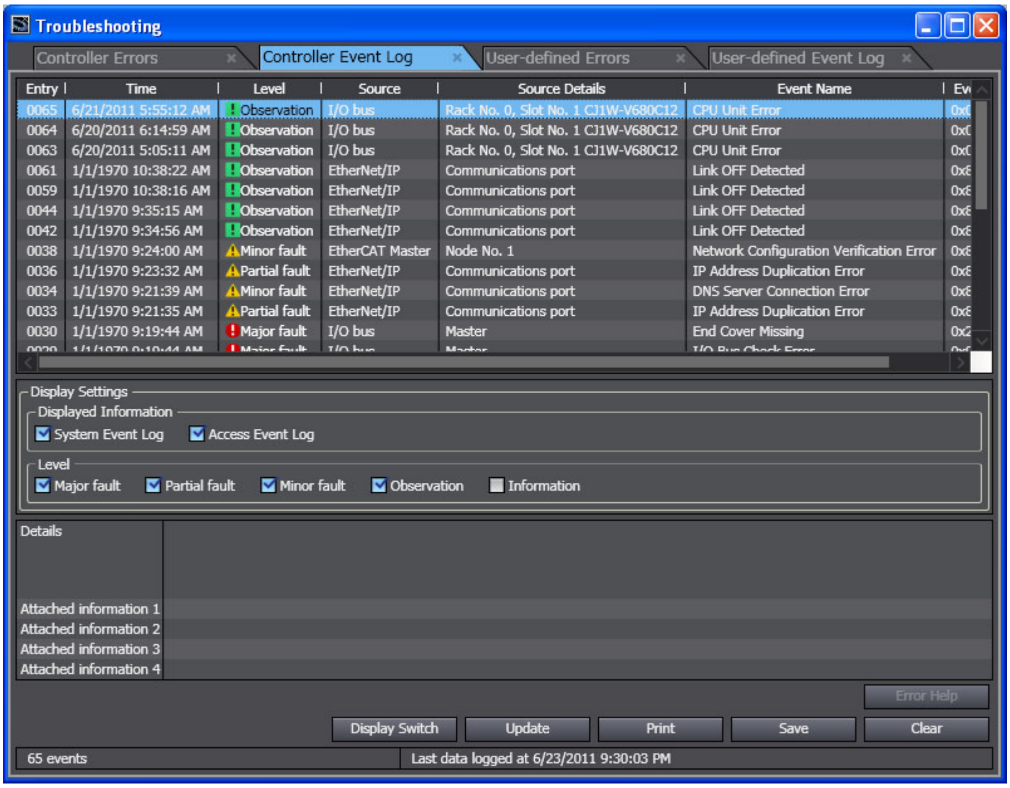
No.	Level	Type	Meaning
1	High	User fault Level 1	These event levels indicate a user-defined error in an application. The user executes the SetAlarm (Create User-defined Error) instruction to create the event.
2	▲	User fault Level 2	
3		User fault Level 3	
4		User fault Level 4	
5		User fault Level 5	
6		User fault Level 6	
7		User fault Level 7	
8	▼	User fault Level 8	
9	Low	User Information	These event levels indicate user-defined information in an application. The user executes the SetInfo (Create User-defined Information) instruction to create the event.

Displaying Event Logs

The Sysmac Studio or an HMI displays two event logs: the Controller event log and the user-defined event log. The Controller logs include both the access log and the system log.

The Sysmac Studio can also display the error logs that are recorded in the CJ-series Units and EtherCAT slaves.

The events in these logs are displayed in tables on the Sysmac Studio. Select an event from the table to display detailed information.



Additional Information

If an event occurs in the Controller that is not supported by the version of the Sysmac Studio or an HMI, the source is displayed as “Unknown” and the event name is displayed as “Unknown Event.” The event code and attached information are displayed correctly.

Clearing Event Logs

● Clearing Event Logs from the Sysmac Studio or an HMI

You can clear the event logs from the Sysmac Studio or from an HMI. You can clear the Controller event log and user-defined event log separately.

Precautions for Correct Use

- If you need to delete event log in the CPU Unit from the Sysmac Studio or an HMI, make sure you do not need any of the event information before you delete the event log. You may have overlooked some important information and observation level Controller events or user-defined events. Always check for these before you delete an event log.
- Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for restrictions on clearing an event log from the PT.

● Clearing Event Logs with the Clear All Memory Operation

When you perform the Clear All Memory operation for an NJ/NX-series CPU Unit from the Sysmac Studio, you can select whether to clear the event logs.

Exporting Event Logs

You can use the Sysmac Studio or an HMI to export the displayed event log to a CSV file.

8-6-3 Controller Events (Controller Errors and Information)

Introduction

Controller errors and information are defined by the NJ/NX-series System.

These events occur when the NJ/NX-series System detects an error or information factor.

● Controller Errors

These are system-defined errors.

"Controller error" is a collective term for major fault level, partial fault level, minor fault level, and observation level Controller events.

Errors in the function modules of the CPU Unit, CJ-series Units, and EtherCAT slaves are detected. When one of these events occurs, a Controller error is recorded in the event log.

To check the status of a Controller error on the user program, you execute the Get Error Status instruction to access the status of the Error Status variable, which is a system-defined variable.

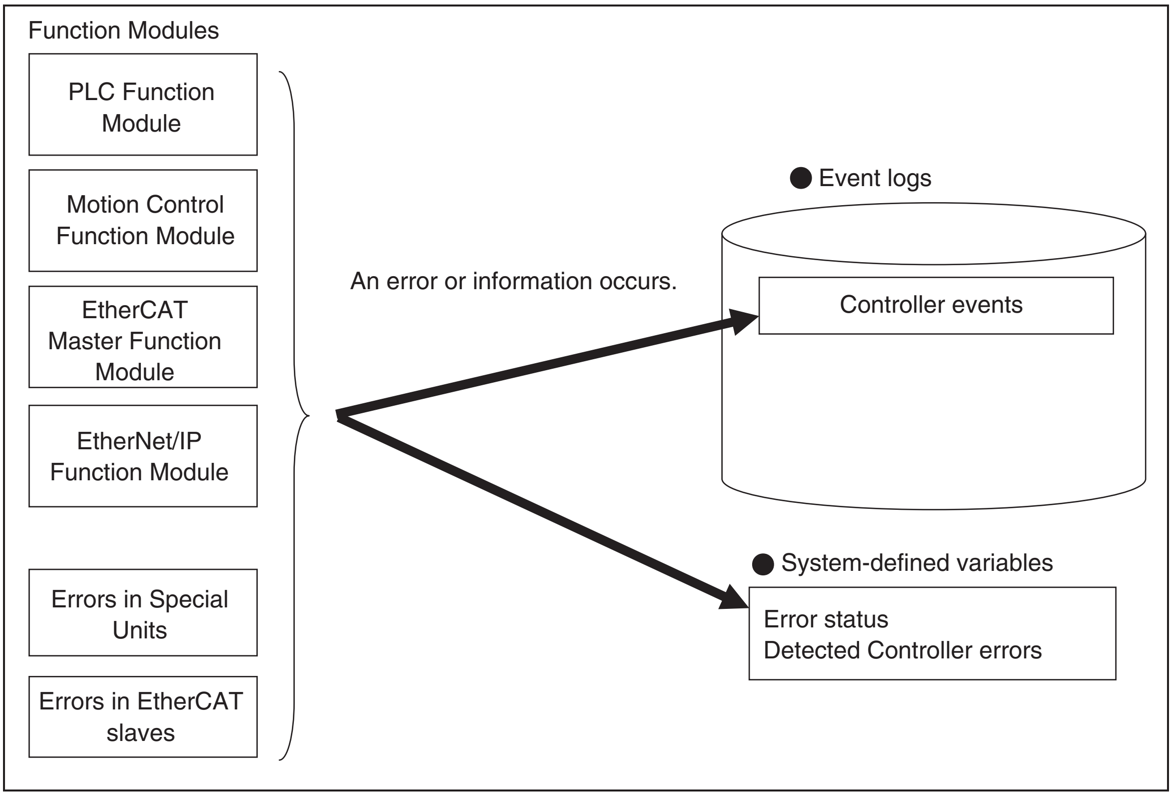
Note You can use CJ-series Units only with NJ-series CPU Units.

Controller errors are not reset when the operating mode changes.

Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for details on Controller errors.

● Controller Information

Controller information is system-defined notification information. This information does not indicate errors. It represents information level Controller events. Examples include events other than errors, such as turning the power ON and OFF, starting and stopping operation, connecting the Sysmac Studio online, and downloading user programs.



8-6-4 User-defined Events (User-defined Errors and Information)

Introduction

These errors and information are defined by the user. You can use instructions to create them.

● User-defined Errors

These errors are defined by the user. Use the Create User-defined Error (SetAlarm) instruction to create user-defined errors. When this instruction is executed, a user-defined error is recorded in the event log.

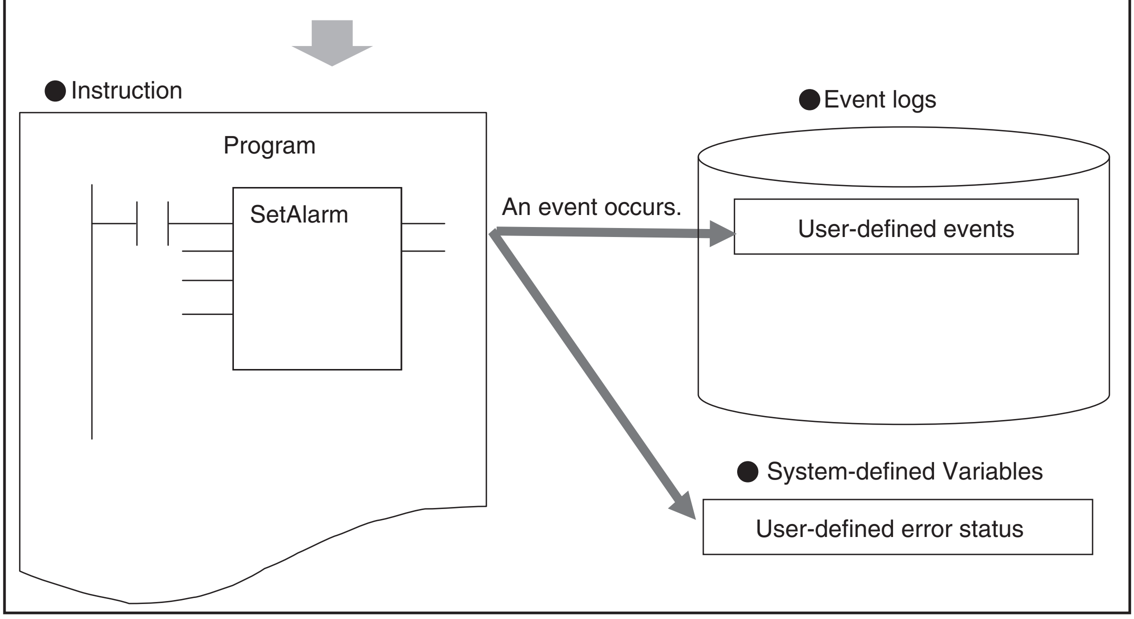
The corresponding system-defined variable changes to TRUE. User-defined errors are not reset when the operating mode changes.

● User-defined Information

User-defined information is user-defined notification information. This information does not indicate errors. Use the Create User-defined Information (SetInfo) instruction to create user-defined information. When this instruction is executed, user-defined information is recorded in the event log.

●Event Setting Tables

Event code	Event name	Level	Group	Details
100	Failure X	User fault level 1	Control A (001)	



Application Procedures

Use the following procedures.

● User-defined Errors

1. Register a user-defined error in the Event Setting Table.

↓

2. Execute the Create User-defined Error (SetAlarm) instruction.
   (Specify an event code that is defined in the Event Setting Table.)

↓

3. A user-defined error occurs.

↓

4. The corresponding system-defined variable _AlarmFlag (User-defined Error Status) changes to TRUE. Execute any process for that condition.

↓

5. Check the user-defined error in the event log with the Sysmac Studio, an instruction, or an HMI.

● User-defined information

1. Register user-defined information in the Event Setting Table.

↓

2. Execute the Create User-defined Information (SetInfo) instruction.

↓

3. Check the record in the event log.

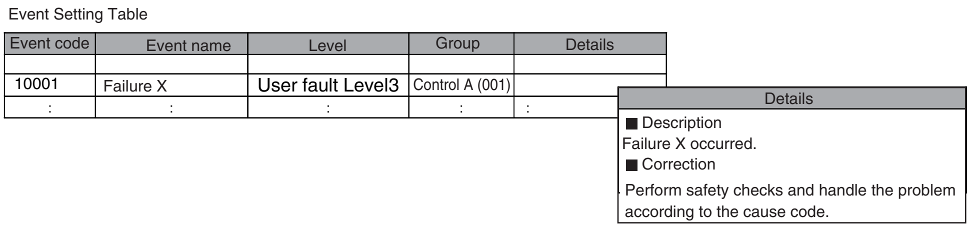
Setting the Event Setting Table

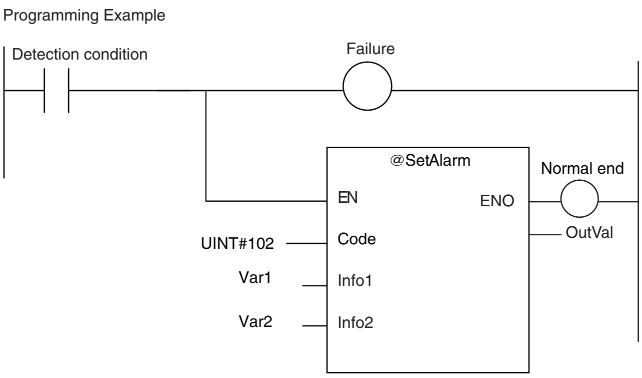
To create a user-defined error or user-defined information, register the user-defined error or user-defined information in the Event Setting Table in the Sysmac Studio in advance.

The user events that you set here can be displayed on the Sysmac Studio or an HMI with the same information.

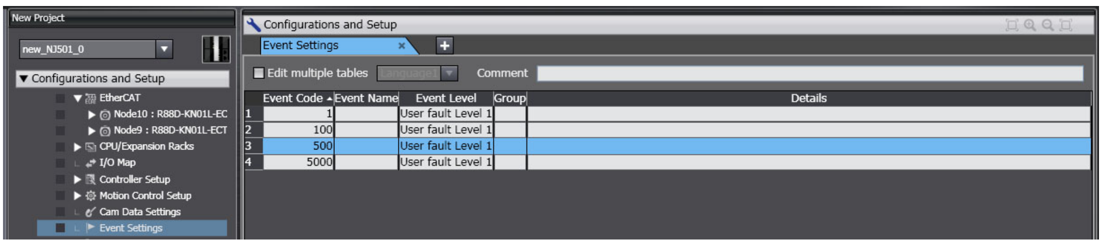
You can register up to 5,120 events in the Event Setting Table.

Event Setting Table





The following items are set in the Event Setting Table.



● Contents of the Event Setting Table

Item	Description	Values
Event Code	You can specify a number to identify the event according to the event level.	User-defined error: 1 to 40,000 User-defined information: 40,001 to 60,000
Event Name	You can include a title for the event.	128 characters max.
Event Level	You can specify the level of the event. The level is indicated with a number. The lower the number is, the higher the level is.	User-defined error: User fault levels: 1 to 8 User-defined information: User information
Group	You can specify a group name to represent the location or type of the event. You can use user-defined groupings for the events.	32 characters max. There are no restrictions on the characters that can be used. Case sensitive. Reserved words: None
Details	You can include a message that describes the event. The user can enter any text string. The message is used when the event is displayed on the Sysmac Studio or an HMI.	1,024 characters max. There are no restrictions on the characters that can be used. Case sensitive. Reserved words: None
Error details that are displayed on the HMI when a major fault level Controller error occurs	Refer to the additional information that is given below on displaying user messages on an HMI when a major fault level Controller error occurs for more details.	128 characters max. There are no restrictions on the characters that can be used. Case sensitive. Reserved words: None
Comment	The comment is attached for each set of table entries.	

Additional Information

You can set up to nine different languages for the same event code for different regions and users. On the Sysmac Studio, you can import an Event Setting Table from a Microsoft Excel file via the clipboard.

Additional Information

Displaying User Messages on an HMI When a Major Fault Level Controller Error Occurs:

When a major fault level Controller error occurs, the user program execution stops.
The NJ/NX-series Controllers can display user messages on an HMI when a major fault level Controller error occurs. You can set the display messages under the list of user-defined events in the Event Setting Table on the Sysmac Studio.

Event Levels and Event Codes

Event classification	Level	Event level category*	Range of corresponding event code	Description
User-defined errors	High	User fault Level1	1 to 5000	Select from eight levels.
	▲	User fault Level2	5001 to 10000	
		User fault Level3	10001 to 15000	
		User fault Level4	15001 to 20000	
		User fault Level5	20001 to 25000	
		User fault Level6	25001 to 30000	
		User fault Level7	30001 to 35000	
	▼ Low	User fault Level8	35001 to 40000	
User-defined Information	Lowest	User Information	40001 to 60000	The event type is user-defined information.

* User-defined error levels are separate from Controller error levels.

Precautions for Correct Use

If you update the Event Setting Table and transfer it to the CPU Unit, the event logs for user-defined events still contain old information. This can result in inconsistencies with the new Event Setting Table. Program operations with caution.

Related Instructions

There are instructions that you can use to create and check user-defined errors and to clear existing user-defined errors.

Creating and Clearing User-defined Errors

Use the following instructions to create and reset user-defined errors and to create user-defined information. Up to 32 events per level can occur simultaneously, for a total of 256 possible simultaneous events.

Instruction name	Instruction	Description
Create User-defined Error	SetAlarm	The SetAlarm instruction creates a user-defined error.
Reset User-defined Error	ResetAlarm	The ResetAlarm instruction resets a user-defined error.
Create User-defined Information	SetInfo	The SetInfo instruction records the specified user-defined information in the event log.

Checking for User-defined Errors

You can use the Get User-defined Error Status (GetAlarm) instruction to obtain the status of the current user-defined errors and the highest priority event level and code of the current user-defined errors.

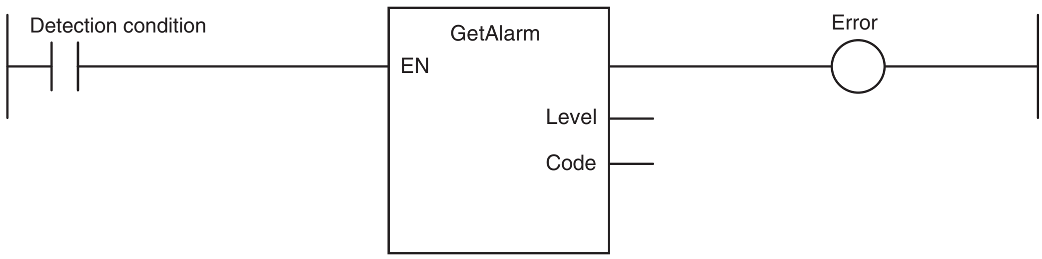
Example:

Event Setting Table

Event code	Event name	Level	Group	Details
10001	Failure X	User fault Level3	Control A (001)	
:	:	:	:	:

Details
■ Description Failure X occurred. ■ Correction Perform safety checks and handle the problem according to the cause code.

Actual Programming



Additional Information

You can use user-defined errors to add a message on possible corrections or other information when a Controller error occurs. Use instructions such as the GetPLCError instruction to obtain information about the error status or event code when a Controller error occurs. You can then use the information to trigger a user-defined error.

Example 1

When a Low Battery Voltage error occurs, the event code (16#000B0000) is obtained and the following message is displayed.

Battery is dead. Apply power for at least five minutes before changing the Battery. Install a new Battery within five minutes of turning OFF the power supply.

Example 2

When a partial fault level Controller error occurs, the event error level is obtained (highest level status: 2) and the following message is displayed.

A device failed. Call the following number for support. Repair Contact Hours: 8:00 AM to 9:00 PM TEL: xxx-xxxx-xxxx

System-defined Variables Related to User-defined Errors

Variable name	Meaning	Description	Data type	R/W
_AlarmFlag	User-defined Error Status	The bit corresponding to the event level is TRUE while there is a user-defined error. Bits 00 to 07 correspond to user fault levels 1 to 8.	WORD	R

Records in Event Log

An event is recorded in the event log when you create user-defined information or a user-defined error, or when you use the ResetAlarm instruction to reset an error. When this happens, the time, event code, event level, and attached information 1 and 2 are recorded in the user-defined event log in the event logs.

Reset User-defined Errors

User-defined errors are cleared when the power supply to the NJ/NX-series Controller is turned ON. You can also clear errors with the Sysmac Studio, the Reset User-defined Error instruction (ResetAlarm) and an HMI.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details.

8-7 Changing Event Levels

Errors, status changes, and user-defined events that occur in the NJ/NX-series Controller are all called events. You can tell what type of event has occurred by viewing the display in Sysmac Studio, or by checking the indicators on the front panel of the CPU Unit.

There are two types of events: Controller events that are defined in the system and user-defined events. The Controller events are further classified into five event levels. Refer to *Event Levels* on page 8-55 for details on event levels.

You can change the event levels that are assigned to some of the Controller events.

Version Information

A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to change event levels.

8-7-1 Applications of Changing Event Levels

The lighting pattern for the indicators on the front panel of the CPU Unit is predefined according to the event level that is assigned to each Controller event. You can change the event level for some events to change how the Controller operates when that event occurs.

For example, the ERROR indicator flashes for minor fault level events and stays unlit for observation level events. You can change the lighting pattern of the ERROR indicator so that it goes out or flashes for a given event.

Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for details on how the Controller operates for different event levels.

8-7-2 Events for Which the Event Level Can Be Changed

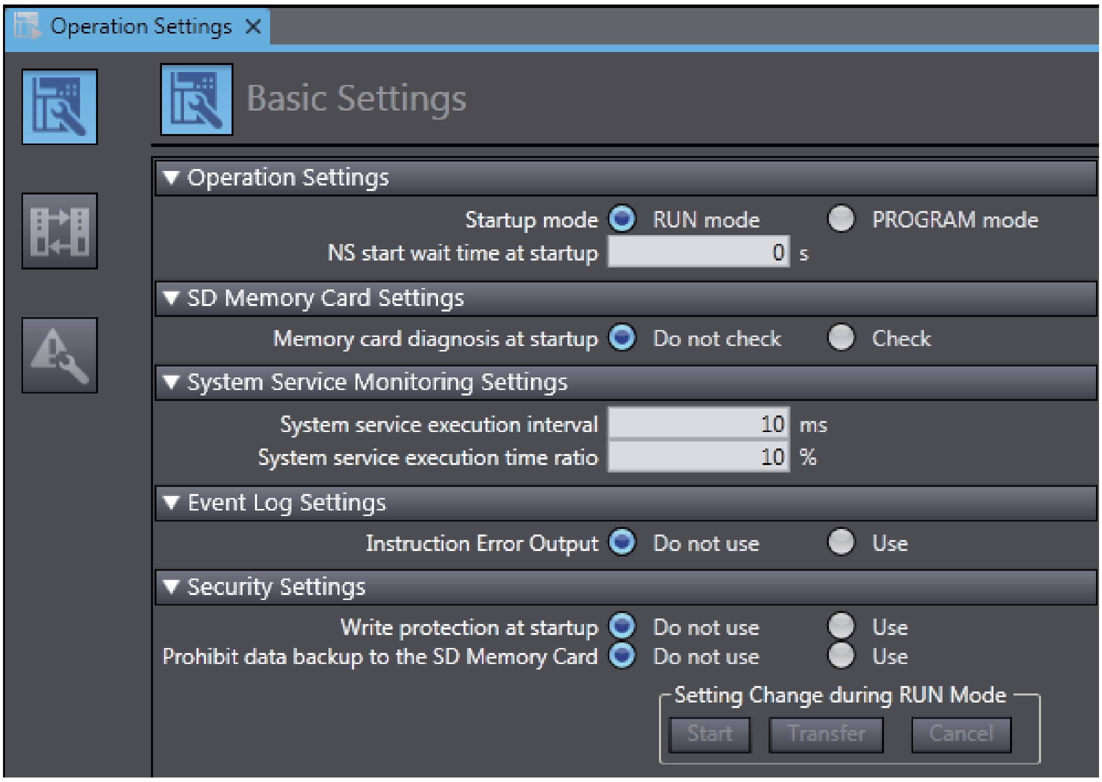
Whether an event level can be changed depends on the specific event.

Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for details on the types and levels of the Controller events, and whether the event levels can be changed.

8-7-3 Procedure to Change an Event Level

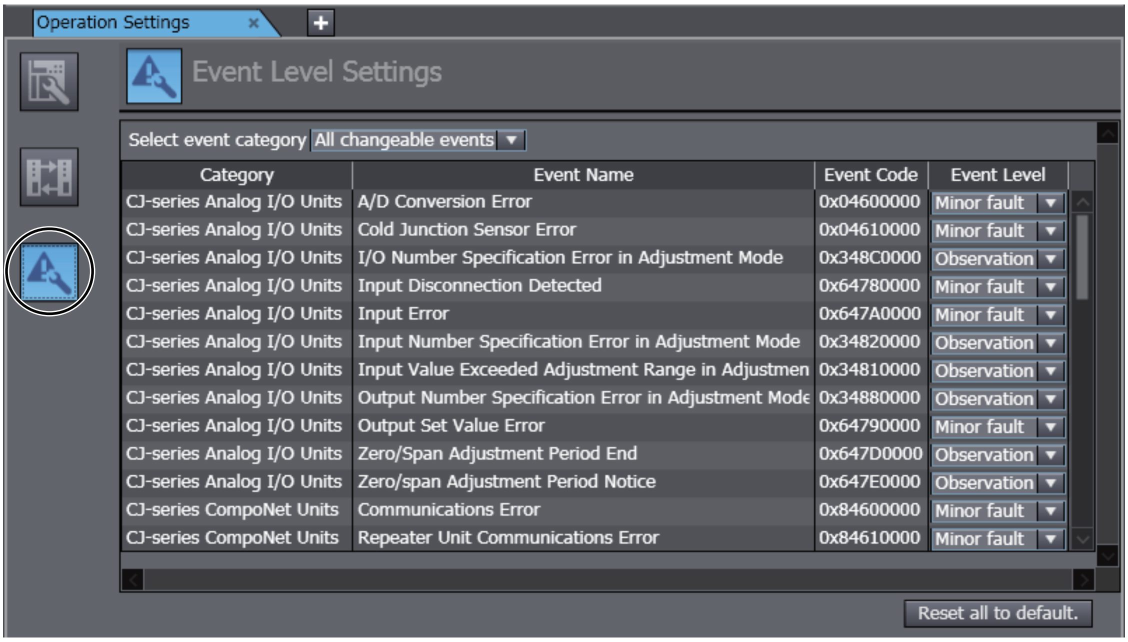
1 Double-click **Operation Settings** under **Configurations and Setup** – **Controller Setup** in the Sysmac Studio. Or right-click **Operation Settings** and select ***Edit*** from the menu.

The Basic Settings Display is displayed on the Operation Setting Tab Page in the Edit Pane.



2 Click the **Event Level Settings** Button.

A list of the events for which you can change the event level is displayed.



3 Change the levels of the required events in the *Event Level* column.

Precautions for Correct Use

If you change an event level on the Sysmac Studio and download the event level setting to the Controller when the event already exists on the Controller, the event will be reset when the download is started. If the same event occurs again while the download is in progress, the Controller will operate according to the previous event level. If the same event occurs after the download is completed, the Controller will operate according to the new level.

9

Backup Functions

This section describes the backup functions for the settings in an NJ/NX-series Controller. There are different types of backup functions that handle different data or different storage locations. First an overall description of the backup functions is provided followed by descriptions of the individual functions.

9-1 The Backup Functions

The following three functions are supported for data backup for an NJ/NX-series Controller.

Function	Description
Backing up data	You can back up all of the data in the Controller to an SD Memory Card or to a computer. The file that is saved is called a backup file.
Restoring data	You can transfer the contents of a backup file on the SD Memory Card or computer to the Controller. The data in the Controller is restored to the data at the time the backup file was made.
Verifying data	You can compare the contents of a backup file on the SD Memory Card or computer with the data in the Controller to see if they are the same.

The following items are described for the backup functions.

Item	Description
Applications of backup functions	Effective usage of the backup functions is described.
Examples of operating procedures for the backup functions	The backup functions are executed with simple procedures. Examples are provided.
Data that is backed up	The data that can be saved with the backup functions from the connected Units and slaves is described.
Types of backup functions	There are different types of backup functions that differ in where the data is saved. The types of backup functions and the difference between them are described.
Relation between the different types of backup functions and data groups	Different types of backup functions handle different data groups. The relation between the different types of backup functions and data groups is described.
Applicable range of the backup functions	The connected Units and slaves for which you can save data with the backup functions are described.

9-1-1 Applications of Backup Functions

You can use the backup functions in the following instances.

Item	Application
Program and setting changes	When you change the user program and settings for equipment that is currently in operation.
Hardware replacements	When you replace the hardware for the CPU Unit, other Units, or slaves.
Troubleshooting equipment failures	When you want to save data in the Controller to analyze the cause of an error that occurs in the equipment.
Equipment backup and recovery	When an error occurs in the equipment, and when you want to restore the equipment with data from an normal operating status. When you want to backup the data in the equipment while it is in operation.
Manufacture of equipment	When you want to manufacture the same equipment and need to transfer the data from the existing equipment to new equipment in its initial state.

9-1-2 Examples of Operating Procedures for the Backup Functions

You can use the backup functions to easily back up, restore, and verify Controller data.

This section provides the procedures to back up, restore, and verify data with an SD Memory Card by using the DIP switch on the front panel of the CPU Unit.

Precautions for Correct Use

For NX-series CPU Units, eight pins, pins 1 to 8, are provided on the DIP switch. Before you use the backup functions, set all of pins 5 to 8 to OFF.

Backup Procedure

● Preparations

1. Insert the SD Memory Card into the CPU Unit.
2. Set pins 1 to 4 on the DIP switch on the CPU Unit as follows: 1: OFF, 2: OFF, 3: ON, and 4: OFF.

● Executing the Backup

3. Press the SD Memory Card power supply switch for 3 seconds.

   The backup is started. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds.

   When the backup operation is completed, the SD PWR indicator will stop flashing and remain lit.

● Ending the Backup Procedure

4. Set all of pins 1 to 4 on the DIP switch on the CPU Unit to OFF.
5. Press the SD Memory Card power supply switch to turn OFF the SD PWR indicator.
6. Remove the SD Memory Card.

Restoration Procedure

● Preparations

1. Turn OFF the power supply to the NJ/NX-series Controller and to the EtherCAT slaves.
2. Insert the SD Memory Card that contains the backup file into the CPU Unit.
3. Set pins 1 to 4 on the DIP switch on the CPU Unit as follows: 1: OFF, 2: OFF, 3: ON, and 4: ON.

● Restoring Data

4. Turn ON the power supply to the NJ/NX-series Controller and to the EtherCAT slaves.

   The restoration operation is started. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds.

   When the restoration operation is completed, the SD PWR indicator will stop flashing and remain lit.

- **Ending the Restoration Procedure**

5 Press the SD Memory Card power supply switch to turn OFF the SD PWR indicator.

6 Turn OFF the power supply to the NJ/NX-series Controller and to the EtherCAT slaves.

- **Starting Normal Operation**

7 Remove the SD Memory Card.

8 Set all of pins 1 to 4 on the DIP switch on the CPU Unit to OFF.

9 Turn ON the power supply to the NJ/NX-series Controller and to the EtherCAT slaves.

Precautions for Correct Use

Restoring Data When EtherCAT Slaves Are Connected

- Always cycle the power supply to the NJ/NX-series Controller and the EtherCAT slaves after you restore data when EtherCAT slaves are connected. If you start operation without cycling the power supply, the Controller may perform unexpected operation.
- To verify the data after you restore data with EtherCAT slaves connected, first turn OFF the power supply to the NJ/NX-series Controller and EtherCAT slaves, and then start in Safe Mode before you perform the verification procedure. If you cycle the power supply normally, the Controller will start operation before you can perform the verification procedure. That means that operation could be started with data that is not correct. For information on Safe Mode, refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503).

Verification Procedure

- **Preparations**

1 Insert the SD Memory Card that contains the backup file into the CPU Unit.

2 Set all of pins 1 to 4 on the DIP switch on the CPU Unit to OFF.

- **Verifying the Data**

3 Press the SD Memory Card power supply switch for 3 seconds.

Data comparison is started. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds.

If the verification operation is completed and the data is the same, the SD PWR indicator will stop flashing and remain lit.

If the verification operation is completed and differences were found in the data, the SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds.

- **Ending the Verification Procedure**

4 Press the SD Memory Card power supply switch to turn OFF the SD PWR indicator.

5 Remove the SD Memory Card.

9-1-3 Data That Is Backed Up

The following data is backed up. This section describes the backup functions based on the following data groups for the backup data.

Data group	Data items
User program and settings	EtherCAT configuration (EtherCAT slave configuration and EtherCAT master settings) Unit Configuration and Unit Setup I/O Map Controller Setup (Operation Settings and Built-in EtherNet/IP Port Settings) Motion Control Setup Cam Data Settings Event Setup Task Setup Data Trace Settings Tag Data Link Tables Controller name Operation authority verification Built-in clock (time zone setting) POUs Data (data types and global variables)
Present values of variables	Values of variables with a Retain attribute*1
Present values of memory used for CJ-series Units	Values of the Holding, DM, and EM Areas in the memory for CJ-series Units*2
Units and slaves settings	Backup parameters for EtherCAT slaves*3, Parameters in the CJ-series Units*2*4
Absolute encoder home offset	The set value to restore the actual position of a Servo Drive with an absolute encoder

*1 Of the system-defined variables with a Retain attribute, some variables are not applicable for the data backup function. Refer to *A-5 Specifications for Individual System-defined Variables* for details on the specifications for individual system-defined variables.

*2 You can use memory for CJ-series Units and parameters in the CJ-series Units only with NJ-series CPU Units.

*3 A part or all of the set parameters are not backed up for some EtherCAT slave models. For the details on the target EtherCAT slaves for the data backup function, refer to *9-8-4 EtherCAT Slaves for Which You Can Back Up Data*.

*4 Refer to the *CJ-series CJ2 CPU Unit Hardware User's Manual* (Cat. No. W472) for details on the data that is backed up.

Precautions for Safe Use

Precautions on the Absolute Encoder Home Offset

The absolute encoder home offsets are backed up with a Battery in the CPU Unit as absolute encoder information. If any of the following conditions is met, clear the absolute encoder home offsets from the list of data items to restore, and then restore the data. Then, define the absolute encoder home again. If you do not define home, unintended operation of the controlled system may occur.

- The Servomotor or Servo Drive was changed since the data was backed up.
- The absolute encoder was set up after the data was backed up.
- The absolute data for the absolute encoder was lost.

9-1-4 Types of Backup Functions

There are backup functions for the NJ/NX-series Controllers that save data to SD Memory Cards and others that save data to a computer. Also, there are three methods used to execute the backup functions: the CPU Unit front-panel DIP switches, system-defined variables, and the Sysmac Studio.

Functions That Save Data to SD Memory Cards

The SD Memory Card backup functions are used to back up, restore, and compare data on SD Memory Cards. Related functions include disabling backups to SD Memory Cards and automatic transfers from SD Memory Cards.

Function name		Description	Operating method			Reference
			CPU Unit front-panel DIP switch	System-defined variables	Sysmac Studio	
SD Memory Card backups	Backing up data	The Controller data is saved in a backup file on the SD Memory Card.	✓	✓	✓	*9-2-1 Backup (Controller to SD Memory Card)*
	Restoring data	The data in a backup file on the SD Memory Card is transferred to the Controller.	✓			*9-2-2 Restore (SD Memory Card to Controller)*
	Verifying data	The Controller data and the data in a backup file on the SD Memory Card are compared.	✓	✓	✓	*9-2-3 Verify (between Controller and SD Memory Card)*
Disabling backups to SD Memory Cards		You can disable backing up data to SD Memory Cards.			✓	*9-3 Disabling Backups to SD Memory Cards*
Automatic transfers from SD Memory Cards		When the power supply is turned ON, the data in a backup file on the SD Memory Card is automatically transferred to the Controller. After the data transfer, the operating mode of the CPU Unit will change to the mode that is specified in Startup Mode setting.	✓			*9-4 Automatic Transfers from SD Memory Cards*

Functions That Save Data to the Computer

The Sysmac Studio Controller backup functions are used to back up, restore, and compare data on the computer. Importing and exporting Sysmac Studio backup file data are used to save and read different types of data between the Sysmac Studio projects and backup files on the computer without using a Controller. The Sysmac Studio variable and memory backup functions are used to back up battery-backup present values to the computer and restore them from the computer.

Function name		Description	Operating method			Reference
			CPU Unit front-panel DIP switch	System-defined variables	Sysmac Studio	
Sysmac Studio Controller backups	Backing up data	The Controller data is saved in a backup file on the computer.			✓	*9-5-1 Backup (Controller to Computer)*
	Restoring data	The data in a backup file on the computer is transferred to the Controller.			✓	*9-5-2 Restore (Computer to Controller)*
	Verifying data	The Controller data and the data in a backup file on the computer are compared.			✓	*9-5-3 Verify (between Controller and Computer)*
Importing and exporting Sysmac Studio backup file data	Exporting data	The data is exported from the project on the Sysmac Studio to a backup file without using a Controller.			✓	*9-6 Importing and Exporting Sysmac Studio Backup File Data*
	Importing data	The data in the backup file is imported into the Sysmac Studio project without using a Controller.			✓	
Sysmac Studio variable and memory backup functions	Backing up data	You can back up the present values of data that is backed up by a battery to an XML file on the computer.			✓	*9-7 Sysmac Studio Variable and Memory Backup Functions*
	Restoring data	You can restore the present values of data that is backed up by a battery from the computer to the CPU Unit.			✓	

Version Information

A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to use the following backup functions: SD Memory Card backups, automatic transfers from SD Memory Cards, Sysmac Studio Controller backups, and importing and exporting Sysmac Studio backup file data.

Additional Information

The backup functions are executed as a system service. This means that if you perform a backup or verification operation in RUN mode with an NJ-series CPU Unit, it may take time for the operation to be completed. If you perform a backup or verification operation in RUN mode, make sure that the sufficient execution time is allocated for the system service. You can reduce the processing time by performing the system service in PROGRAM mode.

9-1-5 Relation Between the Different Types of Backup Functions and Data Groups

Different types of backup functions handle data for different data groups. The relation between the different types of backup functions and data groups is given in the following table.

(OK: Applicable, NA: Not applicable)

Type of backup function		Data group				
		User program and settings	Present values of variables	Present values of memory used for CJ-series Units*1	Units and slaves settings	Absolute encoder home offsets
SD Memory Card backups	Backing up data	OK	OK*2	OK*3	OK	OK
	Restoring data*4	OK	OK*1	OK*2	OK	OK
	Verifying data*3	OK*5	NA	NA	OK	NA
Automatic transfers from SD Memory Cards*6		OK	OK*1	OK*2	NA	NA
Sysmac Studio Controller backups	Backing up data	OK	OK*1	OK*2	OK*7	OK
	Restoring data	OK	OK*1	OK*2	OK*6	OK
	Verifying data	OK*4	NA	NA	OK*6	NA
Importing and exporting Sysmac Studio backup file data	Exporting backup file data	OK*8	NA	NA	NA	NA
	Importing backup file data	OK*7	NA	NA	OK	NA
Sysmac Studio variable and memory backup functions	Backing up and restoring data	NA	OK*1	OK*2	NA	OK

*1 You can use memory for CJ-series Units only with NJ-series CPU Units.

*2 The backup data is processed only for the present values of variables that are specified for retention with the Retain attribute.

*3 The backup data is processed only for the present values of addresses that are specified for retention with the Retain attribute in the memory for CJ-series Units.

*4 For all of the data groups except for the user program and setting group, only the items that are specified to be restored in the restore command file are restored.

*5 Of the user program and setting data groups, the Data Trace Settings are not compared.

*6 For all of the data groups except for the user program and setting group, only the items that are specified to be transferred in the automatic transfer command file are transferred.

*7 If the CJ-series Units are specified for backup, the parameters in the CJ-series Units are backed up. If the EtherCAT slaves are specified for the backup, parameters for the EtherCAT slaves are backed up.

*8 The following data is not processed: Tag data link settings for the built-in EtherNet/IP port, operation authority verification, and the Data Trace Settings.

Additional Information

The files that are handled for backing up variables and memory from the Sysmac Studio are not compatible with other backup files. Refer to *9-7 Sysmac Studio Variable and Memory Backup Functions* for details on the Sysmac Studio variable and memory backup functions.

9-1-6 Applicable Range of the Backup Functions

Different types of backup functions handle data for different Units or slaves. The applicable Units and slaves for each backup function are given in the following table.

(OK: Applicable, NA: Not applicable)

Type of backup function	Units/slaves					
	NJ/NX-series CPU Unit	EtherCAT slaves*1	CJ-series Units*2		Ether-Net/IP slaves	Computer and HMIs
			Units and Master Units	Slaves		
SD Memory Card backups	OK	OK*3	OK	NA	NA	NA
Automatic transfers from SD Memory Cards	OK	NA	NA	NA	NA	NA
Sysmac Studio Controller backups	OK	OK*3	OK	NA	NA	NA
Importing and exporting Sysmac Studio backup file data	OK	OK*4	NA	NA	NA	NA
Sysmac Studio variable and memory backup functions	OK	NA	NA	NA	NA	NA

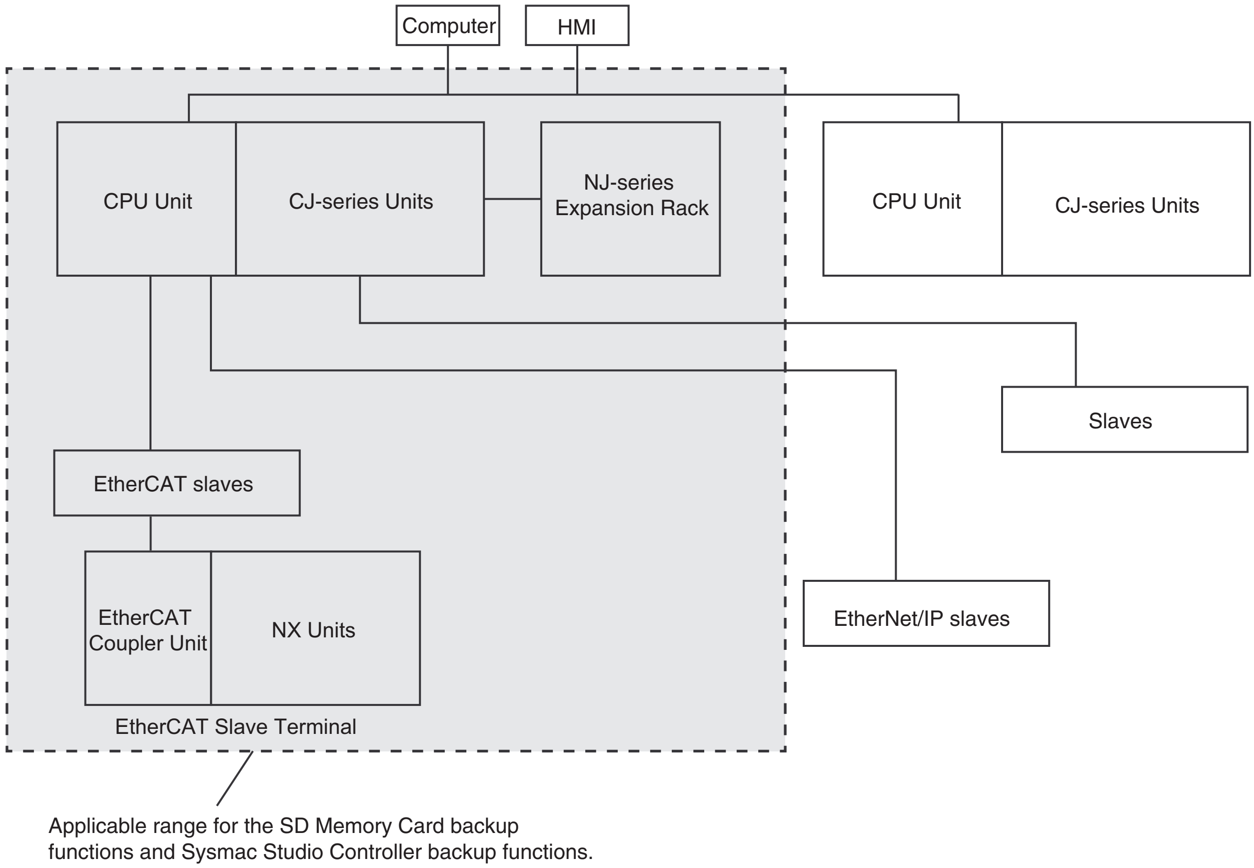
*1 EtherCAT Slave Terminals are included. If EtherCAT Slave Terminals are set for backup, the backup function applies to both the EtherCAT Coupler Unit and the NX Units.

*2 You can use CJ-series Units only with NJ-series CPU Units.

*3 This does not apply to Safety Control Units. Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for information on importing and exporting settings for a Safety Control Unit.

*4 Only importing data is possible. Exporting is not possible.

The Units and slaves that are shown in the following figure are covered by the SD Memory Card backup functions and Sysmac Studio Controller backup functions.



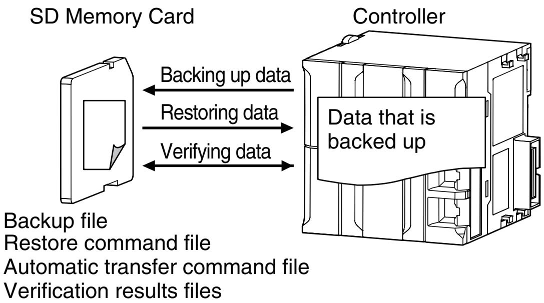
Note You can use CJ-series Units and NJ-series Expansion Racks only with the NJ-series CPU Unit.

Version Information

A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to connect EtherCAT Slave Terminals.

9-2 SD Memory Card Backups

You can use SD Memory Cards to back up, restore, and verify Controller data.



When you back up data, the backup file, restore command file, and automatic transfer command file are created in the specified directory on the SD Memory Card. When you verify data, the verification results files are created in the specified directory. All of these files are collectively referred to as backup-related files. The functions of the backup-related files are given in the following table.

File	Function			
	Contents	Backing up data	Restor-ing data	Verifying data
Backup files	This file contains the Controller data that is handled by the functions that are related to data backup.	Created.	Accessed.	Accessed.
Restore command file	This file specifies the data groups to restore when restoring data. You can edit this file with a text editor on a computer to specify the data groups to restore.	Created.	Accessed.	Accessed.
Automatic transfer command file	This file specifies the data groups to transfer when automatically transferring data from a SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer.	Created.	Nothing is done.	Nothing is done.
Verification results files	These files contain the verification results after data is verified.	Nothing is done.	Nothing is done.	Created.

The execution method for the functions, applicable directory, and applicable operating modes are given in the following table.

Procedure	Directory*1	Applicable operating modes		
		Backing up data	Restoring data	Verifying data
CPU Unit front-panel DIP switch*2	The root directory	RUN mode and PROGRAM mode	At startup	RUN mode and PROGRAM mode
System-defined vari-ables*3*4	The directory that you specified in the system-defined variable	RUN mode and PROGRAM mode	Execution is not possible.	RUN mode and PROGRAM mode
SD Memory Card Window in Sysmac Studio	The directory that you specified on the SD Mem-ory Card Window	RUN mode and PROGRAM mode	Execution is not possible.	RUN mode and PROGRAM mode

*1 You can specify a directory only on the SD Memory Card.

*2 Before you restore or verify data, save the backup file and restore command file in the root directory.

*3 This method is used to control the backup functions from an HMI. You cannot access these system-defined variables from the user program.

*4 Make arrangements to prevent backup or verification operations from being performed on HMIs while a backup or verification operation is in progress. Otherwise, the intended operation may not occur.

9-2-1 Backup (Controller to SD Memory Card)

This operation is used to save data in the Controller to the SD Memory Card in the CPU Unit.

Processing Contents

- This backup operation processes all data groups.
- When you back up data, the backup file, restore command file, and automatic transfer command file are created in the specified directory on the SD Memory Card.
- If the backup-related files are already in the specified directory, they are overwritten.
- If an error occurs while writing the backup-related files to the SD Memory Card, the previous backup-related files will be deleted. Also, the new backup-related files will not be created.
- If an error occurs before the new backup-related files are created, the previous files are retained and the new files are not created.
- The SD Memory Card power supply switch is disabled when a backup is in progress.
- The SD Memory Card will remain mounted after completion of the backup.

Procedure

● Backing Up Data with the CPU Unit Front-panel DIP Switch

Processing stage	Procedure
Start command	The backup starts when the SD Memory Card power supply switch is pressed for 3 seconds with the DIP switch pins set as follows: 1: OFF, 2: OFF, 3: ON, and 4: OFF.*1
Executing	Immediately after Starting Backup*2 The SD PWR indicator will light, go out for 0.5 seconds, and then light again. While Backing Up Data The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	Normal End: The SD PWR indicator will light. Error End: The SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The indicator stop flashing and stay lit when the SD Memory Card power supply switch is pressed.*1

*1 For the NX-series CPU Unit, set all of pins 5 to 8 on the DIP switch to OFF.

*2 If an SD Memory Card is not inserted, the SD PWR indicator will not light.

Backing Up Data with the *_Card1BkupCmd* (SD Memory Card Backup Command) System-defined Variable

Processing stage	Procedure
Start command	The name of the directory where the files are saved is stored in the *_Card1BkupCmd.DirName* (Directory Name) system-defined variable. Example: “dirA/dirB” specifies the dirB directory inside the dirA directory. The backup operation starts when you change the *_Card1BkupCmd.ExecBkup* (Execute Backup Flag) system-defined variable to TRUE.
Cancel command	You can cancel the backup operation. The backup operation ends in an error if you change the *_Card1BkupCmd.CancelBkup* (Cancel Backup Flag) system-defined variable to TRUE.
Executing	The *_Card1BkupSta.Active* (Active Flag) system-defined variable changes to TRUE. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	Normal End: The *_Card1BkupSta.Done* (Done Flag) system-defined variable changes to TRUE. Error End: The *_Card1BkupSta.Err* (Error Flag) system-defined variable changes to TRUE.

Note You cannot access these system-defined variables from the user program.

Backing Up Data from the SD Memory Card Window on the Sysmac Studio

Processing stage	Procedure
Start command	Click the **SD Memory Card Backup** Button on the SD Memory Card Window in the Sysmac Studio, specify the directory to save the backup file in, and execute the backup.
Executing	The progress of the backup is displayed in the dialog box. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	A message will appear when the backup is completed. You will then be asked to confirm whether to verify the backup data.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific procedures.

Backing Up Data with Special Instruction

Processing stage	Procedure
Start command	Execute the BackupToMemoryCard instruction in the user program.
Executing	The value of the *Busy* output variable from the BackupToMemoryCard instruction will change to TRUE. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	Normal End: The value of the *Done* output variable from the BackupToMemoryCard instruction changes to TRUE. Error End: The value of the *Error* output variable from the BackupToMemoryCard instruction changes to TRUE. The error code is stored in the *ErrorID* output variable from the BackupToMemoryCard instruction.

Version Information

A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the BackupToMemoryCard instruction.

Related System-defined Variables

The system-defined variables that are related to the operation when system-defined variables are used to back up data are shown below. Refer to *A-5 Specifications for Individual System-defined Variables* for details on system-defined variables.

Variable / Member name	Meaning	Function	Data type	R/W
_Card1BkupCmd*	SD Memory Card Backup Commands		_sBKUP_CMD	RW
ExecBkup*	Execute Backup Flag	Change this variable to TRUE to back up Controller data to an SD Memory Card.	BOOL	RW
CancelBkup*	Cancel Backup Flag	Change this variable to TRUE to cancel backing up data to an SD Memory Card.	BOOL	RW
DirName*	Directory Name	Use this variable to specify the directory name in the SD Memory Card for which to back up data.	STRING(64)	RW
_Card1BkupSta*	SD Memory Card Backup Status		_sBKUP_STA	R
Done*	Done Flag	TRUE when a backup is completed.	BOOL	R
Active*	Active Flag	TRUE when a backup is in progress.	BOOL	R
Err*	Error Flag	TRUE when processing a backup ended in an error.	BOOL	R
_BackupBusy	Backup Function Busy Flag	TRUE when a backup, restoration, or verification is in progress.	BOOL	R

* You cannot access these system-defined variables from the user program.

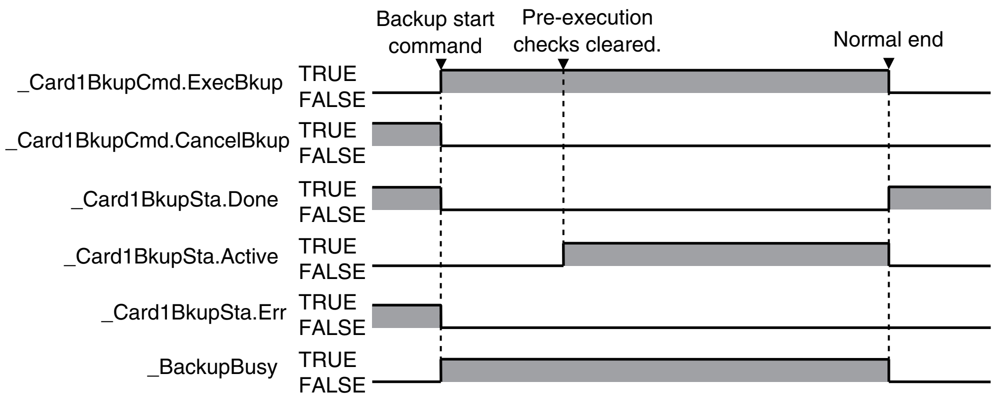
Additional Information

- Refer to the *NA-series Programmable Terminal Software User's Manual* (Cat. No. V118) for information on mapping variables when you connect an NA-series PT to the NJ/NX-series Controller.
- Refer to *A-9 Registering a Symbol Table on the CX-Designer* for the procedure to register these system-defined variables in the variable table of the CX-Designer when you connect an NS-series PT to the NJ/NX-series Controller.

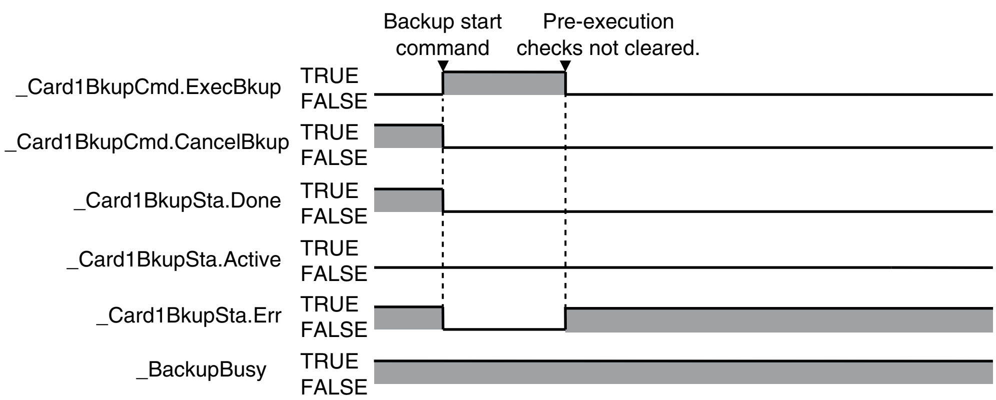
Timing Charts

The operation of the system-defined variables when they are used to backup data is shown below. In the charts, “pre-execution checks” indicates processing to check whether there is a SD Memory Card in the CPU Unit and other items before the backup starts. The value of *_Card1BkupSta.Active* (Active Flag) changes to TRUE only after all of the pre-execution checks are cleared and the actual backup is started.

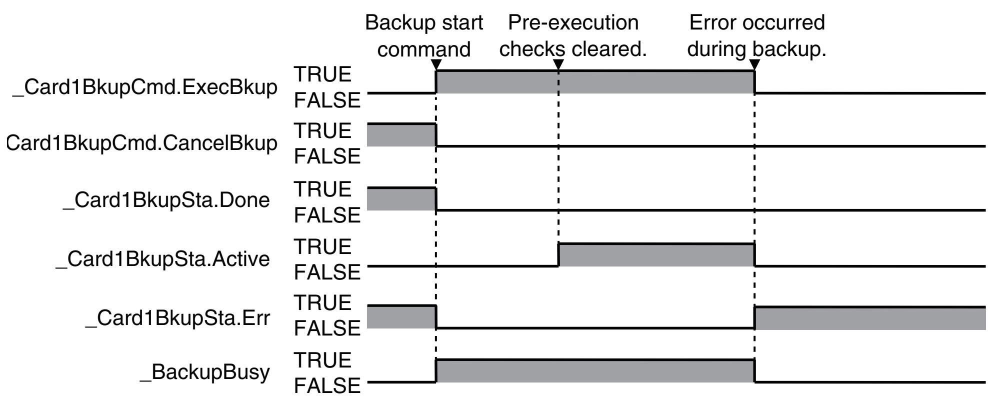
● Normal Operation



● Operation When the Backup Cannot Start Because Another Backup Function Is in Progress

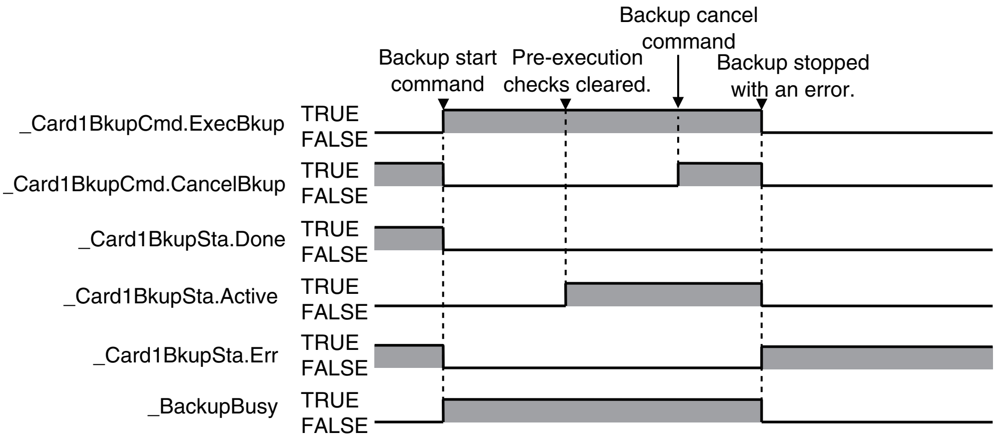


● Operation When the Backup Fails After a Normal Start



- **Operation When the Backup Is Canceled While the Backup Is in Progress**

The time required to stop the backup operation after it is canceled depends on the progress of the backup operation.



Processing Time and Backup File Size

The time that is required to back up the data depends on factors such as the CPU Unit, operating mode, Unit configuration, and user program. The size of the backup file depends on factors such as the Unit configuration and user program. Some guidelines for the backup time and backup file size are given in the following table.

CPU Unit	Operating mode	Connected EtherCAT slaves	Connected CJ-series Units	Number of user-defined POUs	User program memory size (Mbytes)	Backup time (s)	Backup file size (Mbytes)
NX701-□□□□	PRO-GRAM mode	*1	---	113	7.13	Approx. 30	29.45
NJ501-□□□□			*2	53	2.36	Approx. 50	20
NJ301-□□□□		*3		20	0.53	Approx. 30	9.85
NJ101-□□□□		*4		15	0.38	Approx. 30	9.94

*1 Thirty-two each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.

*2 Four CJ1W-SCU22 Serial Communications Units and one CJ1W-EIP21 EtherNet/IP Unit.

*3 Eight each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.

*4 Two each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.

9-2-2 Restore (SD Memory Card to Controller)

When the power supply is turned ON, you can transfer the data in a backup file on the SD Memory Card in the CPU Unit to the Controller. The only way to perform this operation is to use the front-panel DIP switch on the CPU Unit. You cannot specify the source directory for backup-related files. The backup file to restore must be stored in the root directory on the SD Memory Card.

Processing Contents

- The data in a backup file in the root directory on the SD Memory Card is transferred to the Controller.
- The data groups that are processed by the restoration operation in the RestoreCommand.ini file (restore command file) that is stored in the root directory. Refer to *9-11-3 Specifications of a Restore Command File* for details on the restore command file.
- If there is not a restore command file in the root directory of the SD Memory Card, all of the data from the backup files in the root directory that can be transferred to the Controller will be transferred.
- After the operation is completed, the operating mode will change to PROGRAM mode. You cannot start operation in this state. To start operation, turn OFF all DIP switch pins and then cycle the power supply to the Controller or reset the Controller.
- Cycle the power supply to all of the EtherCAT slaves after you restore data.
- While the data is being restored, the CPU Unit will be in startup state.
- If an error occurs in the checks that are performed before starting to restore the data, the previous data will be retained in the Controller.
- If the power supply to the Controller is interrupted while the data is being restored, a User Program/Controller Configurations and Setup Transfer Error (a major fault level Controller error) will occur. If that occurs, the data in the Controller is not dependable. Use one of the following methods to clear the error.
  - Perform the restore operation again.
  - Clear all of memory and then download the project from the Sysmac Studio.
- If the configuration for Units and slaves in the backup file does not match the actual configuration where data is restored, a Restore Execution Error will occur when you restore the data.
- If the present values of variables that are set to be retained (with the Retain attribute) are not set to be restored, the previous present values of those variables will be retained. However, the values of any variables that do not meet the retain conditions are initialized. These are the retain conditions for the variable:
  - The variable name, data type name, and data type size must be the same before and after restoring the data.
- The SD Memory Card will remain mounted after completion of the restore operation.
- The write protection for the CPU Unit that is set in the Write Protection at Startup setting is used after completion of the restore operation.

Procedure

● Backing Up Data with the CPU Unit Front-panel DIP Switch

Processing stage	Procedure
Start command	Turn ON the power supply to the Controller with the DIP switch set as follows: 1: OFF, 2: OFF, 3: ON, and 4: ON.*1
Executing	While Restoring Data The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The RUN indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly.
Execution results	Normal End: The SD PWR indicator will light. Error End: The SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The indicator stop flashing and stay lit when the SD Memory Card power supply switch is pressed.*2

*1 For the NX-series CPU Unit, set all of pins 5 to 8 on the DIP switch to OFF.

*2 If an SD Memory Card is not inserted, the SD PWR indicator will not light.

Processing Time

The time that is required to restore the data depends on factors such as the CPU Unit, Unit configuration, and user program. Guidelines for the restoration time are given in the following table.

CPU Unit	Connected EtherCAT slaves	Connected CJ-series Units	Number of user-defined POUs	User program memory size (Mbytes)	Restoration time (s)
NX701-□□□□	*1	---	113	7.13	Approx. 50
NJ501-□□□□		*2	53	2.36	Approx. 100
NJ301-□□□□	*3		20	0.53	Approx. 70
NJ101-□□□□	*4		15	0.38	Approx. 70

*1 Thirty-two each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.

*2 Four CJ1W-SCU22 Serial Communications Units and one CJ1W-EIP21 EtherNet/IP Unit.

*3 Eight each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.

*4 Two each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.

9-2-3 Verify (between Controller and SD Memory Card)

You can compare the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit.

Processing Contents

- The Controller data and the data in a backup file that is saved in the specified directory of the SD Memory Card are compared.
- The data groups that are processed by the verification operation are specified in the RestoreCommand.ini file (restore command file).
- The present values of variables, the present values in memory used for the CJ-series Units, and the absolute encoder home offsets are not compared because these values may change while the verification is in process.
- When you verify the data, the verification results file (VerifyResult.log) is created in the specified directory. The verification results are stored in this file. If a verification results file already exists in the specified directory, it will be overwritten. However, if the SD Memory Card is write-protected, the verification results files will not be created.
- If there is not a restore command file in the specified directory of the SD Memory Card, all of the data from the backup files in the specified directory that can be compared will be compared.
- If the Unit and slave configuration in the backup file is not the same as the actual configuration of the Controller, a Verification Error will occur.
- The SD Memory Card will remain mounted after completion of the verification operation.

Procedure

● Backing Up Data with the CPU Unit Front-panel DIP Switch

Processing stage	Procedure
Start command	The backup starts when the SD Memory Card power supply switch is pressed for 3 seconds with the DIP switch pins set as follows: 1: OFF, 2: OFF, 3: OFF, and 4: OFF.*1
Executing	Immediately after Starting Backup*2 The SD PWR indicator will light, go out for 0.5 seconds, and then light again. While Verifying Data The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	Normal End with No Differences Found: The SD PWR indicator will light. Normal End with Differences Found: The SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The indicator stop flashing and stay lit when the SD Memory Card power supply switch is pressed. Error End: The SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The indicator stop flashing and stay lit when the SD Memory Card power supply switch is pressed.*1

*1 For the NX-series CPU Unit, set all of pins 5 to 8 on the DIP switch to OFF.

*2 If an SD Memory Card is not inserted, the SD PWR indicator will not light.

- ***_Card1BkupCmd*** **(SD Memory Card Backup Command) System-defined Variable**

Processing stage	Procedure
Start command	The name of the directory where the files are saved is stored in the *_Card1BkupCmd.DirName* (Directory Name) system-defined variable. Example: “dirA/dirB” specifies the dirB directory inside the dirA directory. The verification operation starts when you change the *_Card1BkupCmd.ExecVefy* (Execute Verify Flag) system-defined variable to TRUE.
Cancel command	You can cancel the verification operation. The verification operation ends in an error if you change the *_Card1BkupCmd.CancelVefy* (Cancel Verify Flag) system-defined variable to TRUE.
Executing	The *_Card1VefySta.Active* (Active Flag) system-defined variable changes to TRUE. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	Normal End with No Differences Found: The *_Card1BkupSta.Done* (Done Flag) and the *_Card1BkupSta.VefyRslt* (Verify Result Flag) system-defined variables change to TRUE. Normal End with Differences Found: The *_Card1BkupSta.Done* (Done Flag) system-defined variable changes to TRUE and the *_Card1BkupSta.VefyRslt* (Verify Result Flag) system-defined variable changes to FALSE Error End: The *_Card1BkupSta.Err* (Error Flag) system-defined variable changes to TRUE.

Note You cannot access these system-defined variables from the user program.

- **Backing Up Data from the SD Memory Card Window on the Sysmac Studio**

Processing stage	Procedure
Start command	Click the **Compare SD Memory Card Backup** Button on the SD Memory Card Window in Sysmac Studio, specify the directory that contains the file to compare, and execute the verification.
Executing	The progress of the verification is displayed in the dialog box. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly.
Execution results	The results of the verification are displayed in the dialog box.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific procedures.

Related System-defined Variables

The system-defined variables that are related to the operation when system-defined variables are used to restore data are shown below. Refer to *A-5 Specifications for Individual System-defined Variables* for details on system-defined variables.

Variable name	Member name	Meaning	Function	Data type	R/W
_Card1BkupCmd*		SD Memory Card Backup Commands		_sBKUP_CMD	RW
	ExecVefy*	Execute Verify Flag	Change this variable to TRUE to compare the Controller data to a backup file in the SD Memory Card.	BOOL	RW
	CancelVefy*	Cancel Verify Flag	Change this variable to TRUE to cancel comparing the Controller data to a backup file in the SD Memory Card.	BOOL	RW
	DirName*	Directory Name	Use this variable to specify the directory name in the SD Memory Card for which to back up data.	STRING(64)	RW
_Card1VefySta*		SD Memory Card Verify Status		_sVEFY_STA	R
	Done*	Done Flag	TRUE when a verification is completed.	BOOL	R
	Active*	Active Flag	TRUE when a verification is in progress.	BOOL	R
	VefyRslt*	Verify Result Flag	TRUE if the data was the same. FALSE if differences were found.	BOOL	R
	Err*	Error Flag	TRUE when processing a verification ended in an error.	BOOL	R
_BackupBusy		Backup Function Busy Flag	TRUE when a backup, restoration, or verification is in progress.	BOOL	R

* You cannot access these system-defined variables from the user program.

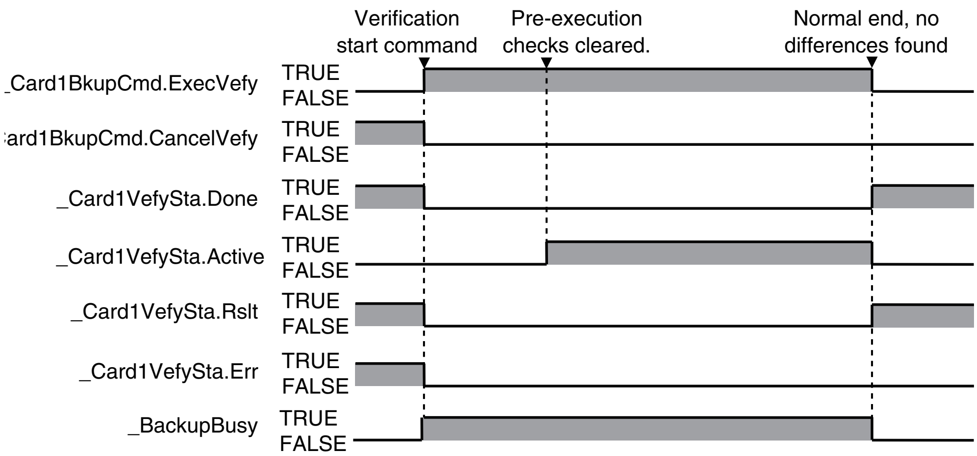
Additional Information

- Refer to the *NA-series Programmable Terminal Software User's Manual* (Cat. No. V118) for information on mapping variables when you connect an NA-series PT to the NJ/NX-series Controller.
- Refer to *A-9 Registering a Symbol Table on the CX-Designer* for the procedure to register these system-defined variables in the variable table of the CX-Designer when you connect an NS-series PT to the NJ/NX-series Controller.

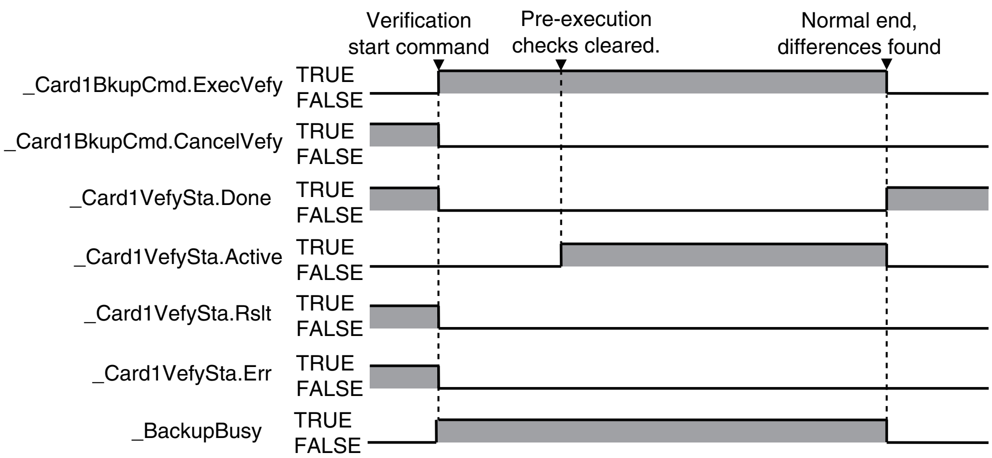
Timing Charts

The operation of the system-defined variables when they are used to verify data is shown below. In the charts, "pre-execution checks" indicates processing to check whether there is a SD Memory Card in the CPU Unit and other items. The value of *_Card1VefySta.Active* (Active Flag) changes to TRUE only after all of the pre-execution checks are cleared and the actual verification is started.

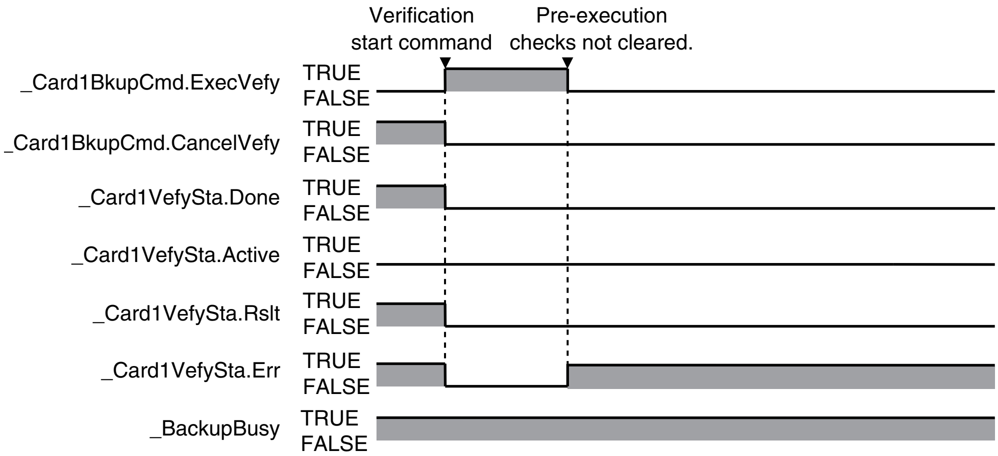
● Normal End with No Differences Found



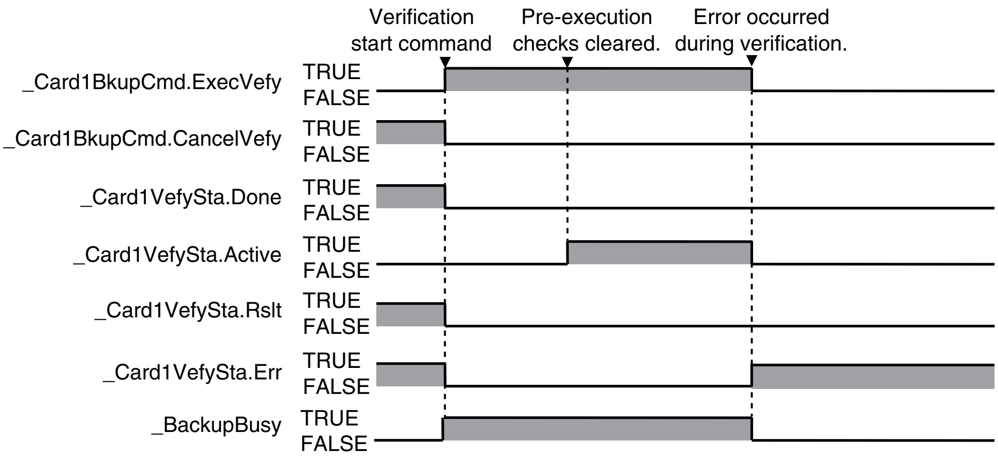
● Normal End with Differences Found



- **Operation When the Verification Cannot Start Because Another Backup Function Is in Progress**

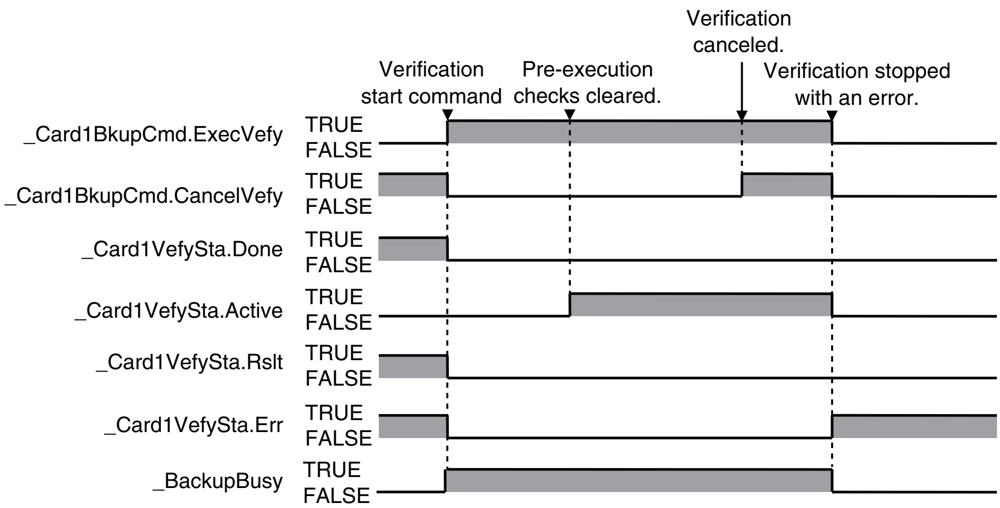


- **Operation When the Verification Fails After a Normal Start**



- **Operation When the Operation Is Canceled While Verification Is in Progress**

The time required to stop the verification operation after it is canceled depends on the progress of the verification operation.



9-3 Disabling Backups to SD Memory Cards

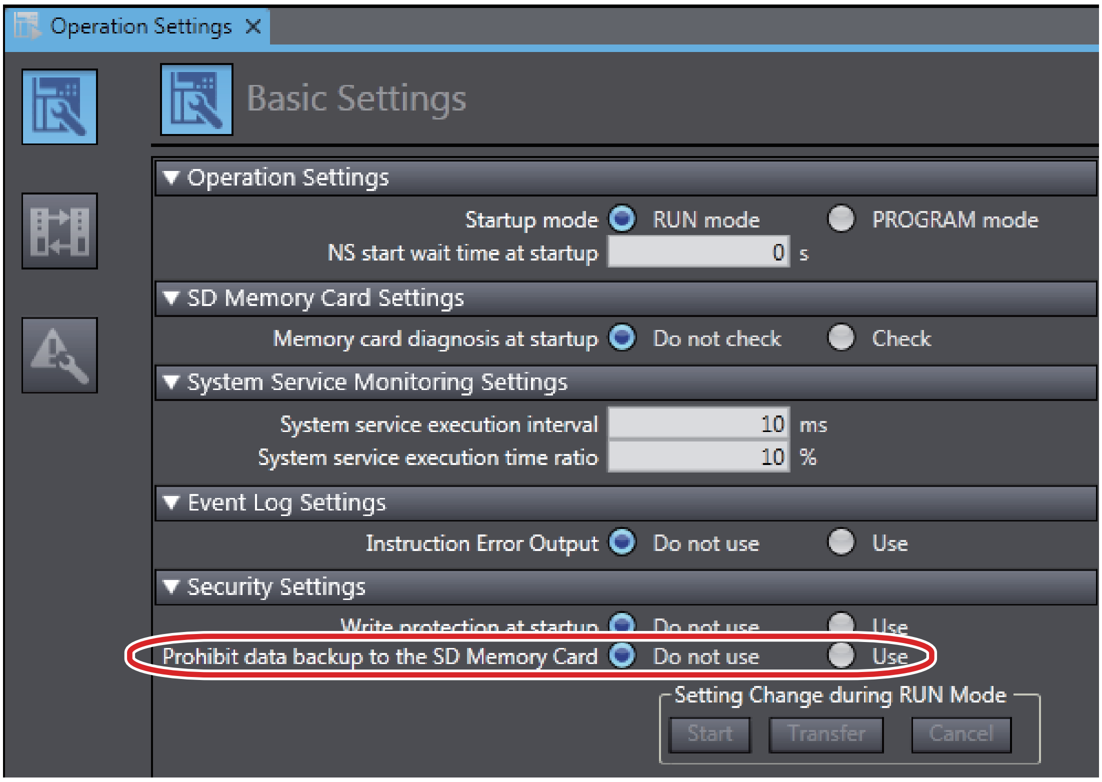
You can disable the backup function from writing data to the SD Memory Card to protect your programming assets.

The following three functions are applicable for disabling backup to SD Memory Card.

- Backups using the CPU Unit front-panel DIP switch
- Backups using system-defined variables
- Backups from the SD Memory Card Window on the Sysmac Studio

Backup function using the BackupToMemoryCard instruction is not applicable. This means that you can backup data using the BackupToMemoryCard instruction even if the *Prohibit data backup to the SD Memory Card* setting is set to be used.

Use the following procedure to set the *Prohibit data backup to the SD Memory Card* setting.
Select the *Use* Option for the **Prohibit data backup to the SD Memory Card** setting in the Basic Settings Display of the Operation Settings Tab Page under **Configurations and Setup** – **Controller Setup** on the Sysmac Studio.

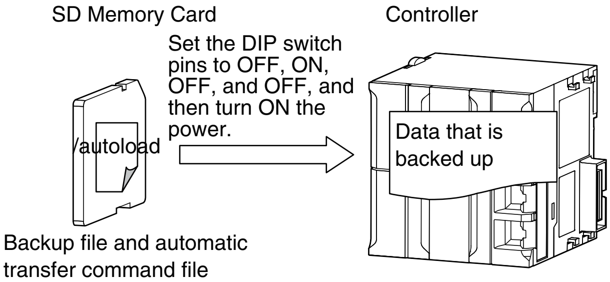


Version Information

A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the BackupToMemoryCard instruction.

9-4 Automatic Transfers from SD Memory Cards

This function automatically transfers the data in a backup file to the Controller when the power supply is turned ON. The backup file must be stored in the */autoload* directory on the SD Memory Card in the CPU Unit. You can use this to operate the CPU Unit with the data in a backup file on the SD Memory Card. The only way to perform this operation is to use the front-panel DIP switch on the CPU Unit.



The automatic transfer uses a backup file that is created with the backup function and an automatic transfer command file. Save both files in the */autoload* directory in advance.

File	Function
Backup file	This file contains the Controller data that is handled by the functions that are related to data backup.
Automatic transfer command file	This file specifies the data groups to transfer when transferring data from a SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer.

The following tables gives the procedure, the applicable directory, and the timing at which the transfer is executed.

Procedure	Directory	Execution timing
CPU Unit front-panel DIP switch	*/autoload* directory on the SD Memory Card	At startup

Processing Contents

- When the power is turned ON, the data in the backup file in the */autoload* directory on the SD Memory Card is automatically transferred to the Controller.
- The automatic transfer function transfers the data in the data groups that are specified in the Autload-Command.ini file in the */autoload* directory. Refer to *9-11-4 Specifications of an Automatic Transfer Command File* for details on the automatic transfer command file.
- If an AutoloadCommand.ini file is not in the */autoload* directory on the SD Memory Card, all of the data from the backup file in the */autoload* directory that can be transferred will be transferred.
- The operating mode that is set in the Startup Mode setting in the Controller Setup is used after completion of the automatic transfer.
- While the data is being automatically transferred, the CPU Unit will be in startup state.
- If an error occurs in the checks that are performed before starting the automatic transfer, the previous data will be retained in the Controller.
- If the power supply to the Controller is interrupted while the data is being automatically transferred, a User Program/Controller Configurations and Setup Transfer Error (a major fault level Controller error) will occur. If that occurs, the data in the Controller is not dependable. Use one of the following methods to clear the error.

  - Perform the automatic transfer again.
  - Clear all of memory and then download the project from the Sysmac Studio.
- All data items that are not specified for the automatic transfer will retain their present values.
- If the present values of variables that are set to be retained (with the Retain attribute) are not set to be transferred, the previous present values of those variables will be retained. However, the values of any variables that do not meet the retain conditions are initialized. These are the retain conditions for the variable:
  - The variable name, data type name, and data type size must be the same before and after transferring the data.
- The SD Memory Card will remain mounted after completion of the automatic transfer operation.
- The write protection for the CPU Unit that is set in the Write Protection at Startup setting is used after completion of the automatic transfer operation.

Procedure

● Transferring Data with the CPU Unit Front-panel DIP Switch

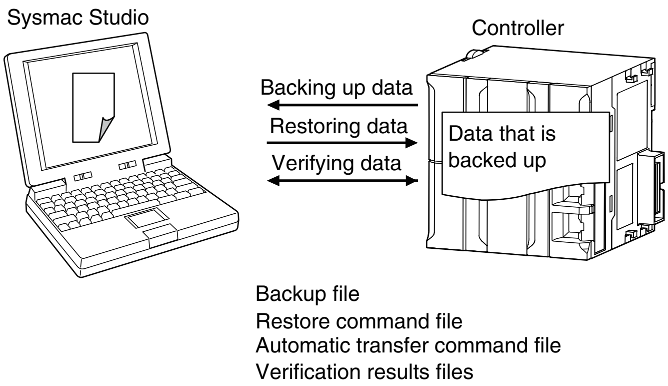
Processing stage	Procedure
Start command	Turn ON the power supply to the Controller with the DIP switch set as follows: 1: OFF, 2: ON, 3: OFF, and 4: OFF.*1
Executing	The transfer is in progress. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The RUN indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly.
Execution results	Normal End: The SD PWR indicator will light. The operating mode that is set in the Startup Mode setting in the Controller Setup is used after completion of the transfer. Error End: The RUN indicator goes out, the ERR indicator lights, and a major fault level Controller error occurs. The SD PWR indicator will light.*2

*1 For the NX-series CPU Unit, set all of pins 5 to 8 on the DIP switch to OFF.

*2 If an SD Memory Card is not inserted, the SD PWR indicator will not light.

9-5 Sysmac Studio Controller Backups

You can use Sysmac Studio to back up, restore, and verify Controller data from a computer.



When you back up data, the backup file, restore command file, and automatic transfer command file are created in the specified directory in the computer. The functions of the backup-related files are given in the following table.

File	Function			
	Contents	**Backing up data**	**Restor-ing data**	**Verifying data**
Backup file	This file contains the Controller data that is handled by the functions that are related to data backup.	Created.	Accessed.	Accessed.
Restore command file	This file specifies the data groups to transfer when restoring data. You can edit this file with a text editor on a computer to specify the data groups to transfer.	Created.	Accessed.	Accessed.
Automatic transfer command file	This file specifies the data groups to transfer when automatically transferring data from a SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer.	Created.	Nothing is done.	Nothing is done.
Verification results files	These files contain the verification results after data is verified.	Nothing is done.	Nothing is done.	Created.

You can execute these functions in the following operating modes.

Processing	Applicable operating modes
Backing up data	RUN mode and PROGRAM mode
Restoring data	PROGRAM mode
Verifying data	RUN mode and PROGRAM mode

Additional Information

You can change the operating mode of the CPU Unit while a backup or verification operation is in progress. However, an error will occur if the backup or verification cannot be processed normally due to faulty memory in the CPU Unit, or some other failure.

9-5-1 Backup (Controller to Computer)

The Controller data is saved in the specified directory on the computer.

Processing Contents

- For the Units and slaves settings in the backup data, you must select all EtherCAT slaves that are connected and all of the CJ-series Units that are connected.
- When you back up data, the backup file, restore command file, and automatic transfer command file are created in the specified directory in the computer.
- If the backup-related files are already in the specified directory, they are overwritten.
- If an error occurs while writing the backup-related files to specified directory, the previous backup-related files will be deleted. Also, the new backup-related files will not be created.
- If an error occurs before the new backup-related files are created, the previous files are retained and the new files are not created.
- The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will be TRUE during the backup operation.

Note You can select CJ-series Units only for NJ-series CPU Units.

Procedure

1. Select ***Backup*** – ***Backup Controller*** from the Tools Menu on the Sysmac Studio.
2. Specify the folder in which to save the backup file, restore command file, and automatic transfer command file.
3. Click the **Execute** Button on the Backup Confirmation Dialog Box.

   The data is backed up and the backup file, restore command file, and automatic transfer command file are created.

9-5-2 Restore (Computer to Controller)

The data in a backup file in the specified directory on the computer is transferred to the Controller. This operation can only be performed in PROGRAM mode.

Processing Contents

- The data in a backup file in the specified directory on the computer is transferred to the Controller.
- You can select the data groups to restore from the Sysmac Studio. The conditions for restoring the data are given in the following table.

Data group	Restoring condition
User program and settings	The CPU Unit must be selected.
Present values of variables	The present values of variables that are specified for retention with the Retain attribute must be selected.
Present values of memory used for CJ-series Units	The present values of memory used for CJ-series Units that are specified for retention with the Retain attribute must be selected.*1
Units and slaves settings	The CJ-series Units and EtherCAT slaves must be selected.*1
Absolute encoder home offsets	The absolute encoder home offsets must be selected.

*1 You can select CJ-series Units only for NJ-series CPU Units.

- If an error occurs in the checks that are performed before starting to restore the data, the previous data will be retained in the Controller.
- If the power supply to the Controller is interrupted while the data is being restored, a User Program/Controller Configurations and Setup Transfer Error (a major fault level Controller error) will occur. If that occurs, the data in the Controller is not dependable. Use one of the following methods to clear the error.
  - Perform the restore operation again.
  - Clear all of memory and then download the project from the Sysmac Studio.
- If the present values of variables that are set to be retained (with the Retain attribute) are not set to be restored, the previous present values of those variables will be retained. However, the values of any variables that do not meet the retain conditions are initialized. These are the retain conditions for the variable:
  - The variable name, data type name, and data type size must be the same before and after restoring the data.
- Cycle the power supply to all of the EtherCAT slaves after you restore data.

Procedure

1 Select ***Backup*** - ***Restore Controller*** from the Tools Menu on the Sysmac Studio.

2 Specify the folder that contains the backup file and restore command file.

3 Click the **Execute** Button on the Restoration Confirmation Dialog Box.

The restoration operation is executed.

9-5-3 Verify (between Controller and Computer)

The Controller data and the data in a backup file in the specified directory on the computer are compared.

Processing Contents

- The Controller data and the data in a backup file in the specified directory on the computer are compared. You can select the data groups to verify from the Sysmac Studio. The conditions for verifying the data are given in the following table. If you specify all data, all of the following data will be compared.

Data group	Verification condition
User program and settings	The CPU Unit must be selected.
Units and slaves settings	The CJ-series Units and EtherCAT slaves must be selected.*1

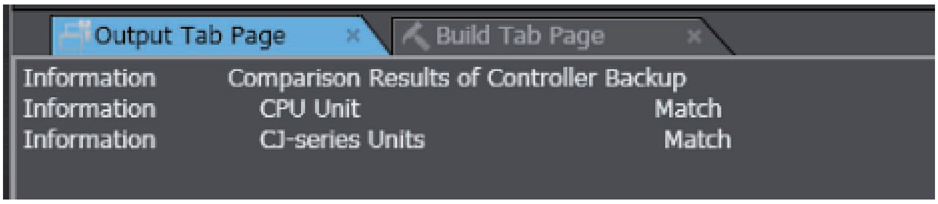
*1 You can select CJ-series Units only for NJ-series CPU Units.

- The results of the verification are displayed in the dialog box on the Sysmac Studio.
- The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will be TRUE during the backup operation.

Procedure

1. Select ***Backup*** - ***Compare with Backup File*** from the Tools Menu on the Sysmac Studio.
2. Specify the folder that contains the backup file.
3. Click the **Execute** Button on the Comparison Confirmation Dialog Box.

   The data is compared and the verification results files are created in the folder that contains the backup file. The comparison results are also displayed in the Output Tab Page.



9-6 Importing and Exporting Sysmac Studio Backup File Data

You can create or read from a backup file in the specified directory on the computer from the Sysmac Studio project without using the Controller. This following data is processed:

Function		Data group				
		User program and settings	Present values of variables	Present values of memory used for CJ-series Units*1	Units and slaves settings	Absolute encoder home offsets
Importing and exporting Sysmac Studio backup file data	Exporting backup file data	OK*2	OK	OK	×	×
	Importing backup file data	OK	OK	OK	OK	×

*1 You can use memory for CJ-series Units only with NJ-series CPU Units.

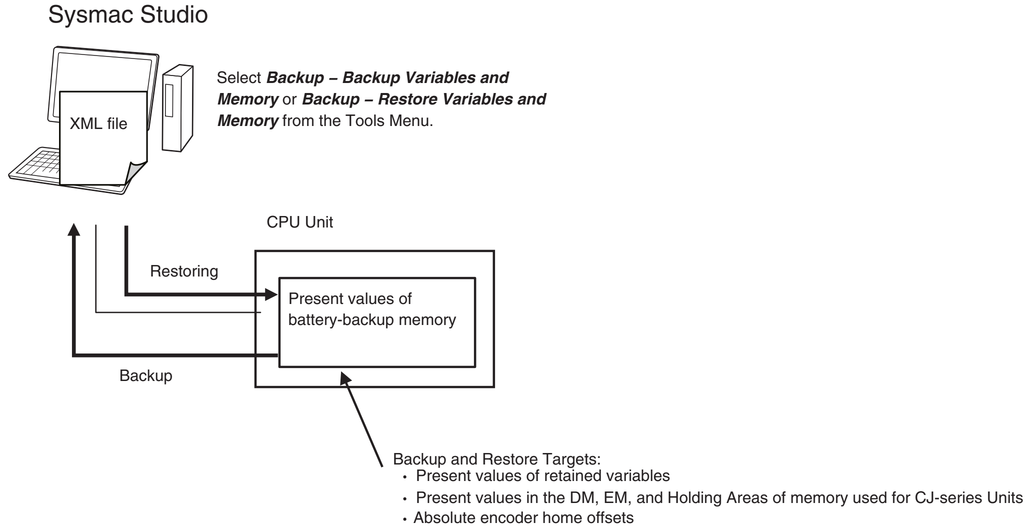
*2 The following data is not processed:
Controller names: the built-in EtherNet/IP port name; Controller Setup: the tag data link settings for the built-in EtherNet/IP port; Unit Configuration and Unit Settings: the CPU Bus Unit areas, operation authority verification, and Data Trace Settings.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for more information on these functions.

9-7 Sysmac Studio Variable and Memory Backup Functions

You can back up the present values of the battery-backup memory in the CPU Unit to an XML file on your computer or restore the battery-backup memory from a previously saved backup file.

This section describes the applicable data, operating procedures, and CPU Unit model compatibility for the Sysmac Studio variable and memory backup functions.



9-7-1 Applicable Data for Sysmac Studio Variable and Memory Backup Functions

Applicable Data for Sysmac Studio Variable and Memory Backup Functions

- Present values of variables with a Retain attribute
- Present values in the DM, EM, and Holding Areas of memory used for CJ-series Units
- Absolute encoder home offsets

Note You can use memory for CJ-series Units only with NJ-series CPU Units.

Version Information

With a CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher, you can select specific variables to back up or restore the present values of variables with a Retain attribute.

9-7-2 Using Sysmac Studio Variable and Memory Backup Functions

The Sysmac Studio procedure is as follows:
Place the Sysmac Studio online with the CPU Unit, and select either ***Backup – Backup Variables and Memory*** or ***Backup – Restore Variables and Memory*** from the Tools Menu.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details.

9-7-3 Compatibility between CPU Unit Models

With the Sysmac Studio variable and memory backup functions, you can restore the data even if the models of the CPU Units for backing up and restoring data are different.

Version Information

The models of the CPU Units for backing up and restoring data can be different only when using a CPU Unit with version 1.04 or later and Sysmac Studio version 1.05 or higher. The compatibility for all other cases is given in the following table.

(C: Compatible, N: Not compatible.)

CPU Unit model where data was backed up	CPU Unit model to restore to	
	NJ501-1500 NJ501-1400 NJ501-1300	NJ301-1200 NJ301-1100
NJ501-1500, NJ501-1400, or NJ501-1300	C	N
NJ301-1200 or NJ301-1100	N	C

The following precautions are required for the data that is being backed up and restored.

Present Values of Variables with a Retain Attribute

There are no precautions for the present values of variables with a Retain attribute. You can restore the data with no problems regardless of the models of the backup and restore CPU Units.

Present Values in the DM, EM, and Holding Areas of Memory Used for CJ-series Units

The following precautions are necessary for the present values in the DM, EM, and Holding Areas of memory used for CJ-series Units if the models of the backup and restore CPU Units are different.

CPU Unit model where data was backed up	CPU Unit model to restore to	Precaution
NJ501-1300, NJ501-1400, or NJ501-1500	NJ301-1200, NJ301-1100, NJ101-1000, or NJ101-9000	For EM Area data, only data for banks E0 to E3 in the backup file is restored. The data for banks E4 to E18 in the backup file is ignored.
NJ301-1200, NJ301-1100, NJ101-1000, or NJ101-9000	NJ501-1300, NJ501-1400, or NJ501-1500	For EM Area data, only data for banks E0 to E3 in the backup file are restored. Banks E4 to E18 for the CJ-series Units retain their previous values.

Absolute Encoder Home Offsets

The following precautions are necessary for the absolute encoder home offsets if the models of the backup and restore CPU Units are different.

CPU Unit model where data was backed up	CPU Unit model to restore to	Precaution
NJ501-1300, NJ501-1400, or NJ501-1500	NJ301-1200, NJ301-1100, or NJ101-1000	Regardless of the number of enabled axes in the backup CPU Unit, the data for all axis in the backup file is restored in order for the number of enabled axes in the restore CPU Unit. Any remaining data in the backup file is ignored.
NJ301-1200, NJ301-1100, or NJ101-1000	NJ501-1300, NJ501-1400, or NJ501-1500	Regardless of the number of enabled axes in the backup CPU Unit, the data for all axis in the backup file is restored in order for the number of enabled axes in the restore CPU Unit. If the number of enabled axes in the restore CPU Unit exceeds the number of enabled axes for which there is data in the backup file, the remaining data in the restore CPU Unit retains the previous values.

9-8 Backup Functions When EtherCAT Slaves Are Connected

For EtherCAT slaves, you can use the SD Memory Card backup functions, the Sysmac Studio Controller backup functions, and Sysmac Studio backup import function.

This section provides precautions for connected EtherCAT slaves for the data that is backed up, backup support according to Controller status, restore conditions, and specific models of EtherCAT slaves.

Additional Information

To use the backup functions for EtherCAT Slave Terminals, refer to *9-9 Backup Functions When EtherCAT Slave Terminals Are Connected*.

9-8-1 Backed Up EtherCAT Slave Data

The data that is backed up for EtherCAT slaves is given in the following table.

Setting	Data that is backed up
EtherCAT Master Settings	The following data is backed up: Model name, Product name, Number of Slaves, PDO Communications Cycle, Fail-soft Operation Setting, Wait Time for Slave Startup, PDO communications timeout detection count, Revision Check Method, and Serial Number Check Method.
EtherCAT Slaves Settings	The following data is backed up: Device name, model name, product name, revision, node address, enabled/disabled settings, serial number, PDO map settings, enable distributed clock, reference clock, and setting parameter settings.

9-8-2 Backup Support Depending on the Controller Status

The following table shows when backup, restore, and verify operations can be performed for EtherCAT slaves based on the Controller status.

Controller status		Execution		
		Backing up data	Restoring data	Verifying data
Link OFF		Not possible.*1	Not possible.*2	Possible.*3
Illegal master status*4		Not possible.*1	Not possible.*2	Possible.*3
Network configuration mismatch with configuration information*5		Not possible.*1	Not possible.*2	Possible.3
Network configuration mismatch with configuration at time of backup		Possible.	Not possible.*2	Possible.*3
Disabled slave in network configuration	Disabled slaves in actual configuration	Possible. *6	Possible. *6	Possible. *6
	No disabled slaves in actual configuration	Possible.	Possible.	Possible.

Controller status		Execution		
		Backing up data	Restoring data	Verifying data
Slave disconnected for “Disconnect” designation in network configuration	Disconnected slaves in actual configuration	Not possible.*1	Possible, but data for disconnected slaves is not restored.	Possible, but data for disconnected slaves is not restored.
	No disconnected slaves in actual configuration	Not possible.*1	Not possible.*2	Possible.*3
Slave Initialization Error		Not possible.*1	Not possible.*2	Possible.*3

*1 EtherCAT Slave Backup Failed events are recorded in the event log.

*2 EtherCAT Slave Restore Operation Failed events are recorded in the event log.

*3 The verification results will show differences.

*4 This refers to the following errors: Duplicate Slave Node Address, Network Configuration Information Error, Network Configuration Error, Slave Initialization Error, Network Configuration Verification Error for Operation Setting of “Stop”, and Link OFF Error.

*5 This refers to the following errors: Network configuration mismatch with configuration when the backup was performed (incorrect connection ports for slaves on branched networks are treated as a mismatch) and network configuration information mismatch with actual network configuration (incorrect connection ports for slaves on branched network are treated as a match).

*6 For a CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher, data for disabled slaves is also covered by the backup functions. Data for disabled slaves is not backed up for other versions.

9-8-3 Conditions for Restoring EtherCAT Slave Data

The following conditions must be met before you can restore the backup data to the EtherCAT slaves.

- The backup files must contain the EtherCAT slave data.
- The Network Configuration Information must match the actual network configuration where data is being restored.
- The revision values that are preset in the EtherCAT slaves must match. The conditions used to evaluate the match are based on the Revision Check Method in the backup file. Even if you set the Revision Check Method to not check revisions, the restoration operation cannot be performed if the set revision is greater than the actual revision of the slave. You cannot change the revision values.
- The serial numbers must match if the Serial Number Verification setting in the backup file is set to verify the serial numbers.
- The node addresses must match if the hardware switches are used to set the node address.

Precautions for Correct Use

- Cycle the power supply to all of the EtherCAT slaves after you restore data.
- All slaves are disconnected after the data is restored. You must connect the target slaves again to reset the disconnected slaves.
- If you set the Serial Number Verification setting in the backup file to verify the serial numbers, the data cannot be restored if you replace any of the hardware for the EtherCAT slaves. In this case, change the network configuration in Sysmac Studio and download the configuration data to the new slaves. Then, transfer the slave parameters to restore the slaves to their original condition. If the node address is set on the hardware switches, use the same setting as when the data was backed up.

9-8-4 EtherCAT Slaves for Which You Can Back Up Data

You can back up data for the following EtherCAT slaves. Observe the precautions.

EtherCAT slaves	Precautions
NX-ECC NX-series EtherCAT Coupler Unit	You cannot back up, restore, or compare data for Safety Control Units on EtherCAT Slave Terminals. Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for information on importing and exporting settings for a Safety Control Unit.
R88D-KN□□□-ECT AC Rotary Servo Drives	*1*2
R88D-KN□□□-ECT-L AC Linear Servo Drives	*1*2
3G3AX-MX2-ECT and 3G3AX-RX-ECT Inverters	Data is sometimes not restored due to Inverter restrictions. If an EtherCAT Slave Restore Operation Failed event (event code 1030 0000 hex) occurs when you try to restore the data, use the procedure that is given below in *Procedure When Restoring Data Fails for an 3G3AX-MX2-ECT or 3G3AX-RX-ECT Inverter* on page 9-38 to transfer the parameters from the Sysmac Studio to the Inverter. Note that even if the restore operation for the Inverter fails, all other data are restored including settings of Units and slaves, user program settings, and present values of variables.
FH-3□□□□ and FH-1□□□□ Vision Sensors	The setup data for these Vision Sensors (such as the scene data and system data) is not backed up, restored, or verified. To transfer the setup data to an external file or to the Vision Sensor, select ***Sensor data*** *–* ***Save to file*** or ***Sensor data*** *–* ***Load from file*** from the Tools Menu on the editing tab page for the Configurations and Setup of the Sysmac Studio. Refer to the *Vision System FH/FZ5 series User's Manual* (Cat. No. Z340) for details.
FQ-M□□□□-ECT and FQ-M□□□□-M-ECT Vision Sensors	The setup data for these Vision Sensors (such as the scene data and system data) is not backed up, restored, or verified. To transfer the setup data to an external file or to the Vision Sensor, select ***Sensor data*** *–* ***Save to file*** or ***Sensor data*** *–* ***Load from file*** from the Tools Menu on the editing tab page for the Configurations and Setup of the Sysmac Studio. For details, refer to the *FQ-M-series Specialized Vision Sensor for Positioning User's Manual* (Cat. No. Z314).
FZM1-□□□-ECT Vision Sensors	The setup data for these Vision Sensors (such as the scene data and system data) is not backed up, restored, or verified. To save the setup data for the Vision Sensor to a USB memory device or to write it to the Controller, use the software tool for the Vision Sensor. Refer to the *FZ3 Series Vision Sensor User*'s *Manual* (Cat. No. Z290) for details.
GX-□D16□□, GX-□D32□8, and GX-OC1601 Digital I/O Terminals	*1
GX-AD0471 and GX-DA0271 Analog I/O Terminals	*1
GX-EC0211 and GX-EC0241 Encoder Input Terminals	---
GX-JC0□ EtherCAT Junction Slaves	There is no internal data that needs to be backed up.
ZW-CE1□T Confocal Fiber Type Displacement Sensors	None of the settings are backed up, restored, or verified. Refer to the *Displacement Sensor ZW Series Confocal Fiber Type Displacement Sensor Users Manual* (Cat. No. Z332) for information on saving the settings and loading them to the Controller.
E3NW-ECT and E3X-ECT Digital Sensors	The parameters in the Sensor are not backed up, restored, or verified.

EtherCAT slaves	Precautions
Slaves from other manufacturers	• Data is backed up, restored, and verified only when it is correctly defined in the ESI. To back up, restore, or verify data that is not defined in the ESI, use the software tool for the slave. • If backing up, restoring, or verifying data fails, contact the manufacturer of the slave for the appropriate procedures.

*1 Cycle the power supply to a slave after you restore data. Cycle the power supply to a slave before you verify the data after you restore it. The verification will fail if you do not cycle the power supply before you perform the verification.

*2 If any of the following conditions applies, do not turn the Servo ON while the data is being backed up or restored before you verify the data. If you turn the Servo ON while the data is being backed up or restored before you verify the data, the parameters are updated before the verification operation and may cause differences in the verification results.

- When the Realtime Autotuning Mode Selection (3002 hex) is set to 1 to 4, or 6 (enabled).
- When the Adaptive Filter Selection (3200 hex) is set to 1 or 2 (enabled).

Procedure When Restoring Data Fails for an 3G3AX-MX2-ECT or 3G3AX-RX-ECT Inverter

If you restore data when a 3G3AX-MX2-ECT or 3G3AX-RX-ECT Inverter is connected, an EtherCAT Slave Restore Operation Failed event (event code 10300000 hex) will occur. Use the follow procedure from the Sysmac Studio to write the backup parameters to the Inverter. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details.

1 Import the Inverter parameters from the backup file with the backup file import function of the Sysmac Studio.

Display the Inverter parameters on the Inverter Parameters Tab Page for the Controller Configurations and Setup of the Sysmac Studio.

2 Confirm that the model number of the Inverter in the parameters that you imported agrees with the model number of the Inverter that is actually connected.

3 Download the parameters to the Inverter using the “To Drive” menu on the Inverter Parameters Tab Page for the Controller Configurations and Setup of the Sysmac Studio.

Precautions for Correct Use

If you use the Inverter Mode Selection parameter (parameter number b171) in a 3G3AX-MX2-ECT Inverter, change the Inverter to the mode that was used when the backup data was created before you write the parameters. After you change the mode setting, you must initialize the Inverter to enable the change.

Additional Information

Even if the restore operation for the Inverter fails, all other data are restored including settings of Units and slaves, user program settings, and present values of variables.

9-9 Backup Functions When EtherCAT Slave Terminals Are Connected

For EtherCAT Slave Terminals, you can use the SD Memory Card backup functions, the Sysmac Studio Controller backup functions, and Sysmac Studio backup import function.

This section provides information on the data that is backed up, backup support according to Controller status, and restore conditions when EtherCAT Slave Terminals are connected.

Precautions for Correct Use

You cannot back up, restore, or compare data for Safety Control Units on EtherCAT Slave Terminals. Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for information on importing and exporting settings for a Safety Control Unit.

Version Information

A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use EtherCAT Slave Terminals.

9-9-1 Backing Up Data in an EtherCAT Slave Terminal

The data that can be backed up for an EtherCAT Slave Terminal is different for the EtherCAT Coupler Unit and the NX Units. The data that is backed up is given in the following table.

(OK: Applicable, NA: Not applicable)

Unit	Data	Backup	Restore	Compare
EtherCAT Coupler Unit	Configuration information*1	OK	OK	OK
	Unit operation settings	OK	OK	OK
NX Units	Configuration information*1	OK	OK	OK
	Unit operation settings	OK	OK	OK
	Unit application data*2	OK	OK	OK

*1 The configuration information includes the Unit configuration information and I/O allocation information.

*2 This is the specific data for each NX Unit. Some NX Units do not have Unit application data.

Precautions for Correct Use

To restore backup data to an EtherCAT Slave Terminal that has an identical Unit configuration to the EtherCAT Slave Terminal from which data was backed up, make sure that all hardware switches are set to the same settings as when the backup was made. Backup data cannot be restored if the hardware switches are set differently from those in the backup data. This will cause a Restore Operation Failed to Start (EtherCAT Slave) observation event to occur.

9-9-2 Backup Support Depending on the EtherCAT Slave Terminal Status

The following table shows when backup, restore, and compare operations can be performed for EtherCAT Slave Terminals based on the EtherCAT Slave Terminal status.

EtherCAT Slave Terminal status	Execution		
	Backup	Restore	Compare
Automatic creation of the Unit configuration information	Possible.*1	Possible.*2	Possible.
Waiting for NX Unit participation	Not possible.*3	Not possible.*4	Possible.*5
Watchdog time error in EtherCAT Coupler Unit or NX Unit	Not possible.*3	Not possible.*4	Possible.*5
During Bus Controller Error	Not possible.*3	Not possible.*4	Possible.*5
During Unit Configuration Information Error	Not possible.*3	Possible.	Possible.*5
During Unit Configuration Verification Error	Not possible.*3	Possible.	Possible.*5
The Unit Configuration does not agree with the Unit Configuration information in the backup data.	---	Not possible.*4	Possible.*5

*1 The backup contains information saying that the Unit configuration information does not exist.

*2 After the data is restored, automatic Unit configuration status continues.

*3 A Backup Failed event is recorded in the event log.

*4 A Restore Operation Failed event is recorded in the event log.

*5 The verification results will show differences.

9-9-3 Conditions for Restoring EtherCAT Slave Terminal Data

The following conditions must be met before you restore the backup data to the EtherCAT Slave Terminals.

- The backup files must contain the data for the EtherCAT Coupler Unit and NX Unit.
- The original Unit Configuration in the backup must match the actual Unit configuration where data is being restored.
- The serial number of the EtherCAT Coupler Unit from which the data was backed up and the serial number of the EtherCAT Coupler Unit to which the data is restored must be the same. However, this assumes that the setting of the Serial Number Check Method in the Unit operation settings of the Communications Coupler Unit in the backup file is set to *Setting = Actual device.*
- The serial numbers of the NX Units from which the data was backed up and the serial numbers of the NX Units to which the data is restored must be the same. However, this assumes that the setting of the Serial Number Check Method in the Unit operation settings of the Communications Coupler Unit in the backup file is set to *Setting = Actual device.*
- The hardware switch settings of the EtherCAT Coupler Unit from which the data was backed and the hardware switch settings of the EtherCAT Coupler Unit to which the data is restored must be the same.
- The unit version setting of the EtherCAT Coupler Unit from which the data was backed up and the unit version of the actual EtherCAT Coupler Unit to which the data is restored must be the same.
- The unit version settings of the NX Unit from which the data was backed up and the unit versions of the actual NX Units to which the data is restored must be the same.

9-10 Backup Functions When CJ-series Units Are Connected

Data in CJ-series Units is covered by the SD Memory Card backup functions and Sysmac Studio Controller backup functions. This section provides precautions for connected CJ-series Units for the data that is backed up, backup support according to Controller status, and restore conditions.

Precautions for Correct Use

You can connect CJ-series Units only with NJ-series CPU Units.

9-10-1 Backed Up CJ-series Unit Data

The present values in memory used for CJ-series Units and the parameters in the CJ-series Units are backed up. Some of this data is in the CJ-series Unit and some are in the CPU Unit. You do not need to be aware of where the data is located because the backup, restoration, and verification operations will automatically process this data.

9-10-2 Backup Support Depending on the Controller Status

The following table shows when backup, restore, and verify operations can be performed for CJ-series Units based on the Controller status.

Controller status	Execution		
	Backing up data	Restoring data	Verifying data
I/O Bus Check Error End Cover Missing Incorrect Unit/Expansion Rack Connection Duplicate Unit Number Error Too Many I/O Points I/O Setting Check Error	Not possible.*1	Not possible.*2	Possible.*3
Restarting the CJ-series Unit	Not possible.*1	Possible.	Possible.*3

*1 CJ-series Unit Backup Failed events are recorded in the event log.

*2 CJ-series Unit Restore Operation Failed events are recorded in the event log.

*3 The verification results will show differences.

9-10-3 Conditions for Restoring CJ-series Unit Data

The following conditions must be met before you can restore the backup data to the CJ-series Units.

- The backup files must contain the CJ-series Unit data.
- The Unit Configuration in the backup file must match the actual Unit Configuration where data is being restored.
- Each CJ-series Unit must meet the conditions for that Unit. (Refer to the manuals for the CJ-series Units for the specific conditions for each Unit.)

Precautions for Correct Use

If you restore data using the SD Memory Card backup functions or the Sysmac Studio Controller backup functions while CJ-series Units are connected, a CPU Unit Service Monitor Error will occur. This means that servicing the CJ-series Units from the CPU Unit was not completed within a specific amount of time. However, this is the result of the time that is required to restore the data and it does not indicate an error. The following will occur at this time.

- For communications-related CJ-series Units, the MS indicator flashes red.
- For CJ-series Units with seven-segment indicator, the indicator displays “HE”.
- For CJ-series Units that have an ERH indicator, the ERH indicator lights.
- An event code of 00000002 hex is recorded in the Controller event logs to indicate a CPU Unit Service Monitor Error or Refresh Timeout event.
- If a CJ-series CJ1W-CT021 High-speed Counter Unit is connected, an event code of 68010000 hex is recorded in the Controller event logs to indicate a Unit Error. The attached information will be 0002 hex.

9-11 Backup-related Files

This section describes the specifications of the backup-related files. These backup-related files apply to all backup functions except for the Sysmac Studio variable and memory backup functions.

9-11-1 Types of Backup-related Files

There are four types of files that are related to backup functions: backup files, restore command files, automatic transfer command files, and verification results files.

● Backup File

This file contains the Controller data that is handled by the backup-related functions. Backup files are created when data is backed up.

● Restore Command File

This file specifies the data groups to transfer by restoring data from a SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer. These files are created when data is backed up.

● Automatic Transfer Command File

This file specifies the data groups to transfer by automatically transferring data from a SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer. These files are created when data is backed up.

● Verification Results Files

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit.

There are four different verification results files, as described below. These files are created when you perform a verification using the SD Memory Card backup function.

Verification results files	Description
Controller verification results file	This file contains the verification results for all backup data specified by the restore command file.
EtherCAT slave verification results file	This file contains the verification results for each EtherCAT slave. It is created when the Unit and slave settings are set to be restored in the restore command file and the EtherCAT slave settings are contained in the backup file.
EtherCAT Slave Terminal verification results file	This file contains the verification results for each EtherCAT Coupler Unit and NX Unit. This file is created when all of the following conditions are met. • The Unit and slave settings are specified for restoration in the restore command file. • The EtherCAT slave settings are included in the backup file. • One or more EtherCAT Slave Terminals is connected. If an EtherCAT Slave Terminal verification results file is created, an EtherCAT slave verification results file is always created at the same time.
CJ-series Unit verification results file*1	This file contains the verification results for each CJ-series Unit. It is created when the Unit and slave settings are set to be restored in the restore command file and the CJ-series Unit settings are contained in the backup file.

*1 You can use CJ-series Units only with NJ-series CPU Units.

9-11-2 Specifications of a Backup File

This section describes the file name, creation timing, and created directory for a backup file.

File Name

A different backup file name is given depending on the CPU Unit series.

File	CPU Units	File name
Backup file	NX-series CPU Units	NXBackup.dat
	NJ-series CPU Units	NJBackup.dat

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	CPU Unit front-panel DIP switch	When backup is executed	Root directory on the SD Memory Card
	System-defined variables	When backup is executed	Directory on the SD Memory Card that you specified with the system-defined variable
	SD Memory Card Window in Sysmac Studio	When backup is executed	Directory on the SD Memory Card that you specified with the Sysmac Studio
	Special instruction*1	When backup is executed	The directory on the SD Memory Card that you specified for the input variable of the BackupToMemoryCard instruction
Sysmac Studio Controller backups	Sysmac Studio Controller Backup Dialog Box	When backup is executed	Directory in the computer that you specified with the Sysmac Studio
Importing and exporting Sysmac Studio backup file data	Sysmac Studio Backup File Export Dialog Box	When data is exported	Directory in the computer that you specified with the Sysmac Studio

*1 A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required.

9-11-3 Specifications of a Restore Command File

This section describes the file name, creation timing, created directory, and data group specification method for a restore command file.

File Name

File	File name
Restore command file	RestoreCommand.ini

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	CPU Unit front-panel DIP switch	When backup is executed	Same directory as backup file
	System-defined variables	When backup is executed	Same directory as backup file
	SD Memory Card Window in Sysmac Studio	When backup is executed	Same directory as backup file
Sysmac Studio Controller backups	Sysmac Studio Controller Backup Dialog Box	When backup is executed	Same directory as backup file
Importing and exporting Sysmac Studio backup file data	Sysmac Studio Backup File Export Dialog Box	When data is exported	Same directory as backup file

Specifying the Data Groups to Restore

The restore command file allows you to specify the data groups to restore. You can change the data group specifications by editing the file with a text editor on a computer.

For example, if you change “Variable=yes” on line 8 in the file contents that are shown in the following table to “Variable=no,” the present values of variables will not be restored.

File contents (defaults when the file is created)	Description
[Restore]	
; --- User Program and Configuration. --- ; Always select "yes". UserProgram=yes	User program and settings This data group is always restored. Always select
; --- Present values of variables (Retained variables only). --- ; "yes":will be restored, "no":will not be restored Variable=yes	Present values of variables (only variables that are set to be retained with the Retain attribute) yes/no: Restore/Do not restore.
; --- Present values of memory used for CJ-series Units (Holding, EM, and DM Area data). --- ; "yes":will be restored, "no":will not be restored Memory=yes	Present values of memory used for CJ-series Units (only addresses that are set to be retained with the Retain attribute) yes/no: Restore/Do not restore.
:---Unit/Slave Parameters.--- ; "yes";will be restored."no";will not be restored UnitConfig=yes	Units and slaves settings yes/no: Restore/Do not restore.
; --- Absolute encoder home offset. --- ; "yes":will be restored, "no":will not be restored AbsEncoder=yes	Absolute encoder home offset yes/no: Restore/Do not restore.

Note 1 The default file contents when the restore command file is created are given above. All of the data groups that are listed in the file are set to be restored.

2 The restore command file lists the restorable data groups that were in the backup file when the backup file was created.

3 Only single-byte alphanumeric characters are used. The text is not case sensitive.

Precautions for Correct Use

When you edit the restore command file, do not change anything in the file except for the “yes” and “no” specifications for the selectable data groups. If you change anything else in the file, the Controller may perform unexpected operation when you restore the data.

9-11-4 Specifications of an Automatic Transfer Command File

This section describes the file name, creation timing, created directory, and data group specification method for an automatic transfer command file.

File Name

File	File name
Automatic transfer command file	AutoloadCommand.ini

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	CPU Unit front-panel DIP switch	When backup is executed	Same directory as backup file
	System-defined variables	When backup is executed	Same directory as backup file
	SD Memory Card Window in Sysmac Studio	When backup is executed	Same directory as backup file
Sysmac Studio Controller backups	Sysmac Studio Controller Backup Dialog Box	When backup is executed	Same directory as backup file
Importing and exporting Sysmac Studio backup file data	Sysmac Studio Backup File Export Dialog Box	When data is exported	Same directory as backup file

Specifying the Data Groups to Automatically Transfer

The automatic transfer command file allows you to specify the data groups to transfer automatically. You can change the data group specifications by editing the file with a text editor on a computer. For example, if you change “Variable=yes” on line 8 in the file contents that are shown in the following table to “Variable=no,” the present values of variables will not be automatically transferred.

File contents (defaults when the file is created)	Description
[Autoload]	
; --- User Program and Configuration. --- ; Always select "yes". UserProgram=yes	User program and settings This data group is always transferred. Always select
; --- Present values of variables (Retained variables only). --- ; "yes":will be transferred, "no":will not be transferred Variable=yes	Present values of variables (only variables that are set to be retained with the Retain attribute) yes/no: Transfer/Do not transfer.
; --- Present values of memory used for CJ-series Units (Holding, EM, and DM Area data). --- ; "yes":will be transferred, "no":will not be transferred Memory=yes	Present values of memory used for CJ-series Units (only addresses that are set to be retained with the Retain attribute) yes/no: Transfer/Do not transfer.

Note 1 The default file contents when the automatic transfer command file is created are given above. All of the data groups that are listed in the file are set to be automatically transferred.

2 The automatic transfer command file lists the transferable data groups that were in the backup file when the backup file was created.

3 Only single-byte alphanumeric characters are used. The text is not case sensitive.

Precautions for Correct Use

When you edit the automatic transfer command file, do not change anything in the file except for the “yes” and “no” specifications for the selectable data groups. If you change anything else in the file, the Controller may perform unexpected operation when you automatically transfer the data.

9-11-5 Specifications of a Controller Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for a Controller verification results file.

File name

File	File name
Controller Verification Results File	VerifyResult.log

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group. You can check the verification results in the portion that gives the verification results for each data group. “Result=Matched” indicates a data group for which no differences were found. “Result=Not matched” indicates a data group for which differences were found. In the file shown below, the user program and configuration data matched, and the Units and slave parameters did not match.

File contents	Description
[UserProgram]	
; --- User Program and Configuration. ---	User program and settings
Result=Matched	Matched: No differences were found, Not matched: Differences were found.
[UnitConfig]	
; --- Unit/Slave Parameters. ---	Units and slaves settings
Result=Not matched	Matched: No differences were found, Not matched: Differences were found.

Note The verification results are given only for the data groups that were compared.

9-11-6 Specifications of an EtherCAT Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for an EtherCAT verification results file.

File Name

File	File name
EtherCAT Verification Results File	VerifyResult_ECAT.log

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group. You can check the verification results in the portion that gives the verification results for each EtherCAT slave. “Result=Matched” indicates a data group for which no differences were found. “Result=Not matched” indicates a data group for which differences were found.

The following table gives an example of the verification results for the following file contents.

- Matched: EtherCAT slave called Master and EtherCAT Slave Terminal E022
- Not matched: EtherCAT slave E001

File contents	Description
[Verification Results] ; --- EtherCAT Parameters. --- : --- See the VerifyResult_ECAT_NX.log about detail result if NX mark is included in square brackets. [Master] Result=Matched [E001] Result=Not matched Factor=Verification error [E002:NX] Result=Matched	The slaves are indicated with the user-set device names. For an EtherCAT Slave Terminal, “:NX” is added to the end of the device name.*1 The verification results are given as follows: Result=Matched Same Result=Not matched Different

* If EtherCAT Slave Terminals are set for verification, the EtherCAT Slave Terminal verification results file is created. The detailed verification results for the EtherCAT Slave Terminals are given in the EtherCAT Slave Terminal verification results file.

Note The verification results are given only for the EtherCAT slaves that were compared.

9-11-7 Specifications of an EtherCAT Slave Terminal Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for an EtherCAT Slave Terminal verification results file.

File Name

File	File name
EtherCAT Slave Terminal verification results file	VerifyResult_ECAT_NX.log

File Creation Timing and Created Directories

Operation	Operating method	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group.

You can check the verification results in the portion that gives the verification results for the EtherCAT Coupler Units and NX Units.

“Result=Matched” indicates a data group for which no differences were found. “Result=Not matched” indicates a data group for which differences were found.

The following table gives an example of the verification results for the following file contents.

- Matched: EtherCAT Coupler Unit E002, NX Unit N1, and NX Unit N2
- Not matched: EtherCAT Coupler Unit E005 and NX Unit N3

File contents	Description
[Verification Results] ; --- NX Parameters. --- [E002:UnitNo.0 NX-ECC201] Result=Matched [N1:UnitNo.1 NX-AD2203] Result=Matched [N2:UnitNo.2 NX-DA2203] Result=Matched [N3:UnitNo.3 NX-TS3201] Result=Not matched Factor=Verification error [E005:UnitNo.0 NX-ECC201] Result=Not matched Factor=Verification error	The Units are indicated in the following format: {Device name}:UnitNo.{Unit number}[blank]{Unit model} Device Name: The device name set by the user. Unit Number: Text string of decimal numbers. The value will be between 0 and 125. Unit Model: Text string that identifies the Unit model. Consecutive spaces at the end of the model number are deleted. The verification results are given as follows: Result=Matched Same Result=Not matched Different

9-11-8 Specifications of CJ-series Unit Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for a CJ-series Unit verification results file.

Precautions for Correct Use

You can use CJ-series Units only with NJ-series CPU Units.

File name

File	File name
CJ-series Unit Verification Results File	VerifyResult_CJUnit.log

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group. You can check the verification results in the portion that gives the verification results for each CJ-series Unit. “Result=Matched” indicates a data group for which no differences were found. “Result=Not matched” indicates a data group for which differences were found. In the file shown below, CJ1W-CRM21 (MODE0) and CJ1W-EIP21 matched, and CJ1W-DRM21 and CJ1W-PRM21-DPV1 did not match.

File contents	Description
[Verification Results] ; --- CJ Unit Parameters. --- [Rack0 Slot0: CJ1W-CRM21(MODE0) UnitNo.10] Result=Matched [Rack0 Slot9: CJ1W-DRM21 UnitNo.0] Result=Not matched [Rack1 Slot0: CJ1W-EIP21 UnitNo.10]] Result=Matched [Rack1 Slot1: CJ1W-PRM21-DPV1 UnitNo.1] Result=Not matched	The Units are given in the following format: Rack{Rack No.}[space]Slot{Slot No}:[space]{Unit model}[space]Unit No.{unit number} Rack No.: Text string of decimal numbers. The value will be between 0 and 3. Slot No.: Text string of decimal numbers. The value will be between 0 and 9. Unit Model Text string that identifies the Unit model. The Unit model is obtained from the cyclic initialization data. Consecutive spaces at the end of the model number are deleted. Unit No.: Text string of decimal numbers. Leading zeros are suppressed. Range for a CPU Bus Unit: 0 to 15. Special I/O Units: 0 to 95. The match/no match results are given in the following format: Result=Matched: No differences were found. Result=No matched: Differences were found.

Note The verification results are given only for the EtherCAT slaves that were compared.

9-12 Compatibility between Backup-related Files

The files may not be compatible if you back up and restore data under different conditions.

The files may not be compatible in these three cases:

- When the function that was used to back up data is different from the function that was used to restore it.
- When the model number of the CPU Unit where the data was backed up from does not match the model number where data is being restored.
- When the unit versions of the CPU Unit, other Units, or slaves where the data was backed up from do not match the unit versions where data is being restored.

In this context, the term “restore” is used collectively for these backup functions: restore, automatic transfer, and read (back up).

9-12-1 Compatibility between Backup Functions

The following table shows the file compatibility when the function used to back up the data is different from the function used to restore it.

(C: Compatible, N: Not compatible.)

Function used to back up data	Function used to restore data				
	Restoring with SD Memory Card backup functions (SD Memory Card to Controller)	Automatic transfer	Restoring with Sysmac Studio Controller backup functions (computer to Controller)	Restoring with Sysmac Studio variable and memory backup functions (computer to Controller)	Importing Sysmac Studio backup file data (computer to project)
Backing up with SD Memory Card backup functions (Controller to SD Memory Card)	C	C	C	N	C*1
Backing up with Sysmac Studio Controller backup functions (Controller to computer)	C	C	C	N	C*1
Backing up with Sysmac Studio variables and memory data backup functions (Controller to computer)	N	N	N	C	N
Exporting from a Sysmac Studio backup file (project to computer)	C*1	C*1	C*1	N	C

*1 The following data is not included.

- The built-in EtherNet/IP port name and built-in EtherNet/IP tag data link settings in the Controller Setup
- Words allocated to CPU Bus Units in the Unit Configuration
- Operation authority verification
- Data Trace Settings
- Time zone setting

- Present values of variables
- Present values of memory used for CJ-series Units
- Absolute encoder home offsets

Additional Information

The files that are handled for backing up variables and memory from the Sysmac Studio are not compatible with other backup files. Refer to *9-7 Sysmac Studio Variable and Memory Backup Functions* for details on these functions.

9-12-2 Compatibility between CPU Unit Models

The following table shows the file compatibility when the CPU Unit model where the data was backed up from is different from the group where the data is being restored.

(C: Compatible, N: Not compatible.)

CPU Unit model where data was backed up	CPU Unit model to restore to				
	NX701-1700 or NX701-1600	NJ501-1500, NJ501-1400, or NJ501-1300,	NJ301-1200, or NJ301-1100	NJ101-1000	NJ101-9000
NX701-1700 or NX701-1600	C	N*1	N*1	N*1	N*1
NJ501-1500, NJ501-1400, or NJ501-1300	N*1	C	N*1	N*1	N*1
NJ301-1200 or NJ301-1100	N*1	N*1	C	N*1	N*1
NJ101-1000	N*1	N*1	N*1	C	N
NJ101-9000	N*1	N*1	N*1	N	C

*1 The Sysmac Studio variable and memory backup functions are compatible. However, a CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher are required. Refer to *9-7-3 Compatibility between CPU Unit Models* for the compatibility between CPU Unit models for the Sysmac Studio variable and memory backup functions.

Even if the CPU Unit models are compatible, there may be restrictions between various CPU Unit models.

The following table shows which restoration function can be used based on whether the CPU Unit models are compatible.

(R: Restored, x: Not restored)

Compatibility between CPU Unit Models	Function used to restore data				
	Restoring with SD Memory Card backup functions (SD Memory Card to Controller)	Automatic transfer	Restoring with Sysmac Studio Controller backup functions (computer to Controller)*1	Restoring with Sysmac Studio variable and memory backup functions (computer to Controller)	Importing Sysmac Studio backup file data (computer to project)
Compatible	R*2	R*2	R	R	R
Not compatible	×*3	×*4	×	×	×

*1 Only the files that were backed up using this function can be restored.

*2 If the contents of the backup file are outside the range of specifications where the data is restored, the Controller will not operate normally. When you operate the Controller, a major fault level Controller error or a partial fault level Controller error will occur. For example, this error occurs if the number of controlled axes that is used is outside the specifications.

*3 A Restore Start Failed observation will occur.

*4 An Error in Starting Automatic Transfer error (a major fault level Controller error) will occur.

9-12-3 Compatibility between Unit Versions of CPU Units

The following table shows the compatibility of backup files when the unit versions of the CPU Unit are different between where the data was backed up and where it is being restored. You can restore data without any restrictions if the unit versions are the same before and after the backup and restoration.

(R: Restored, x: Not restored)

Unit version of CPU Unit	Function used to restore data			
	Restoring with SD Memory Card backup functions (SD Memory Card to Controller)	Automatic transfer	Restoring with Sysmac Studio Controller backup functions (computer to Controller)	Restoring with Sysmac Studio variable and memory backup functions (computer to Controller)
Version of CPU Unit where data is being restored is newer.	R	R	R	R
Unit version of CPU Unit where data is being restored is older.	×	×*1	×	R

*1 An Error in Starting Automatic Transfer error (a major fault level Controller error) will occur.

9-13 Functions That Cannot Be Executed during Backup Functions

The following functions cannot be executed at the same time as any of the backup functions. Do not execute any backup function while the CPU Unit is executing any of these functions. Also, do not execute any of these functions during execution of any of the backup functions.

- While a backup function is being performed
- Synchronization transfer from the computer to the Controller
- Execution of online editing
- Execution of Clear All Memory operation
- Time zone changes
- Execution of the Save Cam Table instruction (MC_SaveCamTable)
- Execution of CPU Unit name write operation
- Execution of transferring Slave Terminal parameters

10

Communications Setup

This section describes how to go online with the CPU Unit and how to connect to other devices.

10-1 Communications System Overview

This section gives an overview of the communications systems that are supported by NJ/NX-series Controllers.

The shaded steps in the overall procedure that is shown below are related to the communications systems.

Step 1. Software Design

- Step 1-1 Designing I/O and Processing
- Step 1-2 Designing Tasks
- Step 1-3 Designing Programs

↓

Step 2. Software Setups and Programming

- Step 2-1 Slave and Unit Configurations
- Step 2-2 Controller Setup
- Step 2-3 Programming
- Step 2-4 Offline Debugging

↓

Step 3. Mounting and Setting Hardware

↓

Step 4. Wiring

↓

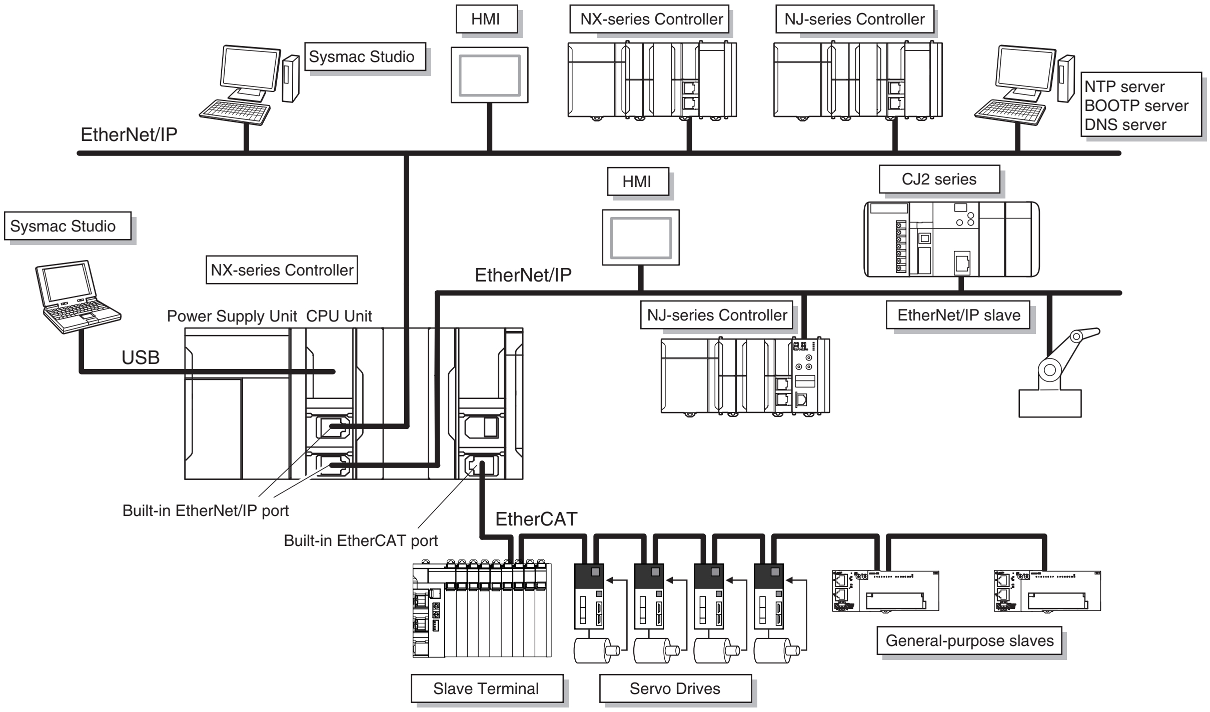
Step 5. Checking Operation and Starting Operation on the Actual System

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series Controller* for details.

10-1-1 Introduction

● **NX-series System**

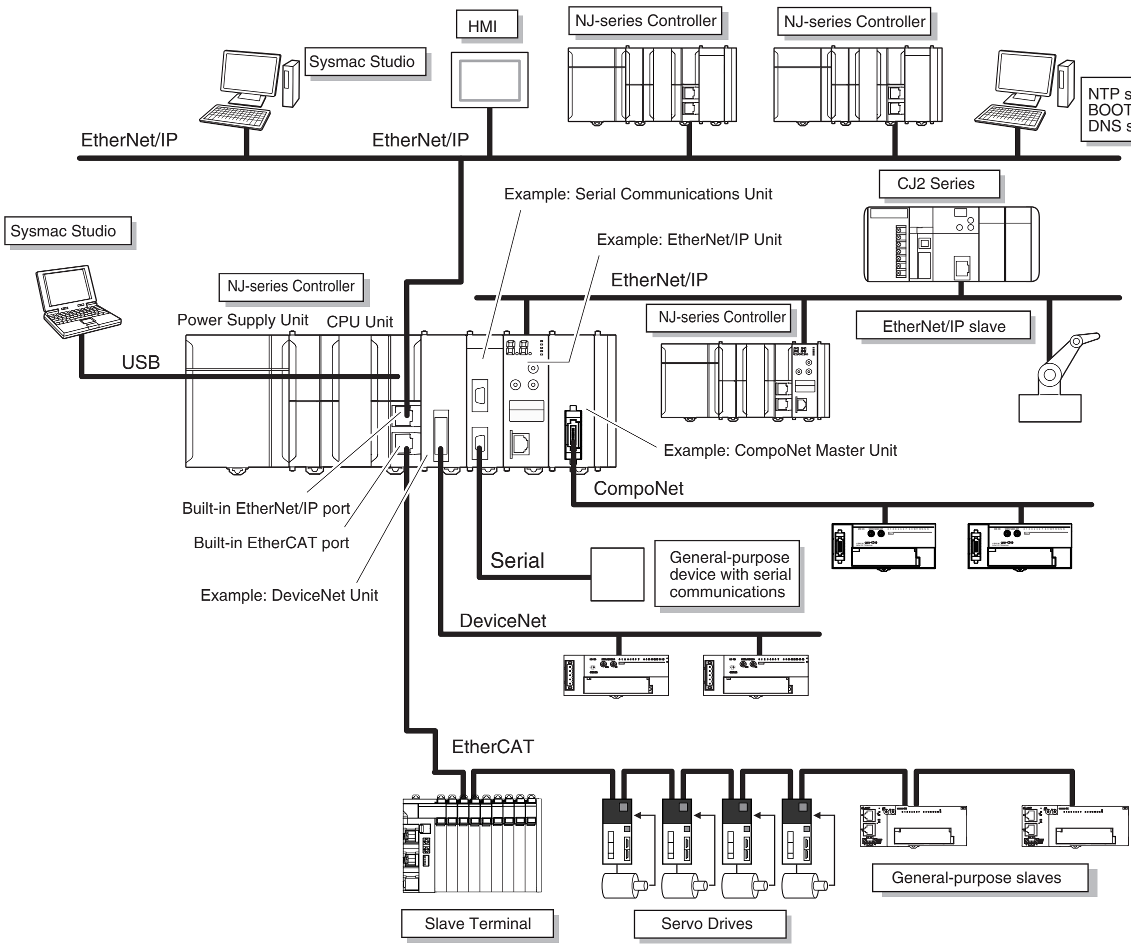
You can use the NX-series System to build the communications system shown below.



Connection		Connection method
Sysmac Studio connection		Use USB or the built-in EtherNet/IP port.
Connections between Controllers	Connections with NJ/NX-series Controller or CJ2 CPU Unit	Use the built-in EtherNet/IP port.
Connections between Controllers and slaves	Connections to Servo Drives and general-purpose slaves	Use the built-in EtherCAT port.
Connections to HMIs		Use the built-in EtherNet/IP port.
Connections to servers	Connections to FTP servers, BOOTP servers, DNS servers, or NTP servers	Use the built-in EtherNet/IP port.

● NJ-series System

You can use the NJ-series System to build the communications system shown below.



Connection		Connection method
Sysmac Studio connection		Use USB or the built-in EtherNet/IP port.
Connections between Controllers	Connections with NJ-series Controller or CJ2 CPU Unit	Use the built-in EtherNet/IP port or a port on an EtherNet/IP Unit*1
Connections between Controllers and slaves	Connections to Servo Drives and general-purpose slaves	Use the built-in EtherCAT port.
	I/O controls	Mount a DeviceNet Unit and use DeviceNet or mount a CompoNet Master Unit and use CompoNet.
Connections to HMIs		Use the built-in EtherNet/IP port or a port on an EtherNet/IP Unit*1, *2
Connections for serial communications		Mount a Serial Communications Unit.
Connections to servers	Connections to FTP servers*3, BOOTP servers, DNS servers, or NTP servers	Use the built-in EtherNet/IP port or a port on an EtherNet/IP Unit*1

*1 Use an EtherNet/IP Unit with unit version 2.1 or later.
Also use a CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher. Refer to *A-14 Version Information for NJ-series Controllers* for information on version upgrades.

*2 Connect an NA-series PT to the built-in EtherNet/IP port on the CPU Unit. To perform troubleshooting from an NS-series PT, connect the PT to the built-in EtherNet/IP port on the CPU Unit.

*3 A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use FTP servers.

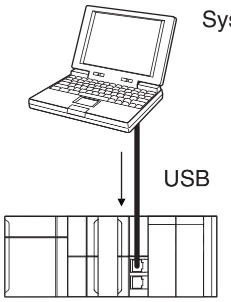
10-2 Connection with Sysmac Studio

This section describes the configurations for connecting the Sysmac Studio to an NJ/NX-series Controller.

10-2-1 Configurations That Allow Online Connections

You can connect online from the Sysmac Studio to the peripheral USB port or built-in EtherNet/IP port of the NJ/NX-series CPU Unit.

● Connecting with USB

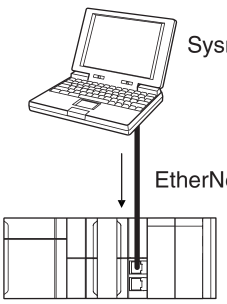


A direct connection is made from the computer that runs Sysmac Studio. You do not need to specify the connection device.

Note Connect a computer and the CPU Unit with a USB 2.0 certified cable. Do not use a USB hub to connect them.

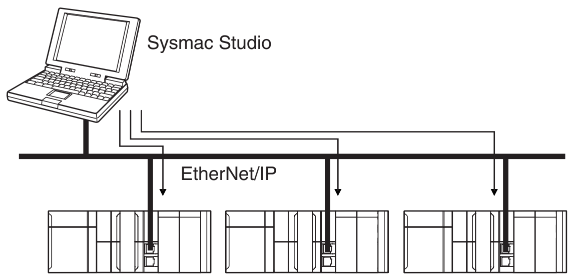
● Connecting with EtherNet/IP

1:1 Connection



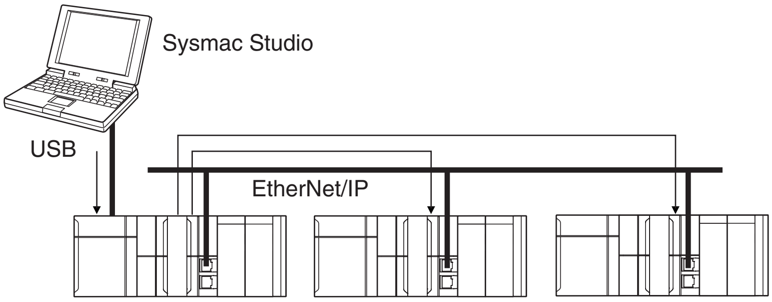
- A direct connection is made from the computer that runs Sysmac Studio. You do not need to specify the IP address or connection device.
- You can make the connection either with or without a Ethernet switch.
- You can use either a cross cable or a straight cable.
- For NX-series CPU Units, 1:1 connection is supported only for port 1.

1:N Connections



- Specify the IP address of the remote node from the Sysmac Studio.
- You can use either a cross cable or a straight cable.

Connecting to EtherNet/IP through USB



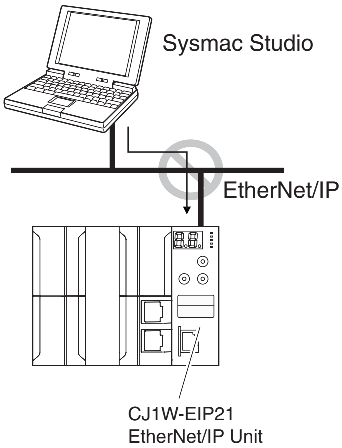
- Specify the IP address of the remote node from the Sysmac Studio.
- You can use either a cross cable or a straight cable.

Note Connect a computer and the CPU Unit with a USB 2.0 certified cable. Do not use a USB hub to connect them.

10-2-2 Configurations That Do Not Allow Online Connections

Connection through an EtherNet/IP Unit

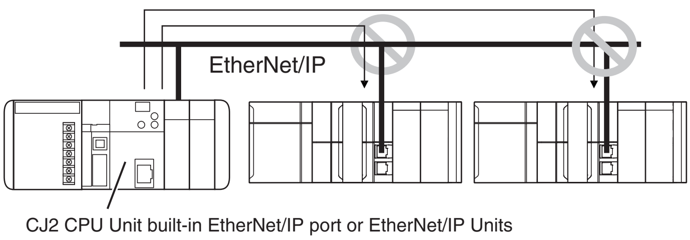
You cannot connect through a CJ1W-EIP21 EtherNet/IP Unit that is connected to an NJ-series CPU Unit.



Note You cannot use a CJ1W-EIP21 EtherNet/IP Unit for NX-series CPU Units.

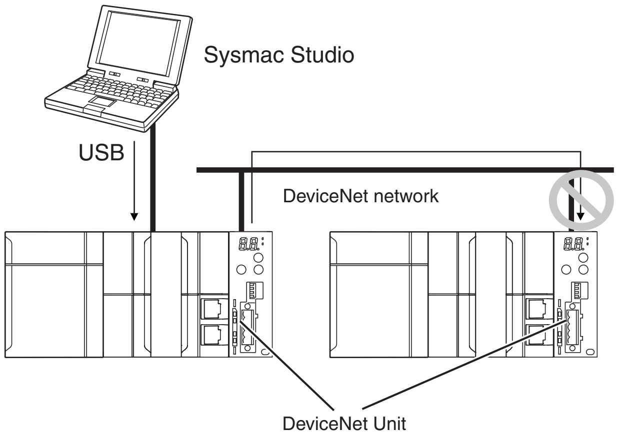
Routing through CS/CJ-series EtherNet/IP Units/Ports

You cannot connect to an NJ/NX-series CPU Unit by routing through a CS/CJ-series Ethernet/IP Unit or port (CS1W-EIP2, CJ1W-EIP21, CJ2 CPU Unit built-in EtherNet/IP port, or CJ2M CPU Unit built-in EtherNet/IP port).



● Routing through Networks Other Than EtherNet/IP, Such as DeviceNet

You cannot route through any networks other than EtherNet/IP networks. (For example, routing is not possible for Controller Link networks and DeviceNet networks.)



Note You cannot connect a DeviceNet Unit with NX-series CPU Units.

10-3 Connection with Other Controllers or Slaves

This section shows the connection configurations that are used between Controllers and between Controllers and slaves.

10-3-1 Connection Configurations between Controllers

EtherNet/IP

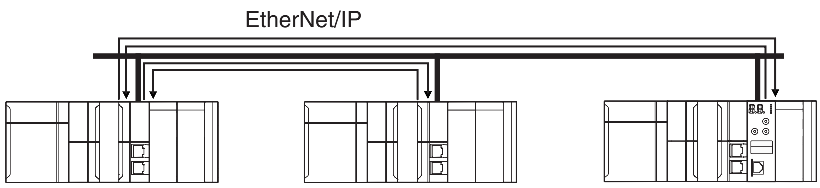
You can use the built-in EtherNet/IP ports or ports on CJ1W-EIP21 EtherNet/IP Units.

For information on the built-in EtherNet/IP port, refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506). For information on the CJ1W-EIP21 EtherNet/IP Unit, refer to the *CJ-series EtherNet/IP Unit Operation Manual for NJ-series CPU Unit* (Cat. No. W495).

Note You cannot use a CJ1W-EIP21 EtherNet/IP Unit for NX-series CPU Units.

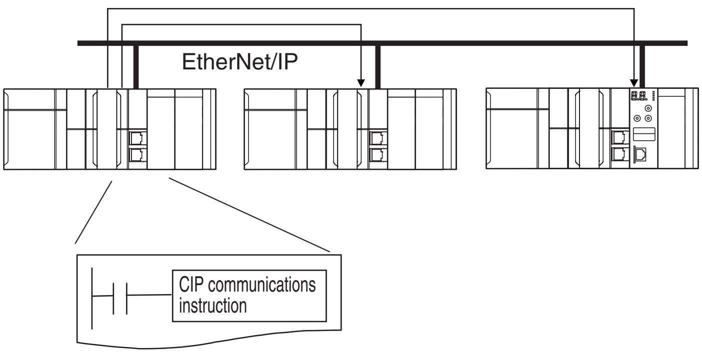
● Tag Data Links

You can create tag data links between NJ/NX-series CPU Units on an EtherNet/IP network.



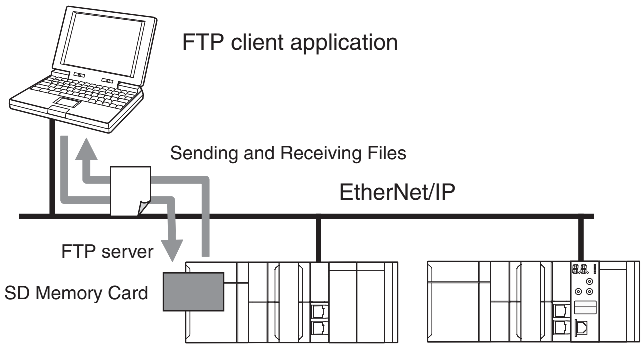
● Message Communications

You can send CIP messages from the user program.



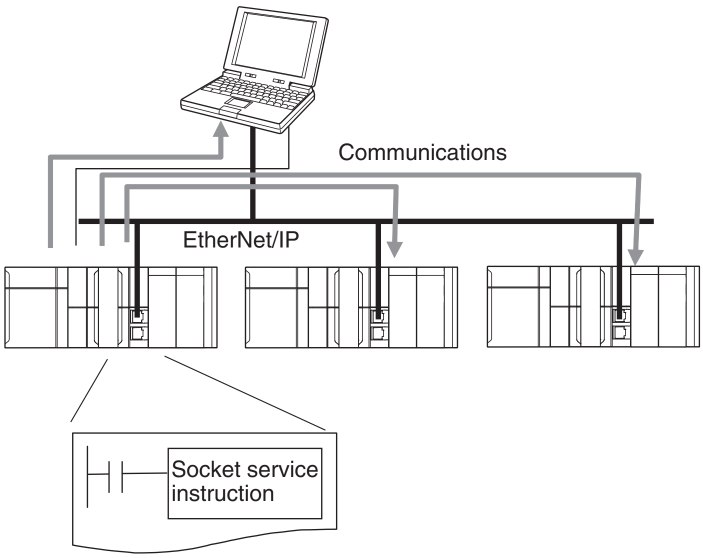
● Sending and Receiving Files

You can send and receive files on the SD Memory Card that is inserted in the NJ/NX-series CPU Unit from an FTP client application.



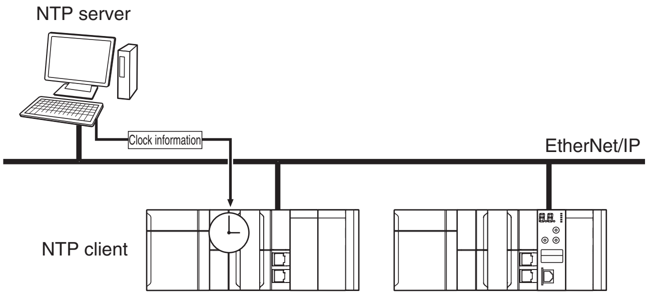
● Socket Services

You can directly use TCP or UDP from the user program to send and receive any data with remote nodes between a host computer and the Controller, or between Controllers. The socket services are supported only for the built-in EtherNet/IP ports.



● Updating Clock Information

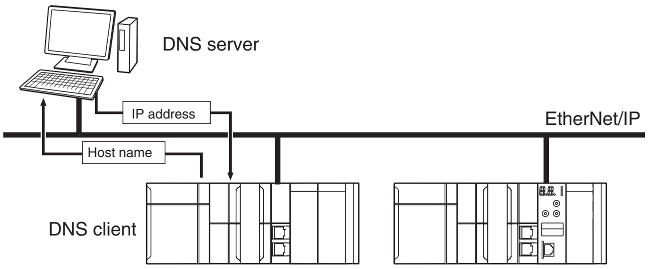
You can obtain clock information from an NTP server to update the built-in clock.



● Specifying Host Names

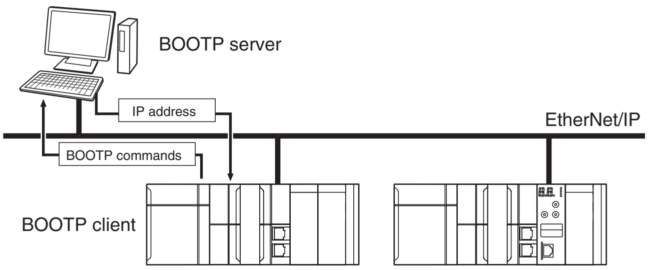
You can use the DNS client or set up your Hosts so that you can specify the IP address of the NTP server or SNMP manager or the target destination of a socket instruction or CIP communications instruction with a host name instead of an IP address.

Example: Setting Host Names on the DNS Server



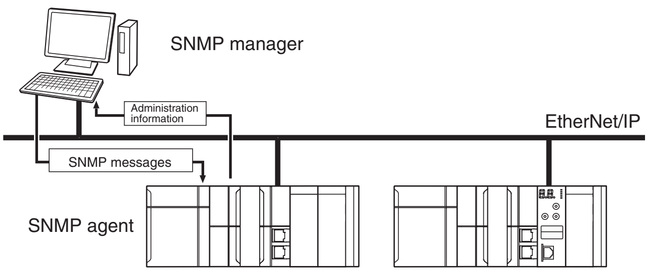
● Obtaining an IP Address When the Power is Turned ON

You can obtain an IP address for the built-in EtherNet/IP port from the BOOTP server when the power supply is turned ON.



● Specifying an SNMP Agent

Built-in EtherNet/IP port internal status information is provided to network management software that uses an SNMP manager.

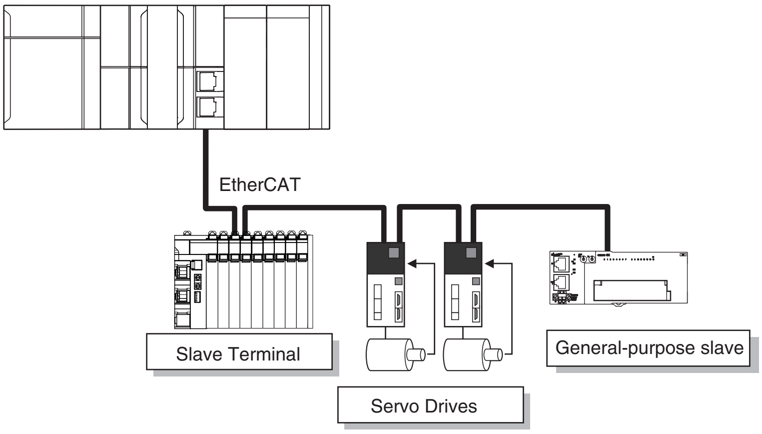


10-3-2 Connection Configuration between Controllers and Slaves

EtherCAT

High-speed, high-precision communications are possible with Servo Drives and general-purpose slaves.

Refer to the *NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual* (Cat. No. W505) for details.



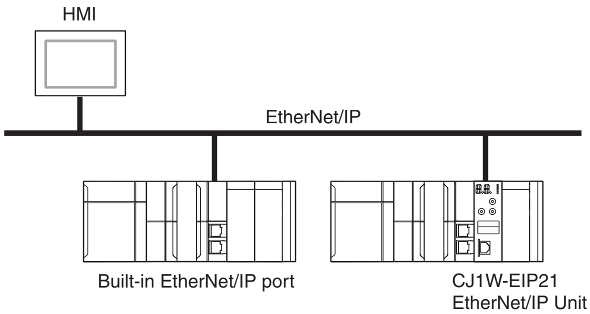
10-4 Connection with HMIs or Serial Communications Devices

This section shows the connection configurations used to connect HMIs and devices with serial communications to the NJ/NX-series Controller.

10-4-1 Connections to HMIs

● EtherNet/IP

You can use a built-in EtherNet/IP port or a CJ1W-EIP21 EtherNet/IP Unit to connect to an HMI.



Note You cannot use a CJ1W-EIP21 EtherNet/IP Unit for NX-series CPU Units.

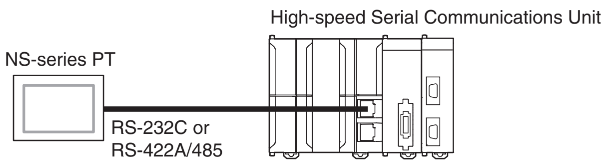
Connect an NA-series PT to the built-in EtherNet/IP port on the CPU Unit.

To perform troubleshooting from an NS-series PT, connect the PT to the built-in EtherNet/IP port on the CPU Unit.

For information on the built-in EtherNet/IP port, refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506). For information on the CJ1W-EIP21 EtherNet/IP Unit, refer to the *CJ-series EtherNet/IP Unit Operation Manual for NJ-series CPU Unit* (Cat. No. W495).

● Serial Communications

You can use a Serial Communications Unit to connect to an NS-series PT.

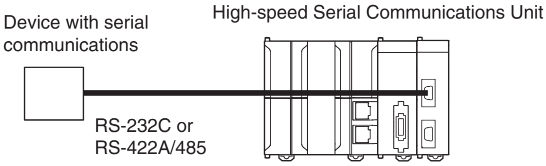


Note You cannot use a Series Communications Unit for NX-series CPU Units.

You can use a Serial Communications Unit to connect to an HMI. Refer to the *CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit* (Cat. No. W494) for details.

10-4-2 Connections to Devices with Serial Communications

You can use a Serial Communications Unit to connect to an HMI. Refer to the *CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit* (Cat. No. W494) for details.



Note You cannot use a Serial Communications Unit for NX-series CPU Units.

11

Example of Actual Application Procedures

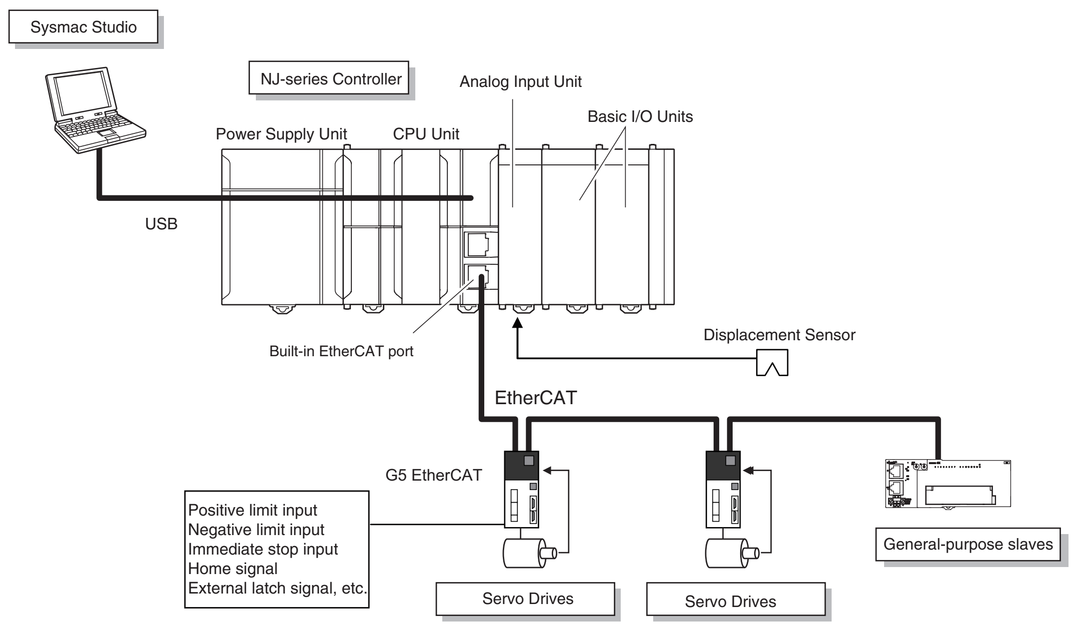
This section describes the procedures that are used to actually operate an NJ-series Controller.

11-1 Example Application

This section describes an example application for an NJ-series Controller.

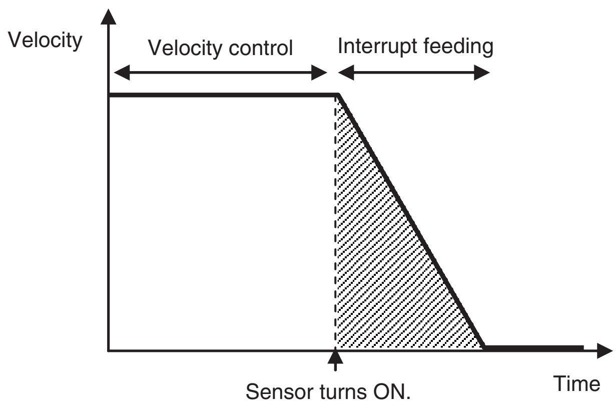
11-1-1 System Configuration

Unit name		Qty	Connected device
Power Supply Unit		1	---
CPU Unit		1	---
CJ-series Basic I/O Units		2	---
CJ-series Analog Input Unit		1	Displacement Sensor
EtherCAT slaves	Servo Drives (G5 EtherCAT)	2	---
	I/O Terminal	1	---



11-1-2 Operation

Interrupt feeding starts when the sensor signal changes to ON during velocity control.



The vertical position changes based on the analog input from the Displacement Sensor.

11-2 Overview of the Example Procedure

This section describes examples of the actual operating procedures for an NJ-series Controller.

11-2-1 Wiring and Settings

Wire the Controller and make the hardware settings.

11-2-2 Software Design

Design the I/O, tasks, POUs, and variables.

I/O Design

- Design the relationship between the external I/O and the unit configuration.
- Determine the intervals at which to refresh external I/O.

Task and POU Design

Consider the following:

- What task configuration is required
- Which programs to assign to which tasks
- Which Units to assign to which tasks
- What processing to place in programs and what processing to place in function blocks and functions

Variable Design

Consider the following:

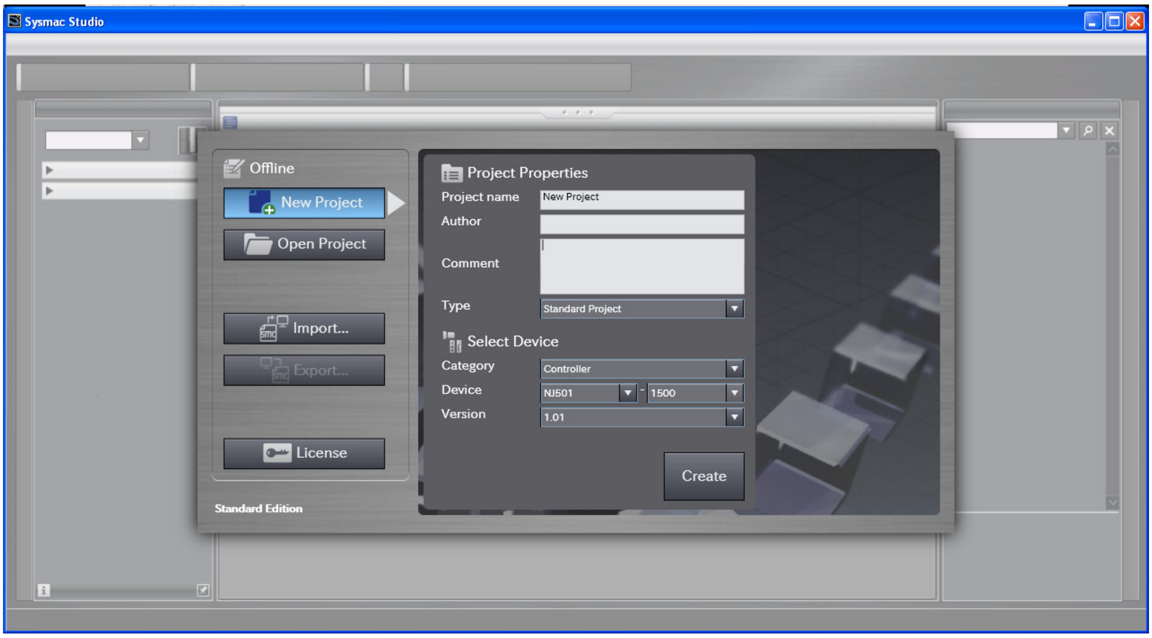
- The separation of variables into those that you use in more than one POU (global variables) and variables that you use in only specific POUs (local variables)
- Defining the variable names for the device variables that you use to access slaves and Units
- Defining the attributes of variables, such as the Name and Retain attributes
- Designing the data types of variables

11-2-3 Software Settings from the Sysmac Studio

On the Sysmac Studio, you set the Unit and slave configurations, register global variables and device variables, create axes (axis variables), and set the Controller Setup and Special Unit Setup.

Start the Sysmac Studio.

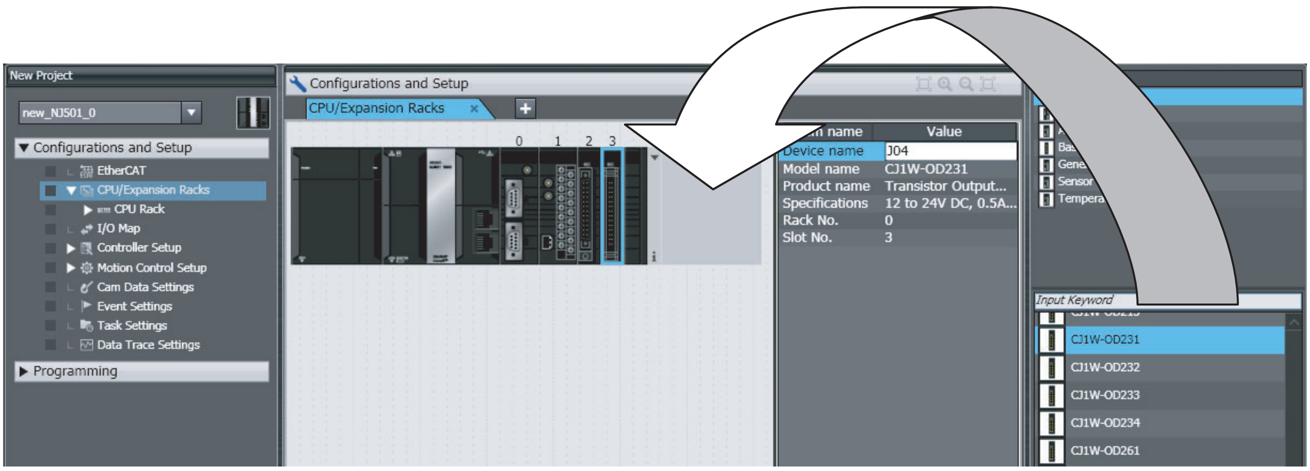
Create a project in Sysmac Studio.



Create the Unit Configuration.

1 Double-click **CPU/Expansion Racks** under **Configurations and Setup**.

2 Create the Unit configuration by dragging Units.

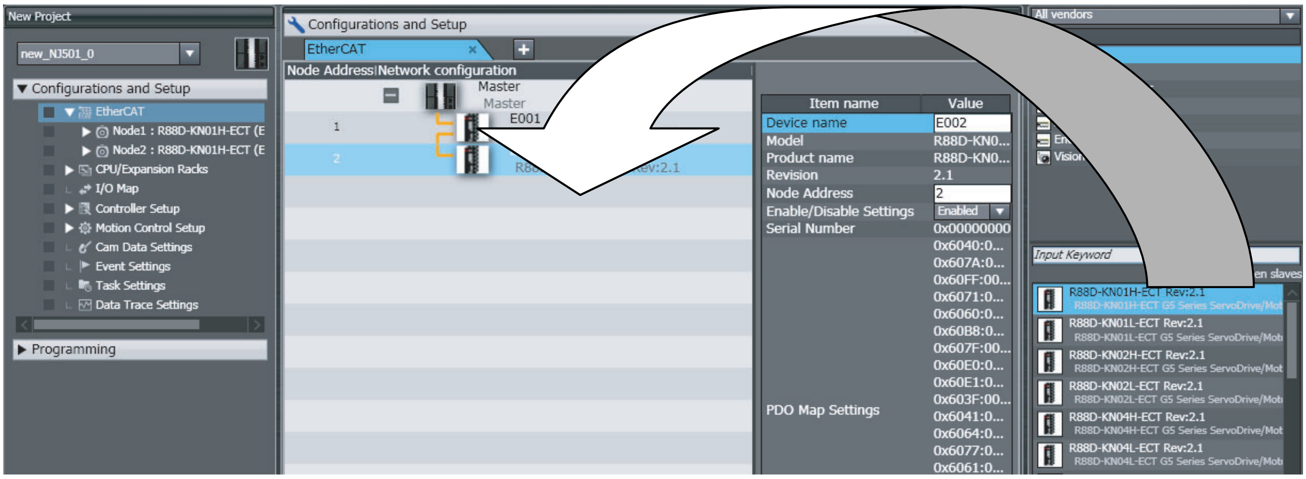


3 Select each Unit and make the required settings.

Create the EtherCAT Slave Configuration.

1 Double-click **EtherCAT** under **Configurations and Setup**.

2 Create the slave configuration by dragging slaves.



3 Select the master and set the master parameters.

4 Select each slave and set the slave parameters.

Additional Information

At this point, you can use forced resetting from the I/O Map to check the wiring.

Register the Global Variables and Device Variables.

● Registering Global Variables

1 Double-click **Global Variables** under **Programming** – **Data**.

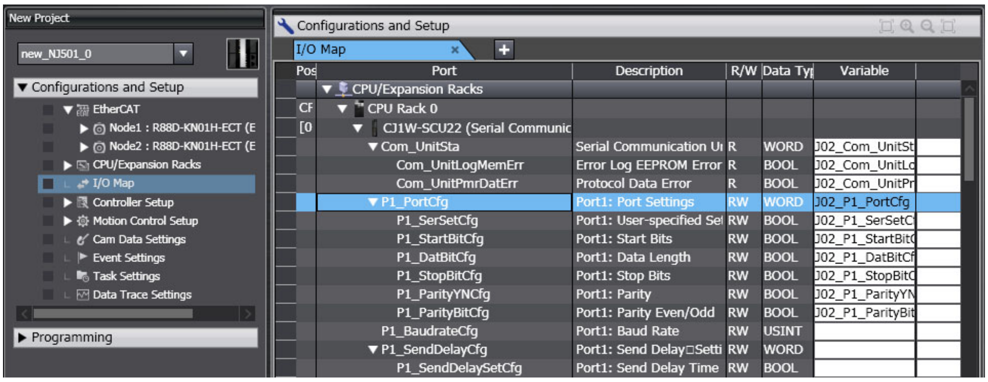
2 Register the global variables in the global variable table.

● Registering Device Variables

1 Double-click **I/O Map** under **Configurations and Setup**.

2 In the I/O Map, assign the variables to the I/O ports. (The I/O ports are created automatically from the Unit and slave configurations.)

You can automatically create device variable names with the Sysmac Studio. To do so, right-click an I/O port and select ***Create Device Variable*** from the menu.

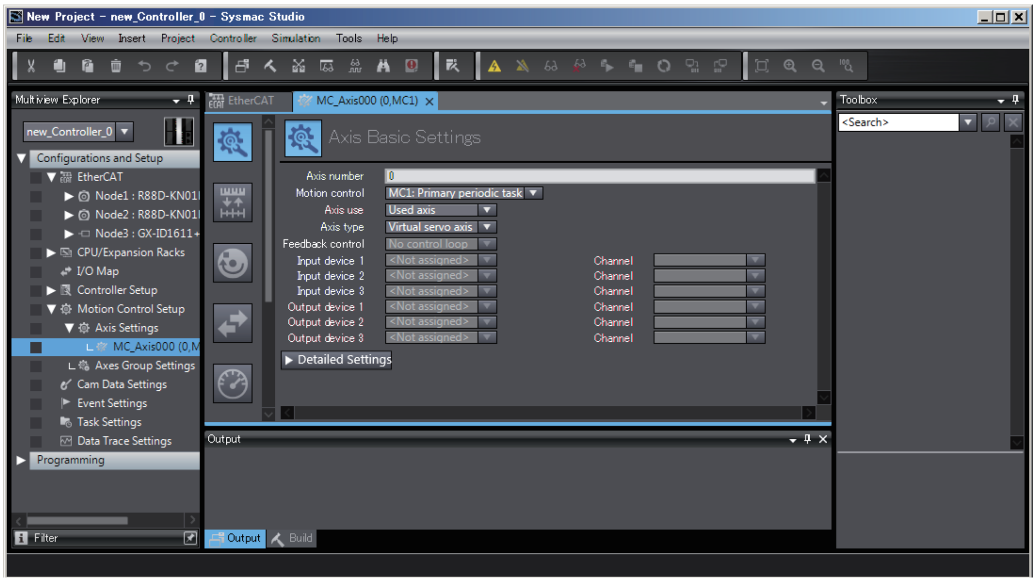


By default, device variables are registered in the global variable table. If necessary, you can change the variable type from a global variable to a local variable (internal variable) for a POU.

Create Axes (Axis Variables)

1 Right-click **Axis Settings** under **Configurations and Setups** – **Motion Control Setup** and select ***Add*** – ***Axis Settings*** from the menu.

2 Assign Servo Drives to the axes (axis variables) that you created in the EtherCAT configuration.



- Set the *Axis Use* parameter to *Used Axis*.
- For an NX-series CPU Unit, set the *Motion Control* parameter to *Primary periodic task*.
- Set the *Axis Type* parameter to *Servo Axis*.
- Set the *Input Device* and *Output Device* parameters to the EtherCAT slaves that you registered in the slave configuration.

Set the other parameters, such as the Unit Conversion Settings and Operation Settings.

Set the Controller Setup and the Special Unit Setup.

● Initial Settings for the PLC Function Module:

The Controller Setup includes the Startup Mode and other parameters.

● Initial Settings for Special Units:

Unit Configuration and Setup: Set the initial settings of the Analog Input Unit.

11-2-4 Programming with the Sysmac Studio

On the Sysmac Studio, create the programs, set the tasks, and build the project.

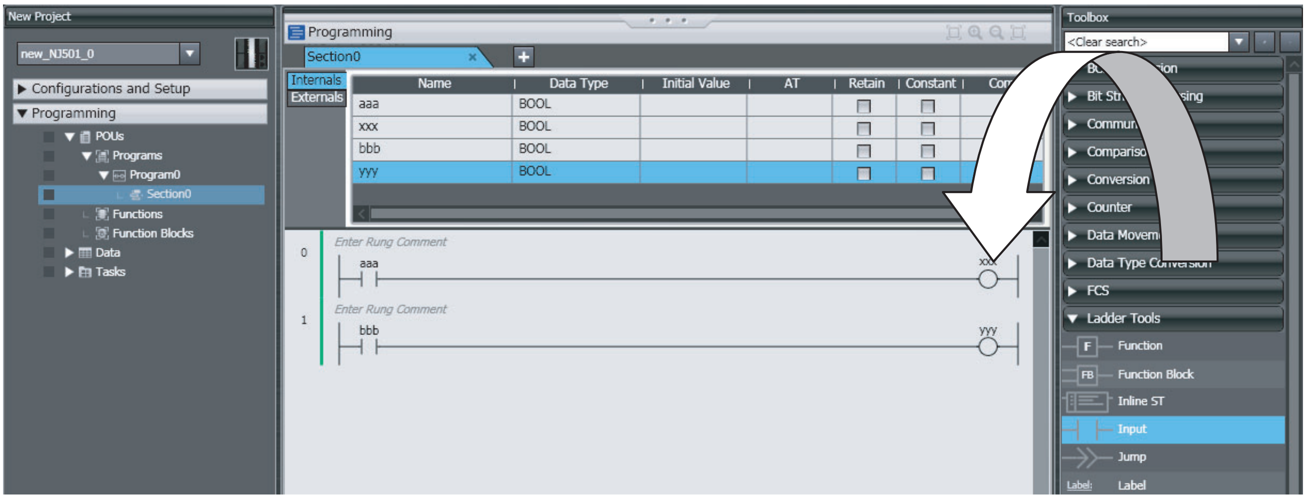
Write the Programs.

1 Right-click **Programs** under ***Programming*** – ***POU*** and select ***Add*** – ***Ladder or Add*** – ***ST*** from the menu.

2 Double-click **Section☐** under the program that you registered.

3 Register the local variables for each program.

4 Enter the programs.



Create a program with the following instructions.

- Homing: MC_Home instruction
- Velocity control: MC_MoveVelocity instruction
- Interrupt feeding: MC_MoveFeed instruction
- Positioning: MC_Move instruction

5 As required, right-click **Functions** or **Function Blocks** under **Programming** – **POU** and select ***Add*** – ***Ladder or Add*** – ***ST*** from the menu.

Double-click the function or function block that you registered. Register local variables for each function and function block. Create the algorithms.

Note For a ladder diagram, press the **R** Key and create the following rungs.

Set Up the Tasks.

Double-click **Task Settings** under **Configurations and Setup**.

- In the *Task Settings*, set the task period and execution condition for the primary periodic task from the pulldown list.
- In the *I/O Control Task Settings*, select the task name to which to assign each Unit and slave.
- In the *Program Assignment Settings*, assign the programs to the primary periodic task or the priority-16 periodic task.

Build the Project.

Select ***Build*** from the Project Menu.

11-2-5 Simulation with the Sysmac Studio

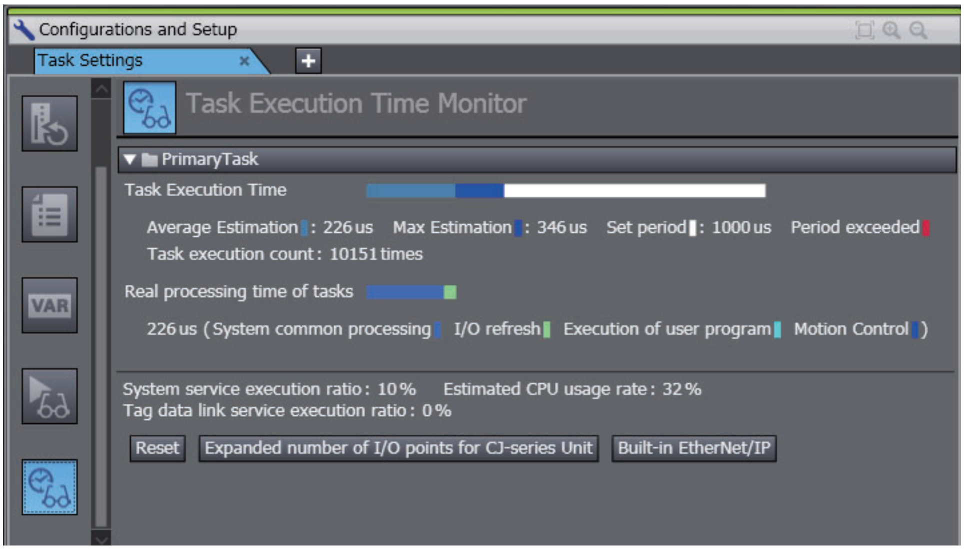
Simulation is used to perform desktop debugging.
Check the task execution times and the real processing times of tasks. Review the task design as required.

Starting the Simulator and Connecting to It

Select ***Run in Execution Time Estimation Mode*** from the Simulation Menu. The Simulator (i.e., the virtual Controller) starts. An online connection is created automatically.

Checking the Task Execution Time on the Simulator

Double-click **Task Settings** under **Configurations and Setup**. Check to see if the task execution times in the Task Execution Time Monitor exceed the task periods.



If necessary, review the task configuration, program assignments, and task periods.

Saving the Project

Select ***Save As*** from the File Menu.

11-2-6 Checking Operation and Starting Operation on the Actual System

Go online with the Controller, download the project, check the wiring and perform test operation before you start actual operation.

Going Online

1 Turn ON the power supply to NJ-series Controller.

2 Connect the computer and the CPU Unit with a USB cable.

3 Select ***Communications Setup*** from the Controller Menu. Select the connection method for the connection configuration in the *Connection Type* Field.

4 Select ***Online*** from the Controller Menu.

Downloading the Project with the Synchronize Menu

Select ***Synchronize*** from the Controller Menu and download the project to the Controller.

Note Use the Synchronize Menu of the Sysmac Studio to upload and download the project.

Checking Wiring

Check the wiring by performing forced-refreshing with user-specified values from the I/O Map or Ladder Editor.

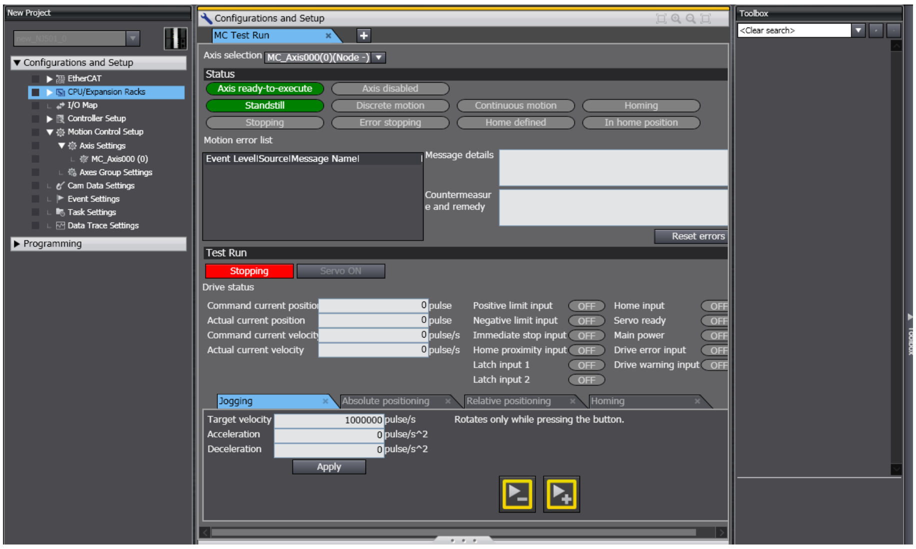
MC Test Run

1 Open the MC Test Run Tab Page.

2 Change the CPU Unit to PROGRAM mode.

3 Monitor input signals on the display to check the wiring.

4 Jog the axis from the display.



Manual Operation

Change the CPU Unit to RUN mode.

- Turning the Servo ON and OFF: Execute the MC_Power motion control instruction.
- Jogging: Execute the MC_MoveJog motion control instruction.

Homing

Homing: Execute the MC_Home instruction.

Actual Operation

Select ***Operation Mode*** – ***RUN Mode*** from the Controller Menu. If an error occurs, investigate the cause and edit the user program.

12

Troubleshooting

This section describes the event codes, error confirmation methods, and corrections for errors that can occur.

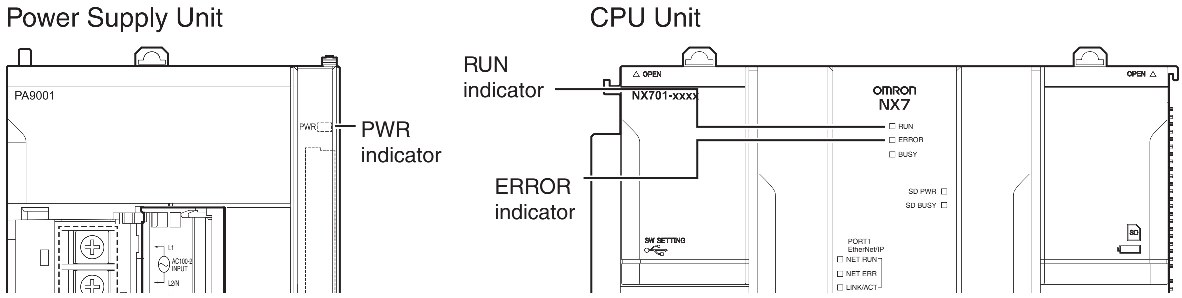
12-1 Operation after an Error

This section describes the error status of the NJ/NX-series Controller and the operation that occurs after an error is detected. Refer to *12-2 Troubleshooting* for details on corrections for specific errors. Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for all of the errors that may occur in an NJ/NX-series Controller.

12-1-1 Overview of NJ/NX-series Status

You can check the operating status of the CPU Unit with the PWR, RUN, and ERROR indicators on the front panels of the Power Supply Unit and CPU Unit.

● NX-series CPU Units



The following table shows the status of front-panel indicators, the status of user program execution, and the ability to connect communications to the Sysmac Studio or an HMI during startup, during normal operation, and when errors occur.

CPU Unit operating status		Power Supply Unit PWR (green)	CPU Unit RUN (green)	CPU Unit ERROR (red)	User program execution status	Communications with Sysmac Studio or an HMI
Startup		Lit	Flashing (2-s intervals followed by 0.5-s intervals)	Not lit	Stopped.	Not possible.
Normal operation	**RUN mode**	Lit	Lit	Not lit	Continues.	Possible.
	PROGRAM mode	Lit	Not lit	Not lit	Stopped.	
Fatal error in CPU Unit	**Power Supply Error*1**	Not lit	Not lit	Not lit	Stopped.	Not possible.
	CPU Unit Reset*1	Lit	Not lit	Not lit	Stopped.	
	CPU Unit Error*1	Lit	Not lit or Flashing (2-s intervals or 0.5-s intervals)	Lit	Stopped.	
	System Initialization Error*1	Lit	Flashing (2-s intervals) for 30 s or longer	Not lit	Stopped.	

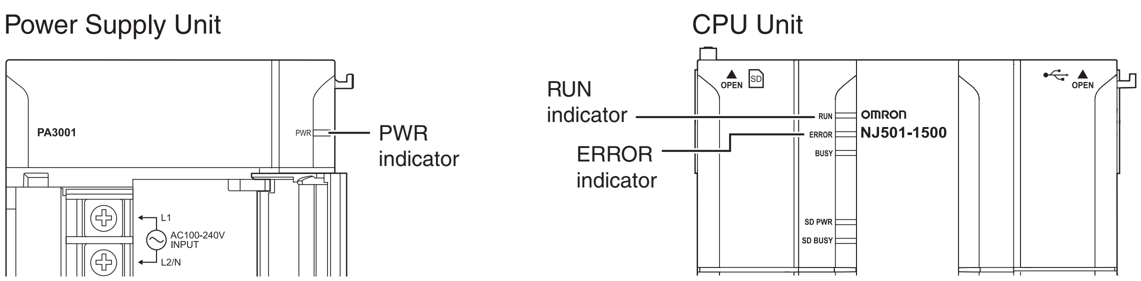
CPU Unit operating status		Power Supply Unit	CPU Unit		User program execution status	Communications with Sysmac Studio or an HMI
		PWR (green)	RUN (green)	ERROR (red)		
Non-fatal error in CPU Unit	Major fault*2	Lit	Not lit	Lit	Stopped.	Possible. (Communications can be connected from an HMI if EtherNet/IP is operating normally.)
	Partial fault*2	Lit	Lit	Flashing (1-s intervals)	Continues.*3	
	Minor fault*2	Lit	Lit	Flashing (1-s intervals)	Continues.	
	Observation*2	Lit	Lit	Not lit	Continues.	

*1 Refer to *12-1-2 Fatal Errors in the CPU Unit* for information on individual errors.

*2 Refer to *12-1-3 Non-fatal Errors in the CPU Unit* for information on individual errors.

*3 The function module where the error occurred stops.

● NJ-series CPU Units



The following table shows the status of front-panel indicators, the status of user program execution, and the ability to connect communications to the Sysmac Studio or an HMI during startup, during normal operation, and when errors occur.

CPU Unit operating status		Power Supply Unit	CPU Unit		User program execution status	Communications with Sysmac Studio or an HMI
		PWR (green)	RUN (green)	ERROR (red)		
Startup		Lit	Flashing (1-s intervals)	Not lit	Stopped.	Not possible.
Normal operation	RUN mode	Lit	Lit	Not lit	Continues.	Possible.
	PROGRAM mode	Lit	Not lit	Not lit	Stopped.	
Fatal error in CPU Unit	Power Supply Error*1	Not lit	Not lit	Not lit	Stopped.	Not possible.
	CPU Unit Reset*1	Lit	Not lit	Not lit	Stopped.	
	Incorrect Power Supply Unit Connected*1	Lit	Flashing (3-s intervals)	Lit	Stopped.	
	CPU Unit Watchdog Timer Error*1	Lit	Not lit	Lit	Stopped.	
Non-fatal error in CPU Unit	Major fault*2	Lit	Not lit	Lit	Stopped.	Possible. (Communications can be connected from an HMI if EtherNet/IP is operating normally.)
	Partial fault*2	Lit	Lit	Flashing (1-s intervals)	Continues.*3	
	Minor fault*2	Lit	Lit	Flashing (1-s intervals)	Continues.	
	Observation*2	Lit	Lit	Not lit	Continues.	

*1 Refer to *12-1-2 Fatal Errors in the CPU Unit* for information on individual errors.

*2 Refer to *12-1-3 Non-fatal Errors in the CPU Unit* for information on individual errors.

*3 The function module where the error occurred stops.

12-1-2 Fatal Errors in the CPU Unit

Types of Fatal Errors

Some errors are fatal and prevent the CPU Unit from operating. This section describes the errors that cause the operation of the CPU Unit to stop. Communications with the Sysmac Studio or an HMI are not possible if there is a fatal error in the Controller.

● Power Supply Error

Power is not supplied, the voltage is outside of the allowed range, or the Power Supply Unit is faulty.

● CPU Unit Reset

The CPU Unit stopped operation because of a hardware error. Other than hardware failures, this error also occurs at the following times.

- The power supply to an Expansion Rack is OFF.
- The I/O Connecting Cable is incorrectly installed.
  - The IN and OUT connectors are reversed.
  - The connectors are not mated properly.
- There is more than one I/O Control Unit on the CPU Rack or there is an I/O Control Unit on an Expansion Rack.

● Incorrect Power Supply Unit Connected

There is a CJ-series Power Supply Unit connected to the NJ-series CPU Unit. The operation of the Controller is stopped.

● CPU Unit Watchdog Timer Error

This error can occur for an NJ-series CPU Unit. This error occurs when the watchdog timer times out because a hardware failure or when temporary data corruption causes the CPU Unit to hang.

● CPU Unit Error

This error can occur for an NX-series CPU Unit. It indicates that there is a hardware failure or that the CPU is running out of control due to temporary data corruption.

● System Initialization Error

This error can occur for an NX-series CPU Unit. It indicates a hardware failure. The RUN indicator will flash at 2-second intervals while the CPU Unit is starting, but if it flashes for 30 seconds or longer, then this error occurs.

Checking for Fatal Errors

You can identify fatal errors based on the status of the PWR indicator on the Power Supply Unit and the RUN and ERROR indicators on the CPU Unit, as well as by the ability to connect communications to the Sysmac Studio.

● NX-series CPU Units

Indicator			Communications with Sysmac Studio	CPU Unit operating status
PWR (green)	RUN (green)	ERROR (red)		
Not lit	Not lit	Not lit	Not possible.*	Power Supply Error
Lit	Not lit	Not lit		CPU Unit Reset
Lit	Not lit or Flashing (2-s intervals or 0.5-s intervals)	Lit		CPU Unit Error
Lit	Flashing (2-s intervals) for 30 s or longer	Not lit		System Initialization Error

* An online connection to the Sysmac Studio is necessary to differentiate between CPU Unit Resets, CPU Unit Errors, and non-fatal errors in the CPU Unit. Power Supply Errors and System Initialization Errors can be differentiated with the indicators. There is no need to see if you can go online with the CPU Unit from the Sysmac Studio.

● NJ-series CPU Units

Indicator			Communications with Sysmac Studio	CPU Unit operating status
PWR (green)	RUN (green)	ERROR (red)		
Not lit	Not lit	Not lit	Not possible.*	Power Supply Error
Lit	Not lit	Not lit		CPU Unit Reset
Lit	Flashing (3-s intervals)	Lit		Incorrect Power Supply Unit Connected
Lit	Not lit	Lit		CPU Unit Watchdog Timer Error

* An online connection to the Sysmac Studio is necessary to differentiate between CPU Unit Resets, CPU Unit Watchdog Timer Errors, and non-fatal errors in the CPU Unit. Power Supply Errors and Incorrect Power Supply Unit Connected errors can be differentiated with the indicators. There is no need to see if you can go online with the CPU Unit from the Sysmac Studio.

12-1-3 Non-fatal Errors in the CPU Unit

Event Levels

Non-fatal errors that occur are managed as Controller events in the NJ/NX-series Controller. Controller events are classified into levels according to the degree of the effect that the events have on control. When an event occurs, the Sysmac Studio or HMI will display the level. Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for details on Controller events.

- Major Fault Level
  These errors prevent control operations for the entire Controller. If a major fault level error is detected, user program execution is stopped immediately and the loads for all slaves (including remote I/O) are turned OFF. With EtherCAT slaves, some NX Units, and some CJ-series Special Units, you can set the slave settings or Unit settings to select whether outputs will go OFF or retain their previous status. You cannot reset major fault level errors from the user program, the Sysmac Studio or an HMI. To recover from a major fault level error, remove the cause of the error, and either cycle the power supply to the Controller or reset the Controller from the Sysmac Studio.
- Partial Fault Level
  These errors prevent control operations in a certain function module in the Controller. The NJ/NX-series CPU Unit continues to execute the user program even after a partial fault level error occurs. You can include error processing in the user program to safely stop any devices in operation. After you remove the cause of the error, execute one of the following to return to normal status.
  - Reset the error from the user program, the Sysmac Studio, or an HMI.
  - Cycle the power supply.
  - Reset the Controller from the Sysmac Studio.
- Minor Fault Level
  These errors prevent part of the control operations in a certain function module in the Controller. The troubleshooting for minor fault level errors is the same as the processing for partial fault level errors.
- Observations
  These errors do not affect the control operations of the Controller. Observations serve as warnings to the user so that the error does not develop into an error at a higher level.
- Information
  Events that are classified as information do not indicate errors.

● Operation for Each Level

The operation that is performed when an error occurs depends on the error level of the Controller event.

Event level / Item		Controller errors				Controller information
		Major fault level	Partial fault level	Minor fault level	Observation	Information
Definition		These errors are serious errors that prevent control operations for the entire Controller.	These errors prevent all of the control in a function module other than PLC Function Module.	Errors that prevent a portion of control in one of the function modules.	Errors that do not affect control.	Information level events are not errors, but information provided to the user in the event log.
Event examples (Only a few examples are provided here. Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for a complete list of errors.)		• Non-volatile Memory Data Corrupted (PLC Function)	• Motion Control Period Exceeded (Motion Control Function Module) • Communications Controller Error (EtherCAT Master Function Module)	• Positive Limit Input Detected (Motion Control Function Module) • Low Battery Voltage (PLC Function Module)	• Packet Discarded Due to Full Receive Buffer (EtherNet/IP Function Module)	• Power Turned ON • Power Interrupted • Memory All Cleared
Front-panel indicators*1	**PWR (green)**	Lit.	Lit.	Lit.	Lit.	Lit.
	RUN (green)	Not lit.	Lit.	Lit.	Lit.	Lit.
	ERROR (red)	Lit.	Flashes at 1-s intervals.	Flashes at 1-s intervals.	Not lit.	Not lit.

Item		Event level: Controller errors				Controller information
		Major fault level	Partial fault level	Minor fault level	Observation	Information
Operation of NJ/NX-series CPU Unit	RUN output on Power Supply Unit	OFF	ON	ON	ON	ON
	User program execution status	Stops.	Continues.*2	Continues.	Continues.	Continues.
	Outputs turned OFF	Yes	No	No	No	No
	Error reset	Not possible.	Depends on the nature of the error.	Depends on the nature of the error.	---	---
	Event logs	Recorded. (Some errors are not recorded.)	Recorded.	Recorded.	Recorded.	Recorded.
Outputs from EtherCAT slaves and Basic Output Units		Refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.	• Errors in EtherCAT Master Function Module: Depends on settings in the slave. • Errors in other function modules: Depends on user program.	Depends on the user program.	Depends on the user program.	Depends on the user program.
Sysmac Studio display (while online)		The error status is automatically displayed in the Controller Status Pane. You can display detailed information in the Troubleshooting Dialog Box.			These items are not displayed on the error display.	

*1 If multiple Controller errors have occurred, the indicators show the error with the highest error level.

*2 Operation stops in the function module (Motion Control Function Module, EtherCAT Master Function Module, or EtherNet/IP Function Module) in which the error occurred.

Operation in the Function Module Where an Error Event Occurred

Event level / Function module	Major fault level	Partial fault level	Minor fault level	Observation
PLC Function Module	User program execution stops.	---	Operation continues.	
Motion Control Function Module	All axes stop. (The stop method depends on the error.)	All axes stop. (The stop method depends on the error.)	• The affected axis/axes group stops. (The stop method depends on the settings.) • The motion control instructions that are related to axis operation are not executed.	• Axis operation continues. • The motion control instructions that are not related to axis operation are not executed.
EtherCAT Master Function Module	I/O refreshing for EtherCAT communications stops. (The slaves operate according to the settings in the slaves.)	EtherCAT communications stop. (The slaves operate according to the settings in the slaves.)	I/O refreshing for EtherCAT communications stops or continues according to the fail-soft operation settings in the master. (If I/O refreshing stops, the slaves operate according to the settings in the slaves.)	I/O refreshing for EtherCAT communications continues.
EtherNet/IP Function Module	Part of the EtherNet/IP communications stop. (Online connection to the Sysmac Studio and communications connection with an HMI are possible. (Output (produce) tags in the tag data links operate according to the tag set settings.)	EtherNet/IP communications stop. (A software connection from the Sysmac Studio or an HMI is not possible.)	Part of EtherNet/IP communications stop. (A software connection from the Sysmac Studio or an HMI is possible if the communications connection is not the cause of the error.)	EtherNet/IP communications continue.

I/O Operation for Major Fault Level Controller Errors

The following table gives the operation of the CPU Unit and the I/O devices for the following errors.

- Unsupported Unit Detected
- I/O Bus Check Error
- End Cover Missing
- Incorrect Unit/Expansion Rack Connection
- Duplicate Unit Number
- Too Many I/O Points
- I/O Setting Check Error

Unit	CPU Unit operation	Unit or slave operation
NX-series Slave Terminal	The NX-series Slave Terminal moves to Safe-Operational state.	Depends on the NX Unit settings.
EtherCAT slave *1	The slave is placed in the Safe-Operational state.	Depends on the slave settings. *2
Servo Drive or NX Unit assigned to an axis	Updating the command values is stopped.	All axes stop immediately.
CJ-series Basic I/O Unit	Refreshing is stopped.	• All outputs are turned OFF. • All inputs are turned OFF.
CJ-series Special Unit	Refreshing is stopped.	Depends on the Unit operating specifications (the ERH indicator lights).
Devices connected with EtherNet/IP	• For the originators of tag data links, the variables and I/O memory addresses for input (consume) tags are not refreshed. • For the targets of tag data links, operation depends on the settings of the tags sets for the output (produce) tags. *3	Depends on the specifications of the connected devices.

*1 Excluding Servo Drives assigned to an axis.

*2 Settings and setting methods depend on the slave. Refer to the manual for the slave. For a Servo Drive, operation depends on the setting of object 605E hex (Fault Reaction Option Code).

*3 You can set whether to clear output or maintain the data from before the error occurred. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506) for details.

The following table gives the operation of the CPU Unit and the I/O devices for the errors that are not listed above.

Unit	CPU Unit operation	Unit or slave operation
NX-series Slave Terminal	The NX-series Slave Terminal moves to Safe-Operational state.	Depends on the NX Unit settings.
EtherCAT slave *1	The slave is placed in the Safe-Operational state.	Depends on the slave settings. *2
Servo Drive or NX Unit assigned to an axis	Updating the command values is stopped.	All axes stop immediately.
CJ-series Basic I/O Unit	• The values of all outputs are cleared to zero. • Input refreshing continues.	• All outputs are turned OFF. • External inputs are refreshed.
CJ-series Special Unit	Refreshing continues.	Depends on the Unit operating specifications.
Devices connected with EtherNet/IP	• For the originators of tag data links, the variables and I/O memory addresses for input (consume) tags are not refreshed. • For the targets of tag data links, operation depends on the settings of the tags sets for the output (produce) tags. *3	Depends on the specifications of the connected devices.

*1 Excluding Servo Drives assigned to an axis.

*2 Settings and setting methods depend on the slave. Refer to the manual for the slave. For a Servo Drive, operation depends on the setting of object 605E hex (Fault Reaction Option Code).

*3 You can set whether to clear output or maintain the data from before the error occurred. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506) for details.

Checking for Non-fatal Errors

Use the following methods to check for non-fatal errors.

Checking method	What you can check
Checking the indicators	You can use the indicators to confirm the Controller error level, the error status of the EtherCAT Master Function Module, and the error status of the EtherNet/IP Function Module.
Checking with the troubleshooting function of Sysmac Studio	You can check for current Controller errors, a log of past Controller errors, error sources, error causes, and corrections. You can also check error logs from CJ-series Special Units.*1
Checking with the Troubleshooter of an HMI*2	You can check for current Controller errors, a log of past Controller errors, error sources, error causes, and corrections.
Checking with instructions that read function module error status	You can check the highest-level status and highest-level event code in the current Controller errors.
Checking with system-defined variables	You can check the current Controller error status for each function module.

*1 Detailed information, such as error causes and corrections, is not displayed.

*2 To perform troubleshooting from an HMI, connect the HMI to the built-in EtherNet/IP port on the CPU Unit. Refer to the appendices of the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for the applicable range of the HMI Troubleshooter.

This section describes the above checking methods.

Checking the Indicators

● Checking the Level of a Controller Error

You can use the PWR indicator on the Power Supply Unit and the RUN and ERROR indicators on the CPU Unit to determine the event level for an error. The following table shows the relationship between the Controller's indicators and the event level.

Indicator			Event level
PWR (green)	RUN (green)	ERROR (red)	
Lit	Not lit	Lit	Major fault level
Lit	Lit	Flashing (1-s intervals)	Partial fault level
			Minor fault level
Lit	Lit	Not lit	Observation

● Checking the Status of EtherCAT and EtherNet/IP Ports

For the EtherCAT and EtherNet/IP ports, use the EtherCAT and EtherNet/IP NET ERR indicators to determine whether an error that affects process data communications has occurred and whether a minor fault level error or higher-level error has occurred. The indicator lets you check the status given in the following table.

Indicator	Indicated status
EtherCAT NET ERR	EtherCAT Port Status • Lit: An error for which normal status cannot be recovered through user actions (i.e., errors for which you must replace the CPU Unit or contact your OMRON representative) has occurred. • Flashing: An error for which normal status can be recovered through user actions has occurred. • Not lit: An error that affects process data communications has not occurred.
EtherNet/IP NET ERR	EtherNet/IP Port Status • Lit: An error for which normal status cannot be recovered through user actions (i.e., errors for which you must replace the CPU Unit or contact your OMRON representative) has occurred. • Flashing: An error for which normal status can be recovered through user actions has occurred. • Not lit: There is no minor fault level or higher-level error.

Checking with the Troubleshooting Function of Sysmac Studio

When an error occurs, you can connect the Sysmac Studio online to the Controller to check current Controller errors and the log of past Controller errors.

● Current Errors

Open the Sysmac Studio's Controller Error Tab Page to check the current error's level, source, source details, event name, event code, cause, and correction. Errors are not displayed for observations.

● Log of Past Errors

Open the Sysmac Studio's Controller Event Log Tab Page to check the times, levels, sources, source details, event names, event codes, details, attached information 1 to 4, actions, and corrections for previous errors.

Error logs from CJ-series Special Units are displayed on the Controller Event Log Tab Page. Detailed information is not displayed. To check detailed information, use the event codes that are displayed and refer to the error codes that are given in the manual for the relevant Unit. Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for the relationship between error codes and event codes.

Refer to the *NJ-Series Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details on troubleshooting with the Sysmac Studio.

Checking with the Troubleshooter of an HMI

If you can connect communications between an HMI and the Controller when an error occurs, you can check for current Controller errors and the log of past Controller errors.

To perform troubleshooting from an HMI, connect the HMI to the built-in EtherNet/IP port on the CPU Unit.

Precautions for Correct Use

Refer to the appendices of the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for the applicable range of the HMI Troubleshooter.

● Current Errors

You can check the current error's event name, event code, level, source, source details, details, and attached information 1 to 4. Observations are not displayed as errors.

● Log of Past Errors

You can check the time of occurrence, level, source, source details, event name, event code, details, attached information 1 to 4 for past errors.

Refer to the relevant HMI manual for information on the HMI Troubleshooter.

Checking with Instructions That Read Function Module Error Status

Instructions are provided that allow you to read the error status of each function module from the user program. These instructions get the status and the event code of the error with the highest level.

Applicable function module	Instruction name	Instruction
PLC Function Module	Get PLC Controller Error Status	GetPLCError
	Get I/O Bus Error Status	GetCJBError
Motion Control Function Module	Get Motion Control Error Status	GetMCError
EtherCAT Function Module	Get EtherCAT Error Status	GetECError
EtherNet/IP Function Module	Get EtherNet/IP Error Status	GetEIPError

For details on the instructions that get error status, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

Checking with System-defined Variables

You can check the error status variables in the system-defined variables to determine the status of errors in a Controller. You can read the error status variables from an external device by using communications. Refer to *A-4 System-defined Variables* for the system-defined variables.

12-2 Troubleshooting

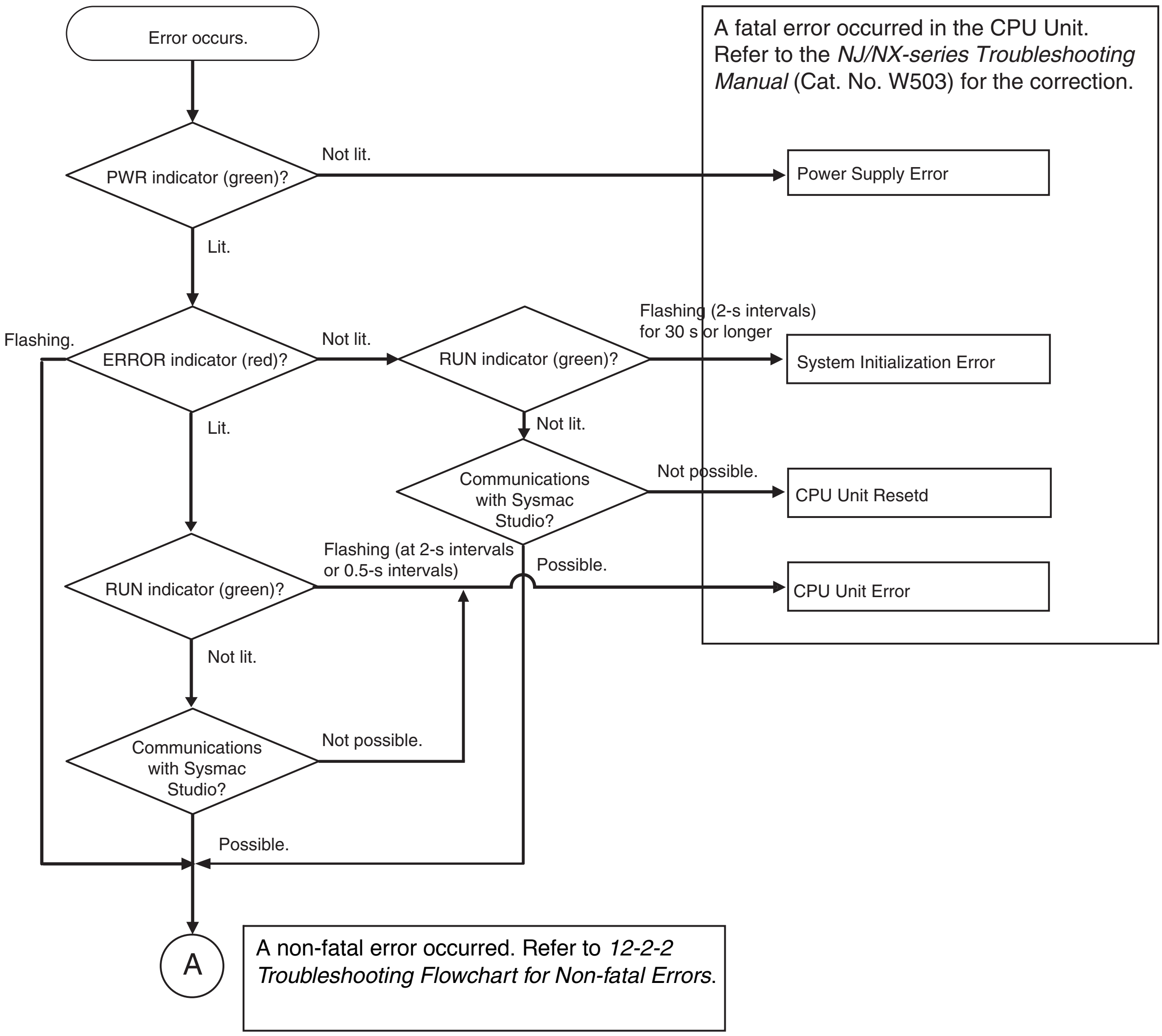
This section provides basic error identification and troubleshooting flowcharts. Use them when an error occurs in the NJ/NX-series Controller. This section also describes the software errors that are related to the PLC Function Module and corrections for those errors.

12-2-1 Checking to See If the CPU Unit Is Operating

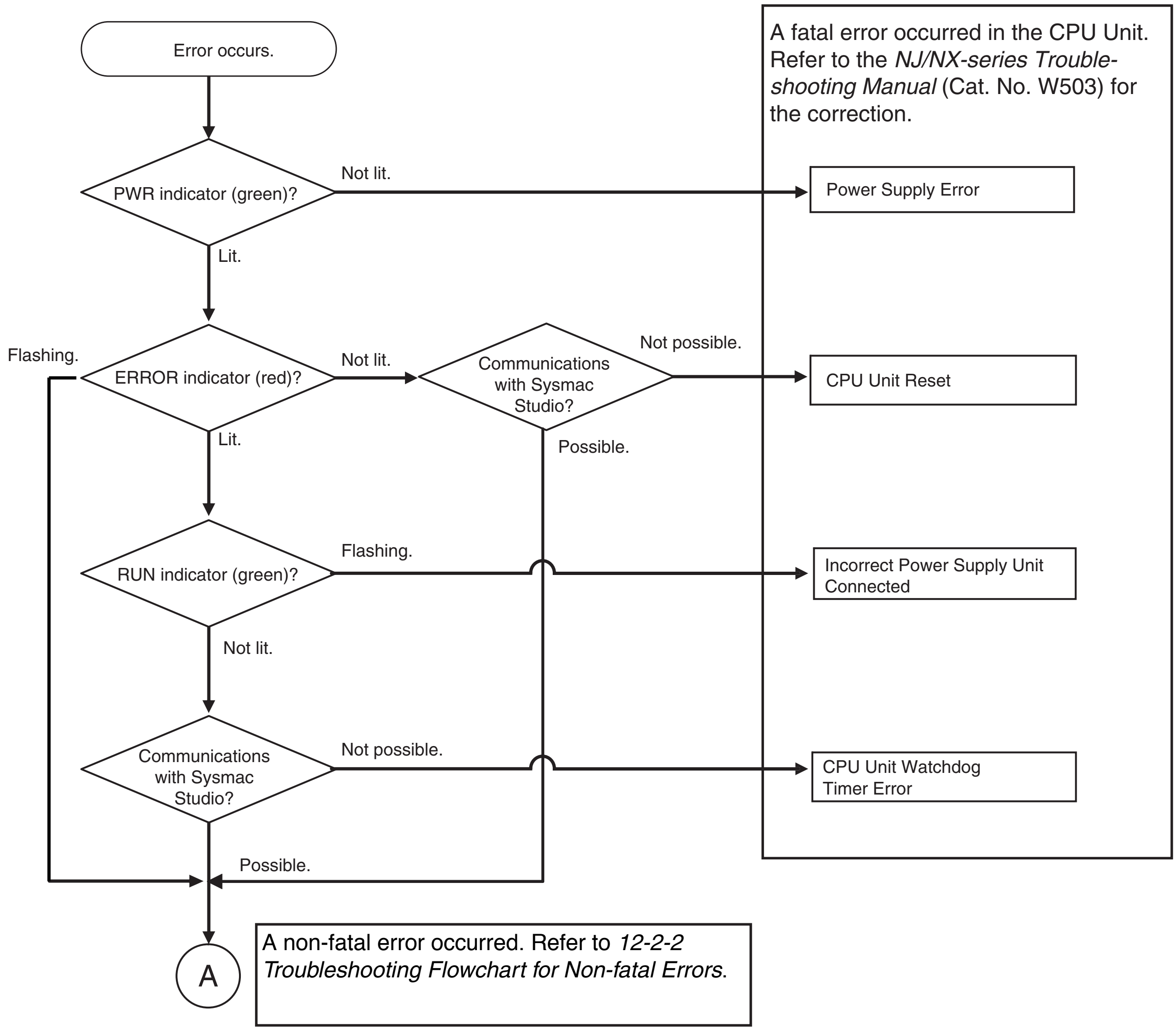
When an error occurs in the NJ/NX-series Controller, use the following flowchart to determine whether the error is a fatal error or a non-fatal error.

Whenever possible, set the Sysmac Studio's communications connection in the flowchart to a direct USB connection. If you use Ethernet, there are many reasons that prevent a communications connection for the Sysmac Studio, so time is required to determine if a fatal or non-fatal error has occurred. If a communications connection from the Sysmac Studio is not possible, perform the troubleshooting procedure that is provided in the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) before you assume that the error is a fatal error.

● NX-series CPU Units



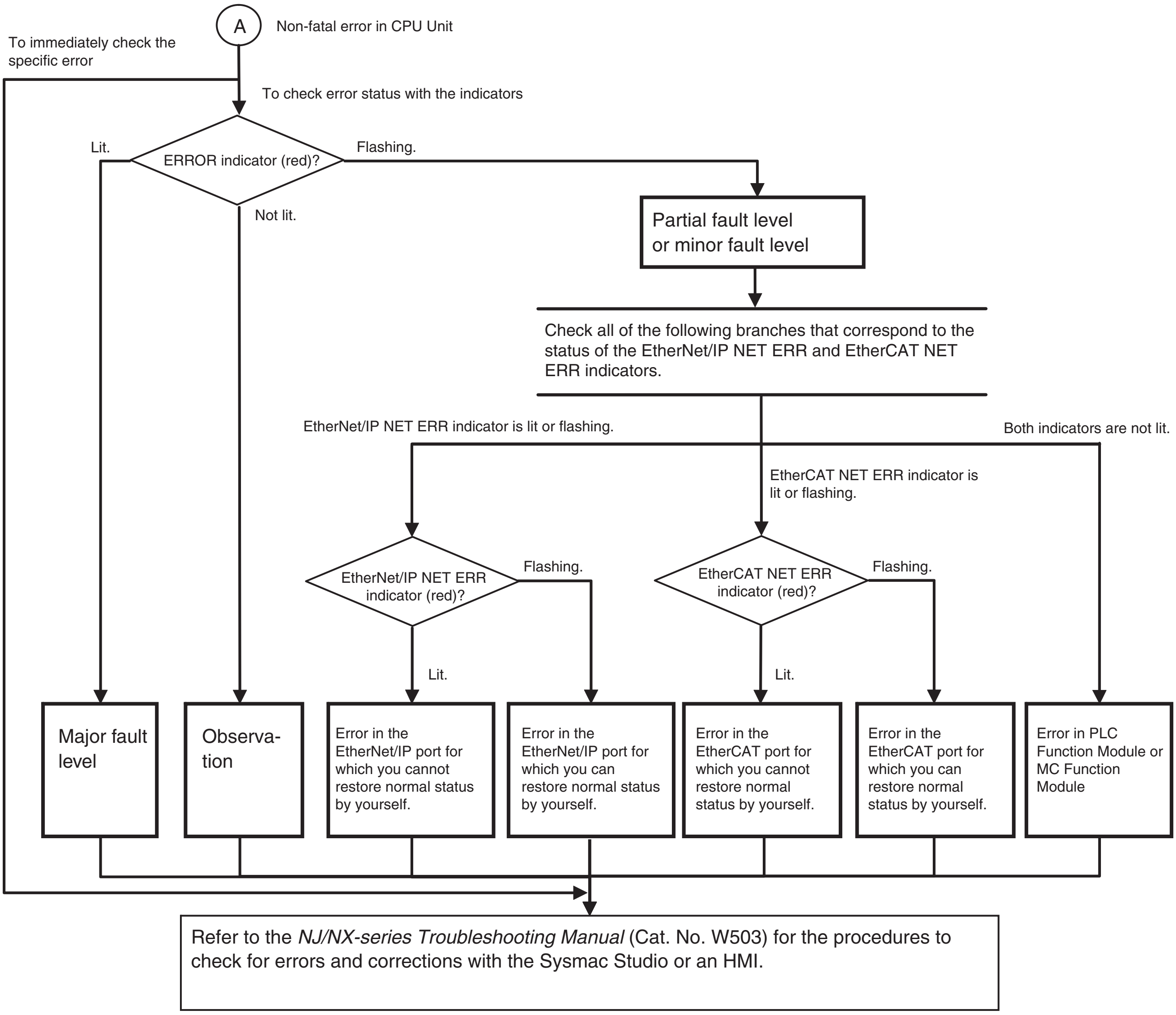
- **NJ-series CPU Units**



12-2-2 Troubleshooting Flowchart for Non-fatal Errors

For a non-fatal error, use the Sysmac Studio or an HMI to troubleshoot the error with the following flowchart. You can use the indicators to check the following:

- Level
- Whether the error is in the EtherNet/IP port or the EtherCAT port
- If the source of the error is the EtherNet/IP port or the EtherCAT port, whether you can restore normal status yourself



Precautions for Correct Use

Refer to the appendices of the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for the applicable range of the HMI Troubleshooter.

12-2-3 Error Table

The software errors (i.e., events) and the Controller operation errors (i.e., events) that involve software that can occur in the PLC Function Module are given on the following pages. Event levels are given in the tables as follows:

Maj: Major fault level

Prt: Partial fault level

Min: Minor fault level

Obs: Observation

Info: Information

A version in parentheses in the *Event code* column is the unit version of the CPU Unit when the event occurs for only specific unit versions of the CPU Unit.

Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for all NJ/NX-series event codes.

Errors Related to Tasks

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
60020000 hex	Task Execution Timeout	Task execution exceeded the timeout detection time.	• The timeout detection time setting is too short. • The task period setting is too short. • A user program is too large. • The number of times that processing is repeated is larger than expected. • Task Priority Error • Frequent Event Task Execution	√					page 12-29
60030000 hex	I/O Refreshing Timeout Error	Consecutive I/O refresh failures occurred during the primary periodic task or periodic task period.	• The task period setting is too short. • Task Priority Error for Periodic Tasks and Event Tasks • There are too many Units and slaves that perform I/O refresh in the task period. • Frequent Event Task Execution	√					page 12-30
60040000 hex	Insufficient System Service Time Error	The specified system service execution time could not be obtained.	• There was not sufficient time to execute the tasks and tag data link service. • The system service execution interval is too short or the system service execution time ratio is too long in the System Service Monitoring Settings.	√					page 12-31
60010000 hex	Task Period Exceeded	Task execution was not completed during the set task period for the primary periodic task or a periodic task.	• The task period setting is too short. • A user program is too large. • The number of times that processing is repeated is larger than expected. • Task Priority Error for Periodic Tasks and Event Tasks • Frequent Event Task Execution			√			page 12-32

Event code	Event name	Meaning	Assumed cause	Level: Maj	Prt	Min	Obs	Info	Reference
60050000 hex	Task Period Exceeded	Task execution was not completed during the set task period for the primary periodic task or fixed periodic task.	• The task period setting is too short. • A user program is too large. • The number of times that processing is repeated is larger than expected. • Task Priority Error for Periodic Tasks and Event Tasks • Frequent Event Task Execution				√		page 12-33

Errors Related to Controller Operation

Event code	Event name	Meaning	Assumed cause	Level: Maj	Prt	Min	Obs	Info	Reference
10200000 hex	User Program/Controller Configurations and Setup Transfer Error	The user program or Controller Configurations and Setup were not transferred correctly.	• The user program or Controller Configurations and Setup are not correct because the power supply to the Controller was interrupted during a download of the user program or the Controller Configurations and Setup. • The user program or Controller Configurations and Setup are not correct because the power supply to the Controller was interrupted during online editing. • The user program or Controller Configurations and Setup are not correct because the power supply to the Controller was interrupted during a Clear All Memory operation. • The user program or Controller Configurations and Setup are not correct because the power supply to the Controller was interrupted during a restore operation. • Non-volatile memory failed.	√					page 12-34
10210000 hex	Illegal User Program Execution ID	The user program execution IDs set in the user program and in the CPU Unit do not match.	• The user program execution IDs set in the user program and in the CPU Unit do not match. • A user program execution ID is set in the CPU Unit but not in the user program.	√					page 12-35
10240000 hex	Illegal User Program	The user program is not correct.	• There are more than 8 nesting levels for functions or function blocks.	√					page 12-35
10250000 hex	Illegal User Program/Controller Configurations and Setup	The upper limit of the usable memory was exceeded or the user program or Controller Configurations and Setup is corrupted.	• The upper limit of the data size was exceeded. • The main memory capacity was exceeded. • Non-volatile memory is deteriorating or has failed.	√					page 12-36

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
10270000 hex (Ver. 1.03 or later)	Error in Starting Automatic Transfer	An error was detected in pre-execution checks for automatic transfer.	• An SD Memory Card is not inserted. • The SD Memory Card type is not correct. • The format of the SD Memory Card is not correct. • There is no autoload folder on the SD Memory Card. • There are no backup files in the autoload folder on the SD Memory Card. • Either the backup files in the autoload folder on the SD Memory Card are corrupted or required data is not in the backup files on the SD Memory Card. • The unit version of the CPU Unit to which to transfer the files is older than the unit version of the backup files on the SD Memory Card. • The model of the CPU Unit to which to transfer the files is not the same as the model of the CPU Unit of the backup files on the SD Memory Card. • Recovery was executed for the SD Memory Card. • The CPU Unit is write-protected. • The settings in the automatic transfer command file (AutoloadCommand.ini) are not correct. • Reading the data for automatic transfer failed because the SD Memory Card is faulty or not formatted correctly. • The SD Memory Card is damaged. • The database connection service version of the CPU Unit to which to transfer the files is older than the database connection service version of the backup files on the SD Memory Card. • The robot version of the CPU Unit to which to transfer the files is older than the robot version of the backup files on the SD Memory Card.	√					page 12-37

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
1028 0000 hex (Ver. 1.03 or later)	Error in Executing Automatic Transfer	The automatic transfer ended in an error.	• It was not possible to read the data for automatic transfer. • The SD Memory Card was removed during an automatic transfer. • There are no backup files in the autoload folder on the SD Memory Card. • The backup files in the autoload folder on the SD Memory Card are corrupted. • The SD Memory Card is damaged.	√					page 12-39
4011 0000 hex	PLC Function Processing Error	A fatal error was detected in the PLC Function Module.	• An error occurred in the software.	√					page 12-40
4016 0000 hex (Ver. 1.02 or earlier)	Safe Mode	The Controller started in Safe Mode.	• The power supply was turned ON to the Controller when Safe Mode was set on the DIP switch on the CPU Unit.	√					page 12-40
4442 0000 hex (Ver. 1.05 or later)	PLC Function Processing Error	A fatal error was detected in the PLC Function Module.	• An error occurred in the software.	√					page 12-41
4012 0000 hex	PLC Function Processing Error	A fatal error was detected in the PLC Function Module.	• An error occurred in the software.		√				page 12-41
4013 0000 hex	PLC Function Processing Error	A fatal error was detected in part of the PLC Function Module.	• An error occurred in the software.			√			page 12-42
1023 0000 hex	Event Log Save Error	Saving the event log failed.	• A low battery voltage prevented retention of memory during a power interruption.				√		page 12-42
1026 0000 hex	Trace Setting Transfer Failure	The power supply was interrupted while transferring the trace settings.	• The power supply was interrupted while transferring the trace settings.				√		page 12-43
1029 0000 hex (Ver. 1.03 or later)	Backup Failed to Start	An error was detected in pre-execution checks for a backup operation.	• An SD Memory Card is not inserted. • The SD Memory Card type is not correct. • The format of the SD Memory Card is not correct. • The SD Memory Card is write protected. • The *Prohibiting backing up data to the SD Memory Card* parameter is set to prohibit backing up data to an SD Memory Card. • Another backup operation is in progress. • Synchronization, online editing, or the Clear All Memory operation is in progress. • The backup was canceled by the user. • The online connection with the Sysmac Studio was disconnected. • The SD Memory Card is damaged.				√		page 12-43

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
102A0000 hex (Ver. 1.03 or later)	Backup Failed	The backup operation ended in an error.	• The capacity of the SD Memory Card is insufficient. • It was not possible to save the data that was specified for backup. • The SD Memory Card was removed during a backup operation. • Failed to back up Unit or slave. • The backup was canceled by the user. • Execution of the Save Cam Table instruction or changing the CPU Unit name is in progress. • The online connection with the Sysmac Studio was disconnected. • It was not possible to save the data that was specified for backup to the computer. • The SD Memory Card is damaged.				√		page 12-45

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
102B0000 hex (Ver. 1.03 or later)	Restore Operation Failed to Start	An error was detected in pre-execution checks for a restore operation.	• An SD Memory Card is not inserted. • The SD Memory Card type is not correct. • The format of the SD Memory Card is not correct. • There are no backup files on the SD Memory Card. • Either the backup files on the SD Memory Card are corrupted or required data is not in the backup files on the SD Memory Card. • The unit version of the CPU Unit to which to restore the files is older than the unit version of the backup files on the SD Memory Card. • The model of the CPU Unit to which to restore the files is not the same as the model of the CPU Unit of the backup files on the SD Memory Card. • Recovery was executed for the SD Memory Card. • The CPU Unit is write-protected. • The settings in the restore command file (RestoreCommand.ini) are not correct. • A backup operation is in progress. • Synchronization, online editing, or the Clear All Memory operation is in progress. • The online connection with the Sysmac Studio was disconnected. • Reading the data for restoration failed because the SD Memory Card is faulty or not formatted correctly. • The SD Memory Card is damaged. • The database connection service version of the CPU Unit to which to restore the files is older than the database connection service version of the backup files on the SD Memory Card. • The robot version of the CPU Unit to which to restore the files is older than the robot version of the backup files on the SD Memory Card.				√		page 12-46
102C0000 hex (Ver. 1.03 or later)	Restore Operation Failed	The restore operation ended in an error.	• It was not possible to read the data to restore. • The SD Memory Card was removed during a restore operation. • Failed to restore Unit or slave. • The SD Memory Card is damaged.				√		page 12-48

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
40140000 hex	PLC System Information	This event provides internal information from the PLC Function Module.	• This event provides internal information from the PLC Function Module. It is recorded to provide additional information for another event.				√		page 12-49
40170000 hex (Ver. 1.03 or later)	Safe Mode	The CPU Unit started in Safe Mode.	• The power supply was turned ON to the Controller when the Safe Mode was set on the DIP switch on the CPU Unit.				√		page 12-49
80230000 hex (Ver. 1.05 or later)	NX Message Communications Error	An error has occurred in message communications.	• The communications cable is broken. • The communications cable connector is disconnected. • The NX message communications load is high.				√		page 12-50
40150000 hex	PLC System Information	This event provides internal information from the PLC Function Module.	• This event provides internal information from the PLC Function Module. It is recorded to provide additional information for another event.					√	page 12-50
44430000 hex (Ver. 1.05 or later)	PLC System Information	This event provides internal information from the PLC Function Module.	• This event provides internal information from the PLC Function Module. It is recorded to provide additional information for another event.					√	page 12-51
90010000 hex	Clock Changed	The clock time was changed.	• The clock time was changed.					√	page 12-51
90020000 hex	Time Zone Changed	The time zone was changed.	• The time zone was changed.					√	page 12-51
90050000 hex (Ver. 1.10 or later)	User Program/Controller Configurations and Setup Download	The user program and the Controller configurations and setup were downloaded.	• The user program and the Controller configurations and setup were downloaded.					√	page 12-52
90070000 hex (Ver. 1.10 or later)	Online Edits Transferred	The user program was edited online.	• The user program was edited online and the edits were transferred to the Controller.					√	page 12-52
90080000 hex	Variable Changed to TRUE with Forced Refreshing	Changing a variable to TRUE with forced refreshing was specified.	• Changing a variable to TRUE with forced refreshing was specified by the user.					√	page 12-53
90090000 hex	Variable Changed to FALSE with Forced Refreshing	Changing a variable to FALSE with forced refreshing was specified.	• Changing a variable to FALSE with forced refreshing was specified by the user.					√	page 12-53
900A0000 hex	All Forced Refreshing Cleared	Clearing all forced refreshing values was specified.	• Clearing all forced refreshing values was specified by the user.					√	page 12-53
900B0000 hex	Memory All Cleared	All memory was cleared.	• A user with Administrator rights cleared all of the memory.					√	page 12-54
900C0000 hex	Event Log Cleared	The event log was cleared.	• The event log was cleared by the user.					√	page 12-54
900F0000 hex (Ver. 1.03 or later)	Automatic Transfer Completed	The automatic transfer was completed.	• The automatic transfer was completed.					√	page 12-54
90110000 hex	Power Turned ON	The power supply was turned ON.	• The power supply was turned ON.					√	page 12-55

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
90120000 hex	Power Interrupted	The power supply was interrupted.	• The power supply was interrupted.					√	page 12-55
90130000 hex	Operation Started	Operation was started.	• A command to start operation was received.					√	page 12-55
90140000 hex	Operation Stopped	Operation was stopped.	• A command to stop operation was received.					√	page 12-56
90150000 hex	Reset Executed	A reset was executed.	• A reset command was received.					√	page 12-56
90160000 hex	User Program Execution ID Write	The user program execution ID was set or changed in the CPU Unit.	• A user with Administrator rights changed the user program execution ID that is set in the CPU Unit.					√	page 12-56
90180000 hex	All Controller Errors Cleared	All current errors were cleared.	• All current errors were changed by the user.					√	page 12-57
90190000 hex	Forced Refreshing Cleared	Clearing a forced refreshing value was specified.	• Clearing a forced refreshing value was specified by the user.					√	page 12-57
901A0000 hex (Ver. 1.03 or later)	Backup Started	A backup operation was started.	• A backup operation was started.					√	page 12-57
901B0000 hex (Ver. 1.03 or later)	Backup Completed	The backup operation ended normally.	• The backup operation ended normally.					√	page 12-58
901C0000 hex (Ver. 1.03 or later)	Restore Operation Started	A restore operation started.	• A restore operation started.					√	page 12-58
901D0000 hex (Ver. 1.03 or later)	Restore Operation Completed	The restore operation ended normally.	• The restore operation ended normally.					√	page 12-59

Errors Related to FINS Communications

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
14010000 hex	CPU Bus Unit Setup Area Error	An error was detected in the memory check of the Setup Area for CPU Bus Units.	• The power supply to the Controller was interrupted or communications with the Sysmac Studio were disconnected while downloading the CPU Bus Unit Settings.			√			page 12-60
34100000 hex	IP Address Table Setting Error	The IP address table settings are incorrect.	• The IP address conversion method is set to the combined method or the IP address table method, but the IP address table settings are incorrect.			√			page 12-60
34130000 hex	FINS/TCP Connection Table Setting Error	The FINS/TCP connection table is incorrect.	• The power supply to the Controller was interrupted or communications with the Sysmac Studio were disconnected while downloading the FINS/TCP connection table.			√			page 12-61
34110000 hex	Unknown Destination Node	The send destination node is not known.	• The send destination node was not found when a FINS message was sent.				√		page 12-61
80100000 hex	Packet Discarded	One or more packets were discarded.	• A FINS response addressed to the CPU Unit was received. • The send designation Unit for the FINS response does not exist.				√		page 12-62
80110000 hex	Packet Discarded	One or more packets were discarded.	• An attempt was made to send a FINS response with over 2002 bytes. • An attempt was made to route a FINS response with over 2002 bytes. • Packet was received with a No Such Unit routing error. • Packet was received with a Routing Error routing error. • Packet was received with a Routing Table Not Registered routing error. • Packet was received with an Event Area Size Over Limit routing error. • There is insufficient space in the internal buffer. • FINS message routing failed because the communications load is too high.				√		page 12-63

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
80120000 hex	Packet Discarded	One or more packets were discarded.	• A FINS response was received with the destination network address (DNA) set to the local network and the destination node address (DA1) not set to the local node. • A FINS command or response was received with a hub network address specification for which the destination network address (DNA) was greater than or equal to 80 hex. • There is insufficient space in the internal buffer. • A FINS command that does not have the minimum command length was received. • A FINS command that exceeded the maximum command length was received. • Sending packets failed. • FINS message routing failed because the communications load is too high. Or a command that was addressed to the built-in EtherNet/IP port was received with the source network address (SNA) set to 0. • A FINS response that was addressed to the built-in EtherNet/IP port was received. • A FINS response or a command for which a response is not required was received when the routing tables were not registered. • A FINS response or a command for which a response is not required was received when there was an error in the routing tables. • A FINS response or a command for which a response is not required was received that exceeded the number of relay points. • Transmission is not possible because the destination address is not set in the routing tables. • Routing is not possible because the FINS node address setting in the Built-in EtherNet/IP Port Settings is set to 0 or 255.				√		page 12-63

12-2-4 Error Descriptions

This section describes the information that is given for individual errors.

Controller Error Descriptions

The items that are used to describe individual errors (events) are described in the following copy of an error table.

Event name	Gives the name of the error.			Event code	Gives the code of the error.	
Meaning	Gives a short description of the error.					
Source	Gives the source of the error.		Source details	Gives details on the source of the error.	Detection timing	Tells when the error is detected.
Error attributes	Level	Tells the level of influence on control.*1	Recovery	Gives the recovery method.*2	Log category	Tells which log the error is saved in.*3
Effects	User program	Tells what will happen to execution of the user program.*4	Operation	Provides special information on the operation that results from the error.		
Indicators	Gives the status of the built-in EtherNet/IP port and built-in EtherCAT port indicators. Indicator status is given only for errors in the EtherCAT Master Function Module and the EtherNet/IP Function Module.					
System-defined variables	Variable		Data type		Name	
	Lists the variable names, data types, and meanings for system-defined variables that provide direct error notification, that are directly affected by the error, or that contain settings that cause the error.					
Cause and correction	Assumed cause		Correction		Prevention	
	Lists the possible causes, corrections, and preventive measures for the error.					
Attached information	This is the attached information that is displayed by the Sysmac Studio or an HMI.*5					
Precautions/ Remarks	Provides precautions, restrictions, and supplemental information. If the user can set the event level, the event levels that can be set, the recovery method, operational information, and other information are also provided.					

*1 One of the following:
Major fault: Major fault level
Partial fault: Partial fault level
Minor fault: Minor fault level
Observation
Information

*2 One of the following:
Automatic recovery: Normal status is restored automatically when the cause of the error is removed.
Error reset: Normal status is restored when the error is reset after the cause of the error is removed.
Cycle the power supply: Normal status is restored when the power supply to the Controller is turned OFF and then back ON after the cause of the error is removed.
Controller reset: Normal status is restored when the Controller is reset after the cause of the error is removed.
Depends on cause: The recovery method depends on the cause of the error.

*3 One of the following:
System: System event log
Access: Access event log

*4 One of the following:
Continues: Execution of the user program will continue.
Stops: Execution of the user program stops.
Starts: Execution of the user program starts.

*5 Refer to the appendices of the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for the applicable range of the HMI Troubleshooter.

Errors Related to Tasks

Event name	Task Execution Timeout			Event code	60020000 hex	
Meaning	Task execution exceeded the timeout detection time.					
Source	PLC Function Module		Source details	None	Detection timing	Continuously
Error attributes	Level	Major fault	Recovery	Cycle the power supply or reset the Controller.	Log category	System
Effects	User program	Stops.	Operation	Stops.*		
System-defined variables	Variable		Data type		Name	
	_<*Task_name*>_Exceeded		BOOL		Task Period Exceeded Flag	
	_<*Task_name*>_ExceedCount		UDINT		Task Period Exceeded Count	
	_<*Task_name*>_LastExecTime		TIME		Last Task Execution Time	
	_<*Task_name*>_MaxExecTime		TIME		Maximum Task Execution Time	
Cause and correction	Assumed cause		Correction		Prevention	
	The timeout detection time setting is too short.		Increase the timeout detection time.		Design the tasks considering the corrections that are given on the left.	
	The task period setting is too short.		Increase the task period.			
	A user program is too large.		Separate the processes into different tasks, for example move processes that need a short execution period to a periodic task with a lower priority.			
	The number of times that processing is repeated is larger than expected.		If there is a program with an extremely high number of repetitions, correct the program to achieve the correct number of repetitions. Set a trap in the user program that monitors the number of times a process is executed to check the number of repetitions.			
	Task Priority Error		Increase the priority of the task. Or, decrease the priorities of the other tasks.			
	Frequent Event Task Execution		Lower the frequency of event task execution. Or, decrease the priorities of the event tasks.			
Attached information	Attached Information 1: Name of task where error occurred					
Precautions/ Remarks	None					

* For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

Event name	I/O Refreshing Timeout Error			Event code	60030000 hex	
Meaning	Consecutive I/O refresh failures occurred during the primary periodic task or periodic task period.					
Source	PLC Function Module		Source details	None	Detection timing	Continuously
Error attributes	Level	Major fault	Recovery	Cycle the power supply or reset the Controller.	Log category	System
Effects	User program	Stops.	Operation	Stops.*		
System-defined variables	Variable		Data type		Name	
	_<*Task_name*>_Exceeded		BOOL		Task Period Exceeded Flag	
	_<*Task_name*>_ExceedCount		UDINT		Task Period Exceeded Count	
	_<*Task_name*>_LastExecTime		TIME		Last Task Execution Time	
	_<*Task_name*>_MaxExecTime		TIME		Maximum Task Execution Time	
Cause and correction	Assumed cause		Correction		Prevention	
	The task period setting is too short.		Check the task execution time and change the task period to an appropriate value.		Design the tasks considering the corrections that are given on the left.	
	Task Priority Error for Periodic Tasks and Event Tasks		Increase the priorities of the periodic tasks. Or, decrease the priorities of the event tasks so that they are lower than the priorities of the periodic tasks.			
	There are too many Units and slaves that perform I/O refresh in the task period.		Move the I/O refresh processes to other tasks, for example move I/O refresh processes within the task to other tasks.			
	Frequent Event Task Execution		Lower the frequency of event task execution. Or, decrease the priorities of the event tasks.			
Attached information	Attached Information 1: Name of task where error occurred					
Precautions/ Remarks	None					

* For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

<table>
<tr><td>Event name</td><td colspan="3">Insufficient System Service Time Error</td><td>Event code</td><td colspan="2">60040000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">The specified system service execution time could not be obtained.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None</td><td>Detection timing</td><td>Continuously</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Major fault</td><td>Recovery</td><td>Cycle the power supply or reset the Controller.</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Stops.</td><td>Operation</td><td colspan="3">Stops.*</td></tr>
<tr><td rowspan="5">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_Exceeded</td><td colspan="2">BOOL</td><td colspan="2">Task Period Exceeded Flag</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_ExceedCount</td><td colspan="2">UDINT</td><td colspan="2">Task Period Exceeded Count</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_LastExecTime</td><td colspan="2">TIME</td><td colspan="2">Last Task Execution Time</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_MaxExecTime</td><td colspan="2">TIME</td><td colspan="2">Maximum Task Execution Time</td></tr>
<tr><td rowspan="3">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">There was not sufficient time to execute the tasks and tag data link service.</td><td colspan="2">Check the available time for task execution and the tag data link settings. Increase the task periods or the packet intervals (RPI) in the tag data link settings to obtain enough time for execution of the tasks and tag data link service.</td><td colspan="2" rowspan="2">Set the System Service Monitoring Settings according to the corrections that are given on the left.</td></tr>
<tr><td colspan="2">The system service execution interval is too short or the system service execution time ratio is too long in the System Service Monitoring Settings.</td><td colspan="2">Check the effect on the processes executed by the system services with this operation and increase the system service execution interval or reduce the system service execution time ratio.</td></tr>
<tr><td>Attached information</td><td colspan="6">None</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

* For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

<table>
<tr><td>Event name</td><td colspan="3">Task Period Exceeded</td><td>Event code</td><td colspan="2">60010000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">Task execution was not completed during the set task period for the primary periodic task or a periodic task.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None</td><td>Detection timing</td><td>Continuously</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Minor fault</td><td>Recovery</td><td>Error reset</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">If the task execution does not finish within the set task period, the I/O refresh operation will be as follows:
• CJ-series Units: No I/O refresh is executed.
When task execution is completed, I/O refreshing for the next period is executed.
• EtherCAT slaves: The same values are output as for the previous output refresh.
If the task execution does not finish within the set task period, overall control of the equipment may become impossible.</td></tr>
<tr><td rowspan="5">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_Exceeded</td><td colspan="2">BOOL</td><td colspan="2">Task Period Exceeded Flag</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_ExceedCount</td><td colspan="2">UDINT</td><td colspan="2">Task Period Exceeded Count</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_LastExecTime</td><td colspan="2">TIME</td><td colspan="2">Last Task Execution Time</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_MaxExecTime</td><td colspan="2">TIME</td><td colspan="2">Maximum Task Execution Time</td></tr>
<tr><td rowspan="6">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">The task period setting is too short.</td><td colspan="2">Check the task execution time and change the task period to an appropriate value.</td><td colspan="2" rowspan="5">Design the tasks considering the corrections that are given on the left.</td></tr>
<tr><td colspan="2">A user program is too large.</td><td colspan="2">Separate the processes into different tasks, for example move processes that need a short execution period to a periodic task with a lower priority.</td></tr>
<tr><td colspan="2">The number of times that processing is repeated is larger than expected.</td><td colspan="2">If there is a program with an extremely high number of repetitions, correct the program to achieve the correct number of repetitions. Set a trap in the user program that monitors the number of times a process is executed to check the number of repetitions.</td></tr>
<tr><td colspan="2">Task Priority Error for Periodic Tasks and Event Tasks</td><td colspan="2">Increase the priorities of the periodic tasks. Or, decrease the priorities of the event tasks so that they are lower than the priorities of the periodic tasks.</td></tr>
<tr><td colspan="2">Frequent Event Task Execution</td><td colspan="2">Lower the frequency of event task execution. Or, decrease the priorities of the event tasks.</td></tr>
<tr><td>Attached information</td><td colspan="6">Attached Information 1: Name of task where error occurred</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">You can change the level of the error to an observation in the task settings.</td></tr>
</table>

<table>
<tr><td>Event name</td><td colspan="3">Task Period Exceeded</td><td>Event code</td><td colspan="2">60050000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">Task execution was not completed during the set task period for the primary periodic task or fixed periodic task.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None</td><td>Detection timing</td><td>Continuously</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Observation</td><td>Recovery</td><td>---</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">If the task execution does not finish within the set task period, the I/O refresh operation will be as follows:
• CJ-series Units: No I/O refresh is executed. When task execution is completed, I/O refreshing for the next period is executed.
• EtherCAT slaves: The same values are output as for the previous output refresh.
If the task execution does not finish within the set task period, overall control of the equipment may become impossible.</td></tr>
<tr><td rowspan="5">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_Exceeded</td><td colspan="2">BOOL</td><td colspan="2">Task Period Exceeded Flag</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_ExceedCount</td><td colspan="2">UDINT</td><td colspan="2">Task Period Exceeded Count</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_LastExecTime</td><td colspan="2">TIME</td><td colspan="2">Last Task Execution Time</td></tr>
<tr><td colspan="2">_<<i>Task_name</i>>_MaxExecTime</td><td colspan="2">TIME</td><td colspan="2">Maximum Task Execution Time</td></tr>
<tr><td rowspan="6">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">The task period setting is too short.</td><td colspan="2">Check the task execution time and change the task period to an appropriate value.</td><td colspan="2" rowspan="5">Design the tasks considering the corrections that are given on the left.</td></tr>
<tr><td colspan="2">A user program is too large.</td><td colspan="2">Separate the processes into different tasks, for example move processes that does not need a short execution period to a periodic task with a lower priority.</td></tr>
<tr><td colspan="2">The number of times that processing is repeated is larger than expected.</td><td colspan="2">If there is a program with an extremely high number of repetitions, correct the program to achieve the correct number of repetitions. Set a trap in the user program that monitors the number of times a process is executed to check the number of repetitions.</td></tr>
<tr><td colspan="2">Task Priority Error for Periodic Tasks and Event Tasks</td><td colspan="2">Increase the priorities of the periodic tasks. Or, decrease the priorities of the event tasks so that they are lower than the priorities of the periodic tasks.</td></tr>
<tr><td colspan="2">Frequent Event Task Execution</td><td colspan="2">Lower the frequency of event task execution. Or, decrease the priorities of the event tasks.</td></tr>
<tr><td>Attached information</td><td colspan="6">Attached Information 1: Name of task where error occurred</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">This error can occur if you change the level of the error to an observation in the task settings.</td></tr>
</table>

Errors Related to Controller Operation

<table>
<tr><td>Event name</td><td colspan="3">User Program/Controller Configurations and Setup Transfer Error</td><td>Event code</td><td colspan="2">10200000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">The user program or Controller Configurations and Setup were not transferred correctly.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None or I/O bus master</td><td>Detection timing</td><td>At power ON or Controller reset</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Major fault</td><td>Recovery</td><td>Cycle the power supply or reset the Controller.</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Stops.</td><td>Operation</td><td colspan="3">Stops.*</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">None</td><td colspan="2">---</td><td colspan="2">---</td></tr>
<tr><td rowspan="6">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">The user program or Controller Configurations and Setup are not correct because the power supply to the Controller was interrupted during a download of the user program or the Controller Configurations and Setup.</td><td colspan="2" rowspan="4">Clear all of memory and then download the project from the Sysmac Studio. If attached information is registered, cycle the power supply to the Controller and then implement the above correction.
If you cannot perform a Clear All Memory operation from the Sysmac Studio, transfer the project to the Controller with a restore operation from an SD Memory Card.</td><td colspan="2">Do not turn OFF the power supply to the Controller during a download of the user program or the Controller Configurations and Setup.</td></tr>
<tr><td colspan="2">The user program or Controller Configurations and Setup are not correct because the power supply to the Controller was interrupted during online editing.</td><td colspan="2">Do not interrupt the power supply to the Controller during online editing.</td></tr>
<tr><td colspan="2">The user program or Controller Configurations and Setup are not correct because the power supply to the Controller was interrupted during a Clear All Memory operation.</td><td colspan="2">Do not interrupt the power supply to the Controller during a Clear All Memory operation.</td></tr>
<tr><td colspan="2">The user program or Controller Configurations and Setup are not correct because the power supply to the Controller was interrupted during a restore operation.</td><td colspan="2">Do not interrupt the power supply to the Controller during a restore operation.</td></tr>
<tr><td colspan="2">Non-volatile memory failed.</td><td colspan="2">If the error persists even after you make the above correction, replace the CPU Unit.</td><td colspan="2">None</td></tr>
<tr><td>Attached information</td><td colspan="6">Attached Information 1: Cause Details
None: Power was interrupted during a download, during online editing, or during restoration.
Downloading/Predownloading: For other causes, the timing of error occurrence (during download or during download preparations) is given.</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

* For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

Event name	Illegal User Program Execution ID			Event code	10210000 hex	
Meaning	The user program execution IDs set in the user program and in the CPU Unit do not match.					
Source	PLC Function Module		Source details	None	Detection timing	At user program download, power ON, or Controller reset
Error attributes	Level	Major fault	Recovery	Cycle the power supply or reset the Controller.	Log category	System
Effects	User program	Stops.	Operation	Stops.*		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The user program execution IDs set in the user program and in the CPU Unit do not match.		Set the same user program execution ID in the user program and CPU Unit.		Set the same user program execution ID in the user program and CPU Unit. Keep a record of the user program execution IDs set in the user program and in the CPU Unit. They are not displayed.	
	A user program execution ID is set in the CPU Unit but not in the user program.		If user program execution ID is not set in the user program, clear the user program execution ID set in the CPU Unit by clearing all memory in the CPU Unit.			
Attached information	None					
Precautions/ Remarks	None					

* For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

Event name	Illegal User Program			Event code	10240000 hex	
Meaning	The user program is not correct.					
Source	PLC Function Module		Source details	None	Detection timing	At download, power ON, or Controller reset
Error attributes	Level	Major fault	Recovery	Cycle the power supply or reset the Controller.	Log category	System
Effects	User program	Stops.	Operation	Stops.*		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	There are more than 8 nesting levels for functions or function blocks.		Find the location in the user program with more than 8 nesting levels for functions or function blocks and reduce the number of nesting levels to 8 or fewer. Then, download the user program again.		Write the user program so that there is never more than 8 nesting levels for functions or function blocks. Use the program check on the Sysmac Studio to confirm that there are not more than 8 nesting levels.	
Attached information	None					
Precautions/ Remarks	None					

* For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

<table>
<tr><td>Event name</td><td colspan="3">Illegal User Program/Controller Configurations and Setup</td><td>Event code</td><td colspan="2">10250000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">The upper limit of the usable memory was exceeded or the user program or Controller Configurations and Setup is corrupted.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None</td><td>Detection timing</td><td>At download, power ON, or Controller reset</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Major fault</td><td>Recovery</td><td>Cycle the power supply or reset the Controller.</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Stops.</td><td>Operation</td><td colspan="3">Stops.*</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">None</td><td colspan="2">---</td><td colspan="2">---</td></tr>
<tr><td rowspan="4">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">The upper limit of the data size was exceeded.</td><td colspan="2">If an event on restrictions on the number of items used occurred at the same time as this event, correct the user program and settings so that the number of items used is not exceeded and then download the data again.</td><td colspan="2" rowspan="3">None</td></tr>
<tr><td colspan="2">The main memory capacity was exceeded.</td><td colspan="2">If an event on restrictions on the number of items used did not occur at the same time as this event, perform the Clear All Memory operation, cycle the power supply, and then confirm that this event was cleared. If it was cleared, reduce the size of the project, e.g., by sharing programming, and then download the project again.</td></tr>
<tr><td colspan="2">Non-volatile memory is deteriorating or has failed.</td><td colspan="2">If this error persists even after you implement the above two corrections, replace the CPU Unit.</td></tr>
<tr><td>Attached information</td><td colspan="6">None</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

* For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

<table>
<tr><td>Event name</td><td colspan="3">Error in Starting Automatic Transfer</td><td>Event code</td><td>10270000 hex[*1]</td></tr>
<tr><td>Meaning</td><td colspan="5">An error was detected in pre-execution checks for automatic transfer.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None</td><td>Detection timing</td><td>At power ON</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Major fault</td><td>Recovery</td><td>Cycle the power supply or reset the Controller.</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Stops.</td><td>Operation</td><td colspan="3">Stops.[*2]</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">None</td><td colspan="2">---</td><td colspan="2">---</td></tr>
<tr><td rowspan="9">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">An SD Memory Card is not inserted.</td><td colspan="2">Insert an SD Memory Card.</td><td colspan="2">Insert an SD Memory Card.</td></tr>
<tr><td colspan="2">The SD Memory Card type is not correct.</td><td colspan="2">Replace the SD Memory Card with an SD or SDHC card.</td><td colspan="2">Use an SD or SDHC card.</td></tr>
<tr><td colspan="2">The format of the SD Memory Card is not correct.</td><td colspan="2">Format the SD Memory Card with the Sysmac Studio and then create an autoload folder and place the backup files in it.</td><td colspan="2" rowspan="4">Use a formatted SD Memory Card, create an autoload folder on the SD Memory Card, and place the backup files in it. Also, do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit.</td></tr>
<tr><td colspan="2">There is no autoload folder on the SD Memory Card.</td><td colspan="2">Create an autoload folder on the SD Memory Card and place the backup files in it.</td></tr>
<tr><td colspan="2">There are no backup files in the autoload folder on the SD Memory Card.</td><td colspan="2">Place the backup files in the autoload folder on the SD Memory Card.</td></tr>
<tr><td colspan="2">Either the backup files in the autoload folder on the SD Memory Card are corrupted or required data is not in the backup files on the SD Memory Card.</td><td colspan="2">Replace the backup files in the autoload folder on the SD Memory Card. If this error occurs again even after replacing the files, create the backup files again and place them in the autoload folder.</td></tr>
<tr><td colspan="2">The unit version of the CPU Unit to which to transfer the files is older than the unit version of the backup files on the SD Memory Card.</td><td colspan="2">Replace the CPU Unit with a CPU Unit that has a unit version that is the same as or newer than the unit version of the CPU that was used to create the backup files. Or, place the backup files with the correct unit version for the CPU Unit in the autoload folder.</td><td colspan="2">Make sure that the unit version of the CPU Unit and the unit version of the backup files are compatible.</td></tr>
</table>

Cause and correction	The model of the CPU Unit to which to transfer the files is not the same as the model of the CPU Unit of the backup files on the SD Memory Card.	Replace the CPU Unit with a CPU Unit that has the same model as the CPU Unit that was used to create the backup files. Or, place the backup files with the same model of CPU Unit in the autoload folder.	Make sure that the model of the CPU Unit is the same as the model of the CPU Unit that was used to create the backup files.
	Recovery was executed for the SD Memory Card.	If there are no backup files or no automatic transfer command file in the autoload folder, place the files in the folder again.	None
	The CPU Unit is write-protected.	If you use automatic transfers, select the *Do not use* Option for the *Write protection at startup* setting of the CPU Unit.	If you use automatic transfers, select the *Do not use* Option for the *Write protection at startup* setting of the CPU Unit.
	The settings in the automatic transfer command file (AutoloadCommand.ini) are not correct.	Make sure that the required files are set to “Yes” in the automatic transfer command file.	Make sure that the required files are set to “Yes” in the automatic transfer command file.
	Reading the data for automatic transfer failed because the SD Memory Card is faulty or not formatted correctly.	Perform the same corrective measures as for when the format of the SD Memory Card is not correct or the SD Memory Card is damaged.	Perform the same preventive measures as for the following events: SD Memory Card Invalid Format or Faulty SD Memory Card.
	The SD Memory Card is damaged.	If none of the above causes applies, replace the SD Memory Card.	Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Or, replace the SD Memory Card periodically according to the write life of the SD Memory Card.
	The database connection service version of the CPU Unit to which to transfer the files is older than the database connection service version of the backup files on the SD Memory Card.	Replace the CPU Unit with a CPU Unit that has a database connection service version that is the same as or newer than the database connection service version of the CPU Unit that was used to create the backup files. Or, place the backup files with the correct database connection service version for the CPU Unit in the autoload folder.	Make sure that the database connection service version of the CPU Unit and the database connection service version of the backup files are compatible.
	The robot version of the CPU Unit to which to transfer the files is older than the robot version of the backup files on the SD Memory Card.	Replace the CPU Unit with a CPU Unit that has a robot version that is the same as or newer than the robot version of the CPU Unit that was used to create the backup files. Or, place the backup files with the correct robot version for the CPU Unit in the autoload folder.	Make sure that the robot version of the CPU Unit and the robot version of the backup files are compatible.
Attached information	Attached Information 1: Error Details 0001 hex: An SD Memory Card is not inserted. 0002 hex: The SD Memory Card is faulty, the format of the SD Memory Card is not correct, or the SD Memory Card is not the correct type of card. 0004 hex: Recovery was executed for the SD Memory Card. 0101 hex: There is no autoload folder on the SD Memory Card. 0102 hex: There are no backup files in the autoload folder on the SD Memory Card. 0103 hex: The backup files are corrupted. 0104 hex: The contents of the automatic transfer command file are not correct. 0105 hex: The required transfer data is not in the backup file. 0201 hex: The unit version of the CPU Unit is old. 0202 hex: The model numbers of the CPU Unit are not the same. 0203 hex: The CPU Unit is write-protected. 0211 hex: The database connection service or robot version of the CPU Unit is old. 0301 hex: Reading data for automatic transfer failed or the SD Memory Card is faulty.		
Precautions/ Remarks	None		

*1 This event code occurs for unit version 1.03 or later of the CPU Unit.

*2 For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

<table>
<tr><td>Event name</td><td colspan="3">Error in Executing Automatic Transfer</td><td>Event code</td><td colspan="2">10280000 hex*1</td></tr>
<tr><td>Meaning</td><td colspan="6">The automatic transfer ended in an error.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None</td><td>Detection timing</td><td>At power ON</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Major fault</td><td>Recovery</td><td>Cycle the power supply or reset the Controller.</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Stops.</td><td>Operation</td><td colspan="3">Stops.*2</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">None</td><td colspan="2">---</td><td colspan="2">---</td></tr>
<tr><td rowspan="6">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">It was not possible to read the data for automatic transfer.</td><td colspan="2">Format the SD Memory Card with the Sysmac Studio and then create an autoload folder and place the backup files in it.</td><td colspan="2">Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Replace the SD Memory Card periodically according to the write life of the SD Memory Card.</td></tr>
<tr><td colspan="2">The SD Memory Card was removed during an automatic transfer.</td><td colspan="2">Insert an SD Memory Card that contains the backup files in an autoload folder, and then cycle the power supply to execute the automatic transfer again.</td><td colspan="2">Do not remove the SD Memory Card during the automatic transfer.</td></tr>
<tr><td colspan="2">There are no backup files in the autoload folder on the SD Memory Card.</td><td colspan="2">Create an autoload folder in the SD Memory Card and store the backup files in it.</td><td colspan="2" rowspan="2">Use a formatted SD Memory Card, create an autoload folder in the SD Memory Card, and store the backup files in the folder. Do not turn OFF the power supply or remove the SD Memory Card while the SD BUSY indicator is lit.</td></tr>
<tr><td colspan="2">The backup files in the autoload folder on the SD Memory Card are corrupted.</td><td colspan="2">Store the backup files in the autoload folder again. If this error occurs again even after replacing the files, create the backup files again and place them in the autoload folder.</td></tr>
<tr><td colspan="2">The SD Memory Card is damaged.</td><td colspan="2">If none of the above causes applies, replace the SD Memory Card.</td><td colspan="2">Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Replace the SD Memory Card periodically according to the write life of the SD Memory Card.</td></tr>
<tr><td>Attached information</td><td colspan="6">Attached Information 1: Error Details
0001 hex: The SD Memory Card was removed.
0102 hex: There are no backup files in the autoload folder on the SD Memory Card.
0103 hex: The backup files are corrupted.
0301 hex: Reading data for automatic transfer failed or the SD Memory Card is damaged.</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

*1 This event code occurs for unit version 1.03 or later of the CPU Unit.

*2 For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

Event name	PLC Function Processing Error			Event code	40110000 hex	
Meaning	A fatal error was detected in the PLC Function Module.					
Source	PLC Function Module		Source details	None	Detection timing	Continuously
Error attributes	Level	Major fault	Recovery	Cycle the power supply.	Log category	System
Effects	User program	Stops.	Operation	Stops.*		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	An error occurred in the software.		Contact your OMRON representative.		None	
Attached information	Attached information 1: System information Attached Information 2: System information Attached information 3: System information Attached information 4: System information					
Precautions/ Remarks	None					

* For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

Event name	Safe Mode			Event code	40160000 hex	
Meaning	The Controller started in Safe Mode.					
Source	PLC Function Module		Source details	None	Detection timing	At power ON or Controller reset
Error attributes	Level	Major fault	Recovery	Cycle the power supply or reset the Controller.	Log category	System
Effects	User program	Stops.	Operation	Stops.*		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply was turned ON to the Controller when Safe Mode was set on the DIP switch on the CPU Unit.		---		---	
Attached information	None					
Precautions/ Remarks	If the Controller is started when the CPU Unit is in Safe Mode, the user program is not executed even if the startup mode is set to RUN mode.					

* For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

Event name	PLC Function Processing Error			Event code	44420000 hex*1	
Meaning	A fatal error was detected in the PLC Function Module.					
Source	PLC Function Module		Source details	None	Detection timing	Continuously
Error attributes	Level	Major fault	Recovery	Cycle the power supply or reset the Controller.	Log category	System
Effects	User program	Stops.	Operation	Stops.*2		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	An error occurred in the software.		Contact your OMRON representative.		None	
Attached information	Attached information 1: System information Attached Information 2: System information Attached information 3: System information Attached information 4: System information					
Precautions/ Remarks	None					

*1 This event code occurs for unit version 1.05 or later of the CPU Unit.

*2 For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

Event name	PLC Function Processing Error			Event code	40120000 hex	
Meaning	A fatal error was detected in the PLC Function Module.					
Source	PLC Function Module		Source details	None	Detection timing	Continuously
Error attributes	Level	Partial fault	Recovery	Cycle the power supply.	Log category	System
Effects	User program	Stops.	Operation	Stops.*		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	An error occurred in the software.		Contact your OMRON representative.		None	
Attached information	Attached information 1: System information Attached Information 2: System information Attached information 3: System information Attached information 4: System information					
Precautions/ Remarks	None					

* Operation is the same as for a major fault level error. For details, refer to *I/O Operation for Major Fault Level Controller Errors* on page 12-10.

Event name	PLC Function Processing Error			Event code	40130000 hex	
Meaning	A fatal error was detected in part of the PLC Function Module.					
Source	PLC Function Module		Source details	None	Detection timing	Continuously
Error attributes	Level	Minor fault	Recovery	Cycle the power supply or reset the Controller.	Log category	System
Effects	User program	Continues.	Operation	Operation is not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	An error occurred in the software.		Contact your OMRON representative.		None	
Attached information	Attached information 1: System information Attached Information 2: System information Attached information 3: System information Attached information 4: System information					
Precautions/ Remarks	None					

Event name	Event Log Save Error			Event code	10230000 hex	
Meaning	Saving the event log failed.					
Source	PLC Function Module		Source details	None	Detection timing	At power ON or Controller reset
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Starts.	Operation	Not affected. However, part or all of the past event log cannot be read.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A low battery voltage prevented retention of memory during a power interruption.		Replace the Battery.		Replace the battery periodically.	
Attached information	Attached information 1: Error Details 0: Failure to save all categories of logs, 1: Failure to save system event log, 2: Failure to save access event log, 100: Failure to save user-defined event log					
Precautions/ Remarks	None					

Event name	Trace Setting Transfer Failure			Event code	10260000 hex	
Meaning	The power supply was interrupted while transferring the trace settings.					
Source	PLC Function Module		Source details	None	Detection timing	At power ON or Controller reset
Error attributes	Level	Observation	Recovery	Cycle the power supply or reset the Controller.	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply was interrupted while transferring the trace settings.		Transfer the trace settings again.		Do not interrupt the power supply while transferring the trace settings.	
Attached information	None					
Precautions/ Remarks	All trace settings are initialized when this error occurs.					

Event name	Backup Failed to Start			Event code	10290000 hex*	
Meaning	An error was detected in pre-execution checks for a backup operation.					
Source	PLC Function Module		Source details	None	Detection timing	When backup is specified by the user
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	

Cause and correction	Assumed cause	Correction	Prevention
	An SD Memory Card is not inserted.	Insert an SD Memory Card.	Insert an SD Memory Card.
	The SD Memory Card type is not correct.	Replace the SD Memory Card with an SD or SDHC card.	Use an SD or SDHC card.
	The format of the SD Memory Card is not correct.	Format the SD Memory Card with the Sysmac Studio.	Use a formatted SD Memory Card. Also, do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit.
	The SD Memory Card is write protected.	Remove write protection from the SD Memory Card.	Make sure that the SD Memory Card is not write protected.
	The *Prohibiting backing up data to the SD Memory Card* parameter is set to prohibit backing up data to an SD Memory Card.	Change the setting of the *Prohibiting backing up data to the SD Memory Card* parameter to enable backing up data to an SD Memory Card.	Set the *Prohibiting backing up data to the SD Memory Card* parameter to enable backing up data to an SD Memory Card.
	Another backup operation is in progress.	Wait for the other backup operation to end and then perform the backup operation again.	Do not attempt to perform other backup operation during a backup operation.
	Synchronization, online editing, or the Clear All Memory operation is in progress.	Wait for the synchronization, online editing, or the Clear All Memory operation to end and then perform the backup operation again.	Do not attempt to perform a backup operation during a synchronization, online editing, or the Clear All Memory operation.
	The backup was canceled by the user.	None	None
	The online connection with the Sysmac Studio was disconnected.	Check the cable connections. Go offline and then go back online and execute the backup again.	Check the cable to see if it is disconnected or broken. Make sure the cable is connected properly.
	The SD Memory Card is damaged.	If none of the above causes applies, replace the SD Memory Card.	Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Replace the SD Memory Card periodically according to the write life of the SD Memory Card.
Attached information	Attached information 1: Operation type 0101 hex: Controller to SD Memory Card for switch operation on front of CPU Unit 0102 hex: Controller to SD Memory Card for system variable operation 0103 hex: Controller to SD Memory Card for Sysmac Studio operation 0201 hex: Controller to computer Attached Information 2: Error Details 0001 hex: An SD Memory Card is not inserted. 0002 hex: The SD Memory Card is faulty, the format of the SD Memory Card is not correct, or the SD Memory Card is not the correct type of card. 0003 hex: The SD Memory Card is write protected. 0204 hex: SD Memory Card backup is prohibited. 0205 hex: Another backup operation is in progress. 0206 hex: Synchronization, online editing, or the Clear All Memory operation is in progress. 0207 hex: A prohibited character is used in the directory name that is specified in the system-defined variable. 0401 hex: The backup was canceled by the user. 0501 hex: The online connection with the Sysmac Studio was disconnected.		
Precautions/ Remarks	None		

* This event code occurs for unit version 1.03 or later of the CPU Unit.

Event name	Backup Failed			Event code	102A0000 hex*	
Meaning	The backup operation ended in an error.					
Source	PLC Function Module		Source details	None	Detection timing	During backup operation
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The capacity of the SD Memory Card is insufficient.		Replace the SD Memory Card for one with sufficient available space.		Use an SD Memory Card that has sufficient available space.	
	It was not possible to save the data that was specified for backup.		Perform the backup operation again when no data write operation to the CPU Unit is in progress.		Do not write to the CPU Unit when a backup operation is in progress.	
	The SD Memory Card was removed during a backup operation.		Insert an SD Memory Card.		Insert an SD Memory Card.	
	Failed to back up Unit or slave.		Refer to the corrections for the following events: CJ-series Unit Backup Failed (102D0000 hex) or EtherCAT Slave Backup Failed (102F0000 hex).		Refer to the prevention information for the following events: CJ-series Unit Backup Failed (102D0000 hex) or EtherCAT Slave Backup Failed (102F0000 hex).	
	The backup was canceled by the user.		None		None	
	Execution of the Save Cam Table instruction or changing the CPU Unit name is in progress.		Perform the operation after execution of the Save Cam Table instruction or changing the CPU Unit name is completed.		Do not perform a backup during execution of the Save Cam Table instruction or while changing the CPU Unit name.	
	The online connection with the Sysmac Studio was disconnected.		Check the cable connections. Go offline and then go back online and execute the backup again.		Check the cable to see if it is disconnected or broken. Make sure the cable is connected properly.	
	It was not possible to save the data that was specified for backup to the computer.		Increase the available space on the hard disk on the computer.		Make sure there is sufficient space available on the hard disk before you perform a backup.	
	The SD Memory Card is damaged.		If none of the above causes applies, replace the SD Memory Card.		Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Replace the SD Memory Card periodically according to the write life of the SD Memory Card.	
Attached information	Attached information 1: Operation type 0101 hex: Controller to SD Memory Card for switch operation on front of CPU Unit 0102 hex: Controller to SD Memory Card for system variable operation 0103 hex: Controller to SD Memory Card for Sysmac Studio operation 0201 hex: Controller to computer Attached Information 2: Error Details 0001 hex: The SD Memory Card was removed. 0005 hex: There is not sufficient space available on the SD Memory Card. 0206 hex: Execution of the Save Cam Table instruction or changing the CPU Unit name is in progress. 0302 hex: Saving the backup data failed or the SD Memory Card is faulty. 0304 hex: The Unit or slave could not be backed up. 0401 hex: The backup was canceled by the user. 0501 hex: The online connection with the Sysmac Studio was disconnected. 0502 hex: It was not possible to save the data that was specified for backup to the computer.					
Precautions/ Remarks	None					

* This event code occurs for unit version 1.03 or later of the CPU Unit.

<table>
<tr><td>Event name</td><td colspan="3">Restore Operation Failed to Start</td><td>Event code</td><td colspan="2">102B0000 hex*</td></tr>
<tr><td>Meaning</td><td colspan="6">An error was detected in pre-execution checks for a restore operation.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None</td><td>Detection timing</td><td>When restoring data is specified by the user</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Observation</td><td>Recovery</td><td>---</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>---</td><td>Operation</td><td colspan="3">Not affected.</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">None</td><td colspan="2">---</td><td colspan="2">---</td></tr>
<tr><td rowspan="16">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">An SD Memory Card is not inserted.</td><td colspan="2">Insert an SD Memory Card.</td><td colspan="2">Insert an SD Memory Card.</td></tr>
<tr><td colspan="2">The SD Memory Card type is not correct.</td><td colspan="2">Replace the SD Memory Card with an SD or SDHC card.</td><td colspan="2">Use an SD or SDHC card.</td></tr>
<tr><td colspan="2">The format of the SD Memory Card is not correct.</td><td colspan="2">Format the SD Memory Card with the Sysmac Studio and then place the backup files on it.</td><td colspan="2" rowspan="3">Use a formatted SD Memory Card and place the backup files on it. Also, do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit.</td></tr>
<tr><td colspan="2">There are no backup files on the SD Memory Card.</td><td colspan="2">Place the backup files in the specified folder on the SD Memory Card.</td></tr>
<tr><td colspan="2">Either the backup files on the SD Memory Card are corrupted or required data is not in the backup files on the SD Memory Card.</td><td colspan="2">Create the backup files again.</td></tr>
<tr><td colspan="2">The unit version of the CPU Unit to which to restore the files is older than the unit version of the backup files on the SD Memory Card.</td><td colspan="2">Replace the CPU Unit with a CPU Unit that has a unit version that is the same as or newer than the unit version of the CPU Unit that was used to create the backup files. Or, specify backup files with the correct unit version for the CPU Unit.</td><td colspan="2">Make sure that the unit version of the CPU Unit and the unit version of the backup files are compatible.</td></tr>
<tr><td colspan="2">The model of the CPU Unit to which to restore the files is not the same as the model of the CPU Unit of the backup files on the SD Memory Card.</td><td colspan="2">Replace the CPU Unit with a CPU Unit that has the same model as the CPU Unit that was used to create the backup files. Or, specify backup files with the correct model for the CPU Unit.</td><td colspan="2">Make sure that the model of the CPU Unit is the same as the model of the CPU Unit that was used to create the backup files.</td></tr>
<tr><td colspan="2">Recovery was executed for the SD Memory Card.</td><td colspan="2">If there are no backup files or no restore command file in the specified folder on the SD Memory Card, place the files in the folder again.</td><td colspan="2">None</td></tr>
<tr><td colspan="2">The CPU Unit is write-protected.</td><td colspan="2">If you use the restore function, select the Do not use Option for the Write protection at startup setting of the CPU Unit.</td><td colspan="2">If you use the restore function, select the Do not use Option for the Write protection at startup setting of the CPU Unit.</td></tr>
<tr><td colspan="2">The settings in the restore command file (RestoreCommand.ini) are not correct.</td><td colspan="2">Make sure that the required files are set to “Yes” in the restore command file.</td><td colspan="2">Make sure that the required files are set to “Yes” in the restore command file.</td></tr>
<tr><td colspan="2">A backup operation is in progress.</td><td colspan="2">Wait for the backup operation to end and then perform the restore operation again.</td><td colspan="2">Do not attempt to perform a restore operation during a backup operation.</td></tr>
<tr><td colspan="2">Synchronization, online editing, or the Clear All Memory operation is in progress.</td><td colspan="2">Wait for the synchronization, online editing, or the Clear All Memory operation to end and then perform the restore operation again.</td><td colspan="2">Do not attempt to perform a restore operation during a synchronization, online editing, or the Clear All Memory operation.</td></tr>
<tr><td colspan="2">The online connection with the Sysmac Studio was disconnected.</td><td colspan="2">Check the cable connections. Go offline and then go back online and execute the backup again.</td><td colspan="2">Check the cable to see if it is disconnected or broken. Make sure the cable is connected properly.</td></tr>
<tr><td colspan="2">Reading the data for restoration failed because the SD Memory Card is faulty or not formatted correctly.</td><td colspan="2">Perform the same corrective measures as for when the format of the SD Memory Card is not correct or the SD Memory Card is damaged.</td><td colspan="2">Perform the same preventive measures as for the following events: SD Memory Card Invalid Format or Faulty SD Memory Card.</td></tr>
</table>

Cause and correction	The SD Memory Card is damaged.	If none of the above causes applies, replace the SD Memory Card.	Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Replace the SD Memory Card periodically according to the write life of the SD Memory Card.
	The database connection service version of the CPU Unit to which to restore the files is older than the database connection service version of the backup files on the SD Memory Card.	Replace the CPU Unit with a CPU Unit that has a database connection service version that is the same as or newer than the database connection service version of the CPU Unit that was used to create the backup files. Or, specify backup files with the correct database connection service version for the CPU Unit.	Make sure that the database connection service version of the CPU Unit and the database connection service version of the backup files are compatible.
	The robot version of the CPU Unit to which to restore the files is older than the robot version of the backup files on the SD Memory Card.	Replace the CPU Unit with a CPU Unit that has a robot version that is the same as or newer than the robot version of the CPU Unit that was used to create the backup files. Or, specify backup files with the correct robot version for the CPU Unit.	Make sure that the robot version of the CPU Unit and the robot version of the backup files are compatible.
Attached information	Attached information 1: Operation type 0101 hex: SD Memory Card to Controller for switch operation on front of CPU Unit 0201 hex: Computer to Controller Attached Information 2: Error Details 0001 hex: An SD Memory Card is not inserted. 0002 hex: The SD Memory Card is faulty, the format of the SD Memory Card is not correct, or the SD Memory Card is not the correct type of card. 0004 hex: Recovery was executed for the SD Memory Card. 0102 hex: There are no backup files. 0103 hex: The backup files are corrupted. 0104 hex: The contents of the restore command file are not correct. 0105 hex: The required transfer data is not in the backup file. 0201 hex: The unit version of the CPU Unit is old. 0202 hex: The model numbers of the CPU Unit are not the same. 0203 hex: The CPU Unit is write-protected. 0205 hex: Another backup operation is in progress. 0206 hex: Synchronization, online editing, or the Clear All Memory operation is in progress. 0211 hex: The database connection service or robot version of the CPU Unit is old. 0301 hex: Reading data for restoration failed or the SD Memory Card is faulty. 0501 hex: The online connection with the Sysmac Studio was disconnected.		
Precautions/ Remarks	None		

* This event code occurs for unit version 1.03 or later of the CPU Unit.

Event name	Restore Operation Failed			Event code	102C0000 hex*	
Meaning	The restore operation ended in an error.					
Source	PLC Function Module		Source details	None	Detection timing	During restore operation
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	---	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	It was not possible to read the data to restore.		Format the SD Memory Card with the Sysmac Studio and then place the backup files on it.		Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Replace the SD Memory Card periodically according to the write life of the SD Memory Card.	
	The SD Memory Card was removed during a restore operation.		Insert an SD Memory Card that contains the backup files, and then execute the restore operation again.		Do not remove the SD Memory Card during the restore operation.	
	Failed to restore Unit or slave.		Refer to the corrections for the following events: CJ-series Unit Restore Operation Failed (102E0000 hex) or EtherCAT Slave Restore Operation Failed (10300000 hex).		Refer to the prevention information for the following events: CJ-series Unit Restore Operation Failed (102E0000 hex) or EtherCAT Slave Restore Operation Failed (10300000 hex).	
	The SD Memory Card is damaged.		If none of the above causes applies, replace the SD Memory Card.		Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Replace the SD Memory Card periodically according to the write life of the SD Memory Card.	
Attached information	Attached information 1: Operation type 0101 hex: SD Memory Card to Controller for switch operation on front of CPU Unit 0201 hex: Computer to Controller Attached Information 2: Error Details 0001 hex: The SD Memory Card was removed. 0102 hex: There are no backup files. 0103 hex: The backup files are corrupted. 0301 hex: Reading data for restoration failed or the SD Memory Card is faulty. 0303 hex: The Unit or slave could not be restored.					
Precautions/ Remarks	None					

* This event code occurs for unit version 1.03 or later of the CPU Unit.

Event name	PLC System Information			Event code	40140000 hex	
Meaning	This event provides internal information from the PLC Function Module.					
Source	PLC Function Module		Source details	None	Detection timing	Continuously
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Operation is not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	This event provides internal information from the PLC Function Module. It is recorded to provide additional information for another event.		---		---	
Attached information	Attached information 1: System information Attached Information 2: System information Attached information 3: System information Attached information 4: System information					
Precautions/ Remarks	None					

Event name	Safe Mode			Event code	40170000 hex*	
Meaning	The CPU Unit started in Safe Mode.					
Source	PLC Function Module		Source details	None	Detection timing	At power ON or Controller reset
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Stops.	Operation	---		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply was turned ON to the Controller when the Safe Mode was set on the DIP switch on the CPU Unit.		---		---	
Attached information	None					
Precautions/ Remarks	If the Controller is started when the CPU Unit is in Safe Mode, the CPU Unit will start in PROGRAM mode even if the startup mode is set to RUN mode.					

* This event code occurs for unit version 1.03 or later of the CPU Unit.

Event name	NX Message Communications Error			Event code	80230000 hex*	
Meaning	An error has occurred in message communications.					
Source	PLC Function Module, EtherCAT Master Function Module, or EtherNet/IP Function Module		Source details	None	Detection timing	During NX message communications
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	---		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The communications cable is broken.		Check the communications cable and replace it if it is broken.		Check the communications cable to see if it is operating properly.	
Attached information	The communications cable connector is disconnected.		Reconnect the connector and make sure it is mated correctly.		Make sure the communications cable is connected properly.	
	The NX message communications load is high.		Reduce the number of times that instructions are used to send NX messages. Or, increase the value of the *TimeOut* input variable to the instruction. If more than one copy of the Sysmac Studio is connected, reduce the frequency of simultaneous operations.		Reduce the number of times that instructions are used to send NX messages. Or, increase the value of the *TimeOut* input variable to the instruction. If more than one copy of the Sysmac Studio is connected, reduce the frequency of simultaneous operations.	
Precautions/ Remarks	Attached Information 1: System Information Attached Information 2: Type of Communications 0: NX bus 1: EtherCAT 65,535: Unit internal communications (routing)					

* This event code occurs for unit version 1.05 or later of the CPU Unit.

Event name	PLC System Information			Event code	40150000 hex	
Meaning	This event provides internal information from the PLC Function Module.					
Source	PLC Function Module		Source details	None	Detection timing	Continuously
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Operation is not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	This event provides internal information from the PLC Function Module. It is recorded to provide additional information for another event.		---		---	
Attached information	Attached information 1: System information Attached Information 2: System information Attached information 3: System information Attached information 4: System information					
Precautions/ Remarks	None					

Event name	PLC System Information			Event code	44430000 hex*	
Meaning	This event provides internal information from the PLC Function Module.					
Source	PLC Function Module		Source details	None	Detection timing	Continuously
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Operation is not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	This event provides internal information from the PLC Function Module. It is recorded to provide additional information for another event.		---		---	
Attached information	Attached information 1: System information Attached Information 2: System information Attached information 3: System information Attached information 4: System information					
Precautions/ Remarks	None					

* This event code occurs for unit version 1.05 or later of the CPU Unit.

Event name	Clock Changed			Event code	90010000 hex	
Meaning	The clock time was changed.					
Source	PLC Function Module		Source details	None	Detection timing	Commands from user
Error attributes	Level	Information	Recovery	---	Log category	Access
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	_CurrentTime		DATE_AND_TIME		System Time	
Cause and correction	Assumed cause		Correction		Prevention	
	The clock time was changed.		---		---	
Attached information	Attached information 1: Time before change					
Precautions/ Remarks	Clock changes by the Set Time instruction (SetTime) are not recorded in the event log. The time stamp for this event will be for the time after the change.					

Event name	Time Zone Changed			Event code	90020000 hex	
Meaning	The time zone was changed.					
Source	PLC Function Module		Source details	None	Detection timing	When downloading
Error attributes	Level	Information	Recovery	---	Log category	Access
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	_CurrentTime		DATE_AND_TIME		System Time	
Cause and correction	Assumed cause		Correction		Prevention	
	The time zone was changed.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	User Program/Controller Configurations and Setup Downloaded		Event code	90050000 hex*	
Meaning	The user program and the Controller configurations and setup were downloaded.				
Source	PLC Function Module	Source details	None	Detection timing	During user program/Controller configurations and setup download
Error attributes	Level: Information	Recovery	---	Log category	Access
Effects	User program: Continues.	Operation	Operation starts according to the user program and the Controller setup data that were downloaded.		
System-defined variables	Variable	Data type		Name	
	None	---		---	
Cause and correction	Assumed cause	Correction		Prevention	
	The user program and the Controller configurations and setup were downloaded.	---		---	
Attached information	Attached Information 1: Connection method 1: Direct USB connection 2: Direct Ethernet connection 3: Remote USB connection or Ethernet hub connection Attached Information 2: Connecting IP address, Connection through proxy: Proxy IP address (When attached information 1 is 2 or 3)				
Precautions/ Remarks	None				

* This event code occurs for unit version 1.10 or later of the CPU Unit.

Event name	Online Edits Transferred		Event code	90070000 hex*	
Meaning	The user program was edited online.				
Source	PLC Function Module	Source details	None	Detection timing	When transferring online edits is started
Error attributes	Level: Information	Recovery	---	Log category	Access
Effects	User program: Continues.	Operation	Operation is performed according to the changed user program.		
System-defined variables	Variable	Data type		Name	
	None	---		---	
Cause and correction	Assumed cause	Correction		Prevention	
	The user program was edited online and the edits were transferred to the Controller.	---		---	
Attached information	Attached Information 1: Connection method 1: Direct USB connection 2: Direct Ethernet connection 3: Remote USB connection or Ethernet hub connection Attached Information 2: Connecting IP address, Connection through proxy: Proxy IP address (When attached information 1 is 2 or 3)				
Precautions/ Remarks	None				

* This event code occurs for unit version 1.10 or later of the CPU Unit.

Event name	Variable Changed to TRUE with Forced Refreshing		Event code	90080000 hex	
Meaning	Changing a variable to TRUE with forced refreshing was specified.				
Source	PLC Function Module		Source details	None	Detection timing	Commands from user
Error attributes	Level	Information	Recovery	---	Log category	Access
Effects	User program	Continues.	Operation	Operation is performed according to the forced refreshing values.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	Changing a variable to TRUE with forced refreshing was specified by the user.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	Variable Changed to FALSE with Forced Refreshing		Event code	90090000 hex		
Meaning	Changing a variable to FALSE with forced refreshing was specified.					
Source	PLC Function Module		Source details	None	Detection timing	Commands from user
Error attributes	Level	Information	Recovery	---	Log category	Access
Effects	User program	Continues.	Operation	Operation is performed according to the forced refreshing values.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	Changing a variable to FALSE with forced refreshing was specified by the user.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	All Forced Refreshing Cleared		Event code	900A0000 hex		
Meaning	Clearing all forced refreshing values was specified.					
Source	PLC Function Module		Source details	None	Detection timing	Commands from user
Error attributes	Level	Information	Recovery	---	Log category	Access
Effects	User program	Continues.	Operation	Forced refreshing values are all cleared and operation is performed according to the user program.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	Clearing all forced refreshing values was specified by the user.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	Memory All Cleared		Event code	900B0000 hex	
Meaning	All of memory was cleared.				
Source	PLC Function Module	Source details	None	Detection timing	Commands from user
Error attributes	Level / Information	Recovery	---	Log category	Access
Effects	User program / ---	Operation	Operation returns to the factory state.		
System-defined variables	Variable	Data type		Name	
	None	---		---	
Cause and correction	Assumed cause	Correction		Prevention	
	A user with Administrator rights cleared all of the memory.	---		---	
Attached information	None				
Precautions/ Remarks	None				

Event name	Event Log Cleared		Event code	900C0000 hex	
Meaning	The event log was cleared.				
Source	PLC Function Module	Source details	None	Detection timing	Commands from user
Error attributes	Level / Information	Recovery	---	Log category	Access
Effects	User program / Continues.	Operation	Not affected.		
System-defined variables	Variable	Data type		Name	
	None	---		---	
Cause and correction	Assumed cause	Correction		Prevention	
	The event log was cleared by the user.	---		---	
Attached information	Attached information 1: Cleared events	0: All log categories were cleared 1: The system event log was cleared. 2: The access event log was cleared. 100: The user-defined event log was cleared.			
Precautions/ Remarks	None				

Event name	Automatic Transfer Completed		Event code	900F0000 hex*	
Meaning	The automatic transfer was completed.				
Source	PLC Function Module	Source details	None	Detection timing	At power ON
Error attributes	Level / Information	Recovery	---	Log category	System
Effects	User program / Continues.	Operation	Operation starts according to the Controller Configurations and Setup data that was automatically transferred.		
System-defined variables	Variable	Data type		Name	
	None	---		---	
Cause and correction	Assumed cause	Correction		Prevention	
	The automatic transfer was completed.	---		---	
Attached information	None				
Precautions/ Remarks	None				

* This event code occurs for unit version 1.03 or later of the CPU Unit.

Event name	Power Turned ON			Event code	90110000 hex	
Meaning	The power supply was turned ON.					
Source	PLC Function Module		Source details	None	Detection timing	At power ON
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	----	Operation	Operation starts.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply was turned ON.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	Power Interrupted			Event code	90120000 hex	
Meaning	The power supply was interrupted.					
Source	PLC Function Module		Source details	None	Detection timing	At power interruption
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Stops.	Operation	All operations stops.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply was interrupted.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	Operation Started			Event code	90130000 hex	
Meaning	Operation was started.					
Source	PLC Function Module		Source details	None	Detection timing	When changing to RUN mode
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Starts.	Operation	User program execution starts.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A command to start operation was received.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	Operation Stopped			Event code	90140000 hex	
Meaning	Operation was stopped.					
Source	PLC Function Module		Source details	None	Detection timing	When changing to PROGRAM mode
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Stops.	Operation	User program execution stops.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A command to stop operation was received.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	Reset Executed			Event code	90150000 hex	
Meaning	A reset was executed.					
Source	PLC Function Module		Source details	None	Detection timing	Commands from user
Error attributes	Level	Information	Recovery	---	Log category	Access
Effects	User program	---	Operation	Operation is started after a reset is executed.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A reset command was received.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	User Program Execution ID Write			Event code	90160000 hex	
Meaning	The user program execution ID was set or changed in the CPU Unit.					
Source	PLC Function Module		Source details	None	Detection timing	When download-ing
Error attributes	Level	Information	Recovery	---	Log category	Access
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A user with Administrator rights changed the user program execution ID that is set in the CPU Unit.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	All Controller Errors Cleared			Event code	90180000 hex
Meaning	All current errors were cleared.				
Source	PLC Function Module		Source details	None	Detection timing	Commands from user
Error attributes	Level	Information	Recovery	---	Log category	Access
Effects	User program	Continues.	Operation	Clearing all errors for which the causes have been removed.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The user cleared all current errors.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	Forced Refreshing Cleared				Event code	90190000 hex
Meaning	Clearing a forced refreshing value was specified.					
Source	PLC Function Module		Source details	None	Detection timing	Commands from user
Error attributes	Level	Information	Recovery	---	Log category	Access
Effects	User program	Continues.	Operation	Forced refreshing values are cleared and operation is performed according to the user program.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	Clearing a forced refreshing value was specified by the user.		---		---	
Attached information	None					
Precautions/ Remarks	None					

Event name	Backup Started				Event code	901A0000 hex*
Meaning	A backup operation was started.					
Source	PLC Function Module		Source details	None	Detection timing	At start of backup operation
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A backup operation was started.		---		---	
Attached information	Attached information 1: Operation type 0101 hex: Controller to SD Memory Card for switch operation on front of CPU Unit 0102 hex: Controller to SD Memory Card for system variable operation 0103 hex: Controller to SD Memory Card for Sysmac Studio operation 0201 hex: Controller to computer					
Precautions/ Remarks	None					

* This event code occurs for unit version 1.03 or later of the CPU Unit.

Event name	Backup Completed			Event code	901B0000 hex*	
Meaning	The backup operation ended normally.					
Source	PLC Function Module		Source details	None	Detection timing	At end of normal backup operation
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The backup operation ended normally.		---		---	
Attached information	Attached information 1: Operation type 0101 hex: Controller to SD Memory Card for switch operation on front of CPU Unit 0102 hex: Controller to SD Memory Card for system variable operation 0103 hex: Controller to SD Memory Card for Sysmac Studio operation 0201 hex: Controller to computer					
Precautions/ Remarks	None					

* This event code occurs for unit version 1.03 or later of the CPU Unit.

Event name	Restore Operation Started			Event code	901C0000 hex*	
Meaning	A restore operation started.					
Source	PLC Function Module		Source details	None	Detection timing	At start of restore operation
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	---	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A restore operation started.		---		---	
Attached information	Attached information 1: Operation type 0101 hex: SD Memory Card to Controller for switch operation on front of CPU Unit 0201 hex: Computer to Controller					
Precautions/ Remarks	None					

* This event code occurs for unit version 1.03 or later of the CPU Unit.

Event name	Restore Operation Completed			Event code	901D0000 hex*	
Meaning	The restore operation ended normally.					
Source	PLC Function Module		Source details	None	Detection timing	At end of normal restore operation
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	---	Operation	Operation does not start after the completion of a restore operation. To start operation according to the restored user program and settings, turn OFF the power supply to the Controller, turn OFF all pins on the DIP switch on the CPU Unit, and then turn ON the power supply again.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The restore operation ended normally.		---		---	
Attached information	Attached information 1: Operation type 0101 hex: SD Memory Card to Controller for switch operation on front of CPU Unit 0201 hex: Computer to Controller					
Precautions/ Remarks	None					

* This event code occurs for unit version 1.03 or later of the CPU Unit.

Errors Related to FINS Communications

Event name	CPU Bus Unit Setup Area Error			Event code	1401 0000 hex	
Meaning	An error was detected in the memory check of the Setup Area for CPU Bus Units.					
Source	PLC Function Module		Source details	None	Detection timing	At power ON, at Controller reset, or when writing CPU Bus Unit Setup Area
Error attributes	Level	Minor fault	Recovery	Error reset or cycling power supply	Log category	System
Effects	User program	Continues.	Operation	The CPU Bus Unit may stop.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply to the Controller was interrupted or communications with the Sysmac Studio were disconnected while downloading the CPU Bus Unit Settings.		Clear all memory or download the CPU Bus Unit Settings. If this error persists, replace the CPU Unit.		Do not interrupt the power supply to the Controller or disconnect communications with the Sysmac Studio while downloading the CPU Bus Unit Settings.	
Attached information	None					
Precautions/ Remarks	None					

Event name	IP Address Table Setting Error			Event code	3410 0000 hex	
Meaning	The IP address table settings are incorrect.					
Source	PLC Function Module		Source details	None	Detection timing	At power ON, Controller reset, or restart of built-in Ethernet port
Error attributes	Level	Minor fault	Recovery	Error reset	Log category	System
Effects	User program	Continues.	Operation	FINS/UDP communications will not operate.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The IP address conversion method is set to the combined method or the IP address table method, but the IP address table settings are incorrect.		Correct the IP address table settings.		Set the IP address table correctly.	
Attached information	None					
Precautions/ Remarks	None					

Event name	FINS/TCP Connection Table Setting Error			Event code	34130000 hex	
Meaning	The FINS/TCP connection table is incorrect.					
Source	PLC Function Module		Source details	None	Detection timing	At power ON, Controller reset, or restart of built-in Ethernet port
Error attributes	Level	Minor fault	Recovery	Error reset	Log category	System
Effects	User program	Continues.	Operation	FINS/UDP communications will not operate.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply to the Controller was interrupted or communications with the Sysmac Studio were disconnected while downloading the FINS/TCP connection table.		Download the FINS/TCP connection table again.		Do not interrupt the power supply to the Controller or disconnect communications with the Sysmac Studio while downloading the FINS/TCP connection table.	
Attached information	None					
Precautions/ Remarks	None					

Event name	Unknown Destination Node			Event code	34110000 hex	
Meaning	The send destination node is not known.					
Source	PLC Function Module		Source details	None	Detection timing	At FINS message reception
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Not affected. Packets are discarded.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The send destination node was not found when a FINS message was sent.		Correct the setting of the send destination node for FINS/UDP communications. Or, check the source FINS message and correct the destination node address.		Set the send destination node for FINS/UDP communications correctly. Or, make sure that the destination node address in the source FINS message is correct.	
Attached information	None					
Precautions/ Remarks	None					

Event name	Packet Discarded			Event code	80100000 hex	
Meaning	One or more packets were discarded.					
Source	PLC Function Module		Source details	None	Detection timing	At FINS message reception
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A FINS response addressed to the CPU Unit was received.		Correct the contents of the FINS message at the source.		Set the FINS messages correctly.	
	The send designation Unit for the FINS response does not exist.					
Attached information	Attached information 1: Cause of packet discard (01 hex: FINS response addressed to CPU Unit received, 02 hex: Response send failed)					
Precautions/ Remarks	None					

<table>
<tr><td>Event name</td><td colspan="3">Packet Discarded</td><td>Event code</td><td colspan="2">80110000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">One or more packets were discarded.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None</td><td>Detection timing</td><td>At FINS message reception</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Observation</td><td>Recovery</td><td>---</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">Not affected.</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">None</td><td colspan="2">---</td><td colspan="2">---</td></tr>
<tr><td rowspan="9">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">1. An attempt was made to send a FINS response with over 2002 bytes.</td><td colspan="2" rowspan="2">Do not send a FINS response with over 2002 bytes.</td><td colspan="2" rowspan="6">Set the FINS message at the source correctly.</td></tr>
<tr><td colspan="2">2. An attempt was made to route a FINS response with over 2002 bytes.</td></tr>
<tr><td colspan="2">3. Packet was received with a No Such Unit routing error.</td><td colspan="2">Check the FINS message at the source and correct the unit number in the response frame or a command frame that does not require a response.</td></tr>
<tr><td colspan="2">4. Packet was received with a Routing Error routing error.</td><td colspan="2">Check the FINS message at the source and correct the unit number in the response frame or a command frame that does not require a response to a unit number that is in the routing table.</td></tr>
<tr><td colspan="2">5. Packet was received with a Routing Table Not Registered routing error.</td><td colspan="2">Check the FINS message at the source, and correct the routing table include the network address of the destination network.</td></tr>
<tr><td colspan="2">6. Packet was received with an Event Area Size Over Limit routing error.</td><td colspan="2">Check the FINS message at the source and correct the event area size in the response frame or a command frame that does not require a response so that it does not exceed the limit.</td></tr>
<tr><td colspan="2">7. There is insufficient space in the internal buffer.</td><td colspan="2" rowspan="2">Reduce the frequency of sending FINS messages at the source.</td><td colspan="2" rowspan="2">Keep the frequency of sending FINS messages as low as possible.</td></tr>
<tr><td colspan="2">8. FINS message routing failed because the communications load is too high.</td></tr>
<tr><td>Attached information</td><td colspan="6">Attached information 1: Cause of discarding packets
1: 01 hex, 2: 02 hex, 3: 03 hex, 4: 04 hex, 5: 05 hex, 6: 06 hex, 7: 07 hex, 8: 08 hex
The numbers refer to the numbers of the above causes.</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

<table>
<tr><td>Event name</td><td colspan="3">Packet Discarded</td><td>Event code</td><td colspan="2">80120000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">One or more packets were discarded.</td></tr>
<tr><td>Source</td><td colspan="2">PLC Function Module</td><td>Source details</td><td>None</td><td>Detection timing</td><td>At FINS message reception</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Observation</td><td>Recovery</td><td>---</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">Not affected.</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">None</td><td colspan="2">---</td><td colspan="2">---</td></tr>
</table>

Cause and correction	Assumed cause	Correction	Prevention
	1. A FINS response was received with the destination network address (DNA) set to the local network and the destination node address (DA1) not set to the local node.	Correct the IP address table settings.	Make sure that the IP address table settings are correct.
	2. A FINS command or response was received with a hub network address specification for which the destination network address (DNA) was greater than or equal to 80 hex.	Correct the FINS message at the source so that the hub network address specification does not specify a destination network address (DNA) that is greater than or equal to 80 hex.	Make sure that the FINS message at the source does not have a hub network address specification that specifies a destination network address (DNA) that is greater than or equal to 80 hex.
	3. There is insufficient space in the internal buffer.	Reduce the frequency of sending FINS messages at the source.	Keep the frequency of sending FINS messages as low as possible.
	4. A FINS command that does not have the minimum command length was received.	Correct the FINS command at the source so that it has at least the minimum command length.	Set the FINS commands at the sources so that they have at least the minimum command length.
	5. A FINS command that exceeded the maximum command length was received.	Correct the FINS command at the source so that it does not exceed the maximum command length.	Set the FINS commands at the sources so that they do not exceed the maximum command length.
	6. Sending packets failed.	If the destination node is not in the network, add it to the network.	Confirm that the destination node is in the network.
	7. FINS message routing failed because the communications load is too high. Or a command that was addressed to the built-in EtherNet/IP port was received with the source network address (SNA) set to 0.	Reduce the frequency of sending FINS messages at the source. Or, correct the source network address (SNA) in the source FINS message.	Keep the frequency of sending FINS messages as low as possible. Or, set the correct source network address (SNA) in the source FINS message.
	8. A FINS response that was addressed to the built-in EtherNet/IP port was received.	Correct the contents of the FINS message at the source.	Set the FINS messages correctly.
	9. A FINS response or a command for which a response is not required was received when the routing tables were not registered.	Register the routing tables.	Register the routing tables.
	A: A FINS response or a command for which a response is not required was received when there was an error in the routing tables.	Register the routing tables again. If there is an error in the routing tables, there will be an Illegal User Program/Controller Configurations and Setup (10250000 hex) error.	None
	B: A FINS response or a command for which a response is not required was received that exceeded the number of relay points.	Increase the set value of the gateway counter in the routing table at the source.	Set the gateway counter in the routing table at the source so that it is suitable for the system configuration.
	C: Transmission is not possible because the destination address is not set in the routing tables.	Register the destination address in the routing tables.	Register the destination address in the routing tables.
	D: Routing is not possible because the FINS node address setting in the Built-in EtherNet/IP Port Settings is set to 0 or 255.	Set the FINS node address in the Built-in EtherNet/IP Port Settings to any value other than 0 or 255 from the Sysmac Studio.	If you set the lower eight bits of the IP address in the TCP/IP Settings in the Built-in EtherNet/IP Port Settings to 0 or 255, set the IP Address-FINS Address Conversion Method to any setting other than Automatic Generation.
Attached information	Attached Information 1: Cause of Discarding Packets 1: 01 hex, 2: 02 hex, 3: 03 hex, 4: 04 hex, 5: 05 hex, 6: 06 hex, 7: 07 hex, 8: 08 hex, 9: 09 hex, A: 0A hex, B: 0B hex, C: 0C hex, D: 0D hex The numbers correspond to the above cause numbers.		
Precautions/ Remarks	None		

12-2-5 Troubleshooting Errors That Are Not in the CPU Unit

Security Errors

No.	Problem	Correction
1	Forgot the Administrator password.	You cannot access the Administrator's password. Always record the Administrator password so that you do not forget it.
2	Cannot release the operation lock with the Sysmac Studio.	Log in with verification authority that is equal to or higher than the verification rights when you connected online.
3	Operation was locked when verifying operation authority on the Sysmac Studio.	If the password for verification of operation authority is entered incorrectly five time in row, operation is locked for 10 minutes. Wait until the operation lock is released.
4	An online connection was made with the operation authority that is required for operation, but operation authority verification was requested for a specific operation.	Verification of operation authority is required every time for the following functions to prevent hazards to equipment and people. • I/O monitoring (writing) by an Operator • Operating mode change by a Maintainer • Online editing by a Maintainer
5	Cannot release the operation lock with the Sysmac Studio after the operator left the Sysmac Studio unattended.	You can release the operation lock with an operation authority that is equal to or higher than the operator. The required operation authority will be that of an operator (the operation authority that was verified when going online with the Sysmac Studio).
6	Some of the user program data cannot be read for certain operations. • Monitoring Variables • Operation Commands SET/RESET, forced refreshing, online editing, data tracing, MC Test Run, and setting the user program execution ID in the CPU Unit • Synchronizing, Uploading, Verification, and Backup POU algorithms	The source data was not downloaded along with the user program. You will be able to read the data if you download the user program normally.
7	Writing to the CPU Unit is not possible for some operations. • Names CPU Unit name • Operation Commands Online editing, Clear All Memory, event log clearing, and setting the user program execution ID in the CPU Unit • Synchronizing and Downloading User program, CPU/Expansion Rack Configuration and Setup, EtherCAT Settings, Controller Setup, Axis Settings, Cam Table Settings, Data Trace Settings, User-defined Event Setup, restoring	The CPU Unit is write protected. Release the write protection.

No.	Problem	Correction
8	I do not know how to change the user program execution ID.	The user program execution ID cannot be changed or deleted after it is set.
9	I forgot the user program execution ID assigned to user program.	There is no way to access the user program execution ID that is set. Always record the user program execution ID so that you do not forget it.
10	I forgot the user program execution ID that is registered in the CPU Unit.	This is no way to access the user program execution ID that is set. Set the user program execution ID again. You can also clear the user program execution ID if you execute the Clear All Memory operation.

A

Appendices

The appendices provide the CPU Unit specifications, real processing times of tasks, system-defined variable lists, and other supplemental information for the body of this manual.

A-1 Specifications

This section gives the specifications of the NJ/NX-series Controllers.

A-1-1 General Specifications

Item		NX701-□□□□	NJ501-□□□□ NJ301-□□□□ NJ101-□□□□
Enclosure		Mounted in a panel	
Grounding method		Ground to less than 100 Ω.	
Dimensions (height × depth × width)		100 mm × 100 mm × 132 mm	90 mm × 90 mm × 90 mm
Weight		880 g (including the End Cover)	550 g (including the End Cover)
Power consumption or current consumption		40 W (including SD Memory Card and End Cover)	5 VDC 1.90 A (including SD Memory Card and End Cover)
Operating environment	Ambient operating temperature	0 to 55°C	
	Ambient operating humidity	10% to 95% (with no condensation)	10% to 90% (with no condensation)
	Atmosphere	Must be free from corrosive gases.	
	Ambient storage temperature	–25 to 70°C (excluding battery and Fan Unit)	–20 to 75°C (excluding battery)
	Altitude	2,000 m max.	
	Pollution degree	2 or less: Conforms to JIS B 3502 and IEC 61131-2.	
	Noise immunity	2 kV on power supply line (Conforms to IEC 61000-4-4.)	
	Overvoltage category	Category II: Conforms to JIS B 3502 and IEC 61131-2.	
	EMC immunity level	Zone B	
	Vibration resistance	Conforms to JIS C 60068-2-6. 5 to 8.4 Hz with 3.5-mm amplitude, 8.4 to 150 Hz, acceleration of 9.8 m/s^2, 100 min in X, Y, and Z directions (10 sweeps of 10 min each = 100 min total)	
	Shock resistance	Conforms to JIS C 60068-2-27. 147 m/s^2, 3 times in X, Y, and Z directions	Conforms to JIS C 60068-2-27. 147 m/s^2, 3 times in X, Y, and Z directions (100 m/s^2 for Relay Output Units)
Battery	Life*1	2.5 years at 25°C	5 years at 25°C
	Model	CJ1W-BAT01	
Applicable standards*2		cULus, EC Directives and KC	cULus, EC Directives, NK, LR and KC*3

*1 This is the value when the power-ON time rate is 0% (power OFF).
*2 Refer to the OMRON website (http://www.ia.omron.com/) or contact your OMRON representative for the most recent applicable standards for each model.
*3 The KC complies with a CPU Unit with a unit version of 1.01 or later.

A-1-2 Performance Specifications

Item				NX701-		NJ501-			NJ301-		NJ101-	
				17□□	16□□	15□□	14□□	13□□	12□□	11□□	10□□	90□□
Processing time	Instruction execution times	LD instruction		0.37 ns or more		1.2 ns (1.9 ns or less)			2.0 ns (3.0 ns or less)		3.3 ns (5.0 ns or less)	
		Math instructions (for long real data)		3.2 ns or more		26 ns or more			42 ns or more		70 ns or more	
Programming	Program capacity*1	Size		80 MB		20 MB			5 MB		3 MB	
		Quantity	Number of POU definitions	6,000		3,000			750		450	
			Number of POU Instances	48,000		9,000 *			3,000 *		1,800	
	Memory capacity for variables	Retain attributes*2	Size	4 MB		2 MB			0.5 MB			
			Number of variables	40,000		10,000			5,000 *			
		No Retain attributes*3	Size	256 MB		4 MB			2 MB			
			Number of variables	360,000		90,000			22,500			
	Data types	Number of data types		8,000		2,000			1,000			
	Memory for CJ-series Units (Can be specified with AT specifications for variables.)	CIO Area		---		6,144 words (CIO 0 to CIO 6143)						
		Work Area		---		512 words (W0 to W511)						
		Holding Area		---		1,536 words (H0 to H1535)						
		DM Area		---		32,768 words (D0 to D32767)						
		EM Area		---		32,768 words × 25 banks (E0_00000 to E18_32767)			32,768 words × 4 banks (E0_00000 to E3_32767)			
Motion control	Number of controlled axes	Maximum number of controlled axes*4		256 axes	128 axes	64 axes	32 axes	16 axes	15 axes (*)		6 axes	---
		Maximum number of used real axes*5		256 axes	128 axes	64 axes	32 axes	16 axes	8 axes	4 axes	2 axes	---
		Maximum number of axes for single-axis control		256 axes for single-axis control	128 axes for single-axis control	64 axes for single-axis control	32 axes for single-axis control	16 axes for single-axis control	15 axes for single-axis control (*)		6 axes	---
		Maximum number of axes for linear interpolation axis control		4 axes per axes group								---
		Number of axes for circular interpolation axis control		2 axes per axes group								---
	Maximum number of axes groups			64 axes groups		32 axes groups						---
	Motion control period			The same control period as that is used for the process data communications cycle for EtherCAT.								---
	Cams	Number of cam data points	Maximum points per cam table	65,535 points								---
			Maximum points for all cam tables	1,048,560 points					262,140 points			---
		Maximum number of cam tables		640 tables					160 tables			---
	Position units			Pulse, mm, μm, nm, degree, and inch								---
	Override factors			0.00%, or 0.01% to 500.00%								---
Peripheral USB port	Supported services			Sysmach Studio connection								
	Physical layer			USB 2.0-compliant B-type connector								
	Transmission distance			5 m max.								

<table>
<tr><th colspan="4" rowspan="2">Item</th><th colspan="2">NX701-</th><th colspan="3">NJ501-</th><th colspan="2">NJ301-</th><th colspan="2">NJ101-</th></tr>
<tr><th>17□□</th><th>16□□</th><th>15□□</th><th>14□□</th><th>13□□</th><th>12□□</th><th>11□□</th><th>10□□</th><th>90□□</th></tr>
<tr><td rowspan="26">Built-in EtherNet/IP port</td><td colspan="3">Number of ports</td><td colspan="2">2</td><td colspan="7">1</td></tr>
<tr><td colspan="3">Physical layer</td><td colspan="2">10BASE-T/100BASE-TX/1000BASE-T</td><td colspan="7">10BASE-T/100BASE-TX</td></tr>
<tr><td colspan="3">Frame length</td><td colspan="9">1,514 bytes max.</td></tr>
<tr><td colspan="3">Media access method</td><td colspan="9">CSMA/CD</td></tr>
<tr><td colspan="3">Modulation</td><td colspan="9">Baseband</td></tr>
<tr><td colspan="3">Topology</td><td colspan="9">Star</td></tr>
<tr><td colspan="3">Baud rate</td><td colspan="2">1 Gbps (1000BASE-T)</td><td colspan="7">100 Mbps (100BASE-TX)</td></tr>
<tr><td colspan="3">Transmission media</td><td colspan="9">STP (shielded, twisted-pair) cable of Ethernet category 5, 5e or higher</td></tr>
<tr><td colspan="3">Maximum transmission distance between Ethernet switch and node</td><td colspan="9">100 m</td></tr>
<tr><td colspan="3">Maximum number of cascade connections</td><td colspan="9">There are no restrictions if an Ethernet switch is used.</td></tr>
<tr><td rowspan="12">CIP service: Tag data links (cyclic communications)</td><td colspan="2">Maximum number of connections</td><td colspan="2">256 per port
512 total</td><td colspan="7">32</td></tr>
<tr><td colspan="2">Packet interval*6</td><td colspan="2">Can be set for each connection.
0.5 to 10,000 ms in 0.5-ms increments</td><td colspan="7">1 to 10,000 ms in 1.0-ms increments (*)
Can be set for each connection.</td></tr>
<tr><td colspan="2">Permissible communications band</td><td colspan="2">40,000 pps*7 (including heartbeat)</td><td colspan="7">3,000 pps*7 (including heartbeat) (*)</td></tr>
<tr><td colspan="2">Maximum number of tag sets</td><td colspan="2">256 per port
512 total</td><td colspan="7">32</td></tr>
<tr><td colspan="2">Tag types</td><td colspan="2">Network variables</td><td colspan="7">Network variables, CIO, Work, Holding, DM, and EM Areas</td></tr>
<tr><td colspan="2">Number of tags per connection (i.e., per tag set)</td><td colspan="9">8 (7 tags if Controller status is included in the tag set.)</td></tr>
<tr><td colspan="2">Maximum number of tags</td><td colspan="2">256 per port
512 total</td><td colspan="7">256</td></tr>
<tr><td colspan="2">Maximum link data size per node (total size for all tags)</td><td colspan="2">369,664 bytes</td><td colspan="7">19,200 bytes</td></tr>
<tr><td colspan="2">Maximum data size per connection</td><td colspan="2">1,444 bytes</td><td colspan="7">600 bytes</td></tr>
<tr><td colspan="2">Maximum number of registrable tag sets</td><td colspan="2">256 per port
512 total
(1 connection = 1 tag set)</td><td colspan="7">32 (1 connection = 1 tag set)</td></tr>
<tr><td colspan="2">Maximum tag set size</td><td colspan="2">1,444 bytes (Two bytes are used if Controller status is included in the tag set.)</td><td colspan="7">600 bytes (Two bytes are used if Controller status is included in the tag set.)</td></tr>
<tr><td colspan="2">Multi-cast packet filter*8</td><td colspan="9">Supported</td></tr>
<tr><td rowspan="3">CIP message service: Explicit messages</td><td colspan="2">Class 3 (number of connections)</td><td colspan="2">128 per port
256 total
(clients plus server)</td><td colspan="7">32 (clients plus server)</td></tr>
<tr><td rowspan="2">UCMM (non-connection type)</td><td>Maximum number of clients that can communicate at one time</td><td colspan="9">32 per port
64 total</td></tr>
<tr><td>Maximum number of servers that can communicate at one time</td><td colspan="9">32 per port
64 total</td></tr>
<tr><td colspan="3">Number of TCP sockets</td><td colspan="9">30 (*)</td></tr>
</table>

Item				NX701-		NJ501-			NJ301-		NJ101-	
				17□□	16□□	15□□	14□□	13□□	12□□	11□□	10□□	90□□
Built-in EtherCAT port	Communications standard			IEC 61158 Type12								
	EtherCAT master specifications			Class B (Feature Pack Motion Control compliant)								
	Physical layer			100BASE-TX								
	Modulation			Baseband								
	Baud rate			100 Mbps (100BASE-TX)								
	Duplex mode			Auto								
	Topology			Line, daisy chain, and branching								
	Transmission media			Twisted-pair cable of category 5 or higher (double-shielded straight cable with aluminum tape and braiding)								
	Maximum transmission distance between nodes			100 m								
	Maximum number of slaves			512		192					64	
	Range of node addresses that can be set			1 to 512		1 to 192						
	Maximum process data size			Inputs: 11,472 bytes Outputs: 11,472 bytes However, the maximum number of process data frames is 8.		Inputs: 5,736 bytes Outputs: 5,736 bytes However, the maximum number of process data frames is 4.						
	Maximum process data size per slave			Inputs: 1,434 bytes Outputs: 1,434 bytes								
	Communications cycle			• Primary periodic task 125 µs 250 µs to 8 ms (in 250-µs increments) • Priority-5 periodic task 125 µs 250 µs to 100 ms (in 250-µs increments)		500, 1,000, 2,000, or 4,000 µs (*)					1,000, 2,000, or 4,000 µs	
	Sync jitter			1 µs max.								
Unit configuration	Maximum number of connectable Units	Maximum number of CJ Units per CPU Rack or Expansion Rack		---		10						
		Maximum number of CJ Units for entire controller		---		40						
		Maximum number of NX Units for entire controller		4,000 (On EtherCAT Slave Terminals)								
	Maximum number of Expansion Racks			0		3						
	I/O capacity	Maximum number of I/O points on CJ-series Units		---		2,560						
	Power Supply Unit for CPU Rack and Expansion Racks	Model		NX-PA9001 NX-PD7001		NJ-P□3001						
		Power OFF detection time	AC power supply	30 to 45 ms								
			DC power supply	5 to 20 ms		22 to 25 ms						
Internal clock				At ambient temperature of 55°C: –3.5 to +0.5 min error per month At ambient temperature of 25°C: –1.5 to +1.5 min error per month At ambient temperature of 0°C: –3 to +1 min error per month								

*1 Execution objects and variable tables (including variable names)
*2 Does not include Holding, DM, and EM Area memory for CJ-series Units.
*3 Does not include CIO and Work Area memory for CJ-series Units.
*4 This is the total for all axis types.
*5 This is the total number of axes that are set as servo axes or encoder axes and are also set as used axes.
*6 Data will be refreshed at the set interval, regardless of the number of nodes.
*7 “pps” means packets per second, i.e., the number of communications packets that can be sent or received in one second.
*8 As the EtherNet/IP port implements the IGMP client, unnecessary multi-cast packets can be filtered by using an Ethernet switch that supports IGMP Snooping.

Note Items that are marked with asterisks in the table are improvements that were made during version upgrades. Refer to *A-13 Version Information for NX-series Controllers* and *A-14 Version Information for NJ-series Controllers* for information on version upgrades.

A-1-3 Function Specifications

Item				NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Tasks	Function			I/O refresh and the user program are executed in units that are called tasks. Tasks are used to specify execution conditions and execution priority.			
		Periodically executed tasks	Maximum number of primary periodic tasks	1			
			Maximum number of periodic tasks	4	3		
		Conditionally executed tasks (*)	Maximum number of event tasks	32			
			Execution conditions	When Activate Event Task instruction is executed or when condition expression for variable is met			
	Setup	System Service Monitoring Settings		---	The execution interval and the percentage of the total user program execution time are monitored for the system services (processes that are executed by the CPU Unit separate from task execution).		
Programming	POUs (program organization units)	Programs		POUs that are assigned to tasks.			
		Function blocks		POUs that are used to create objects with specific conditions.			
		Functions		POUs that are used to create an object that determine unique outputs for the inputs, such as for data processing.			
	Programming languages	Types		Ladder diagrams*1 and structured text (ST)			
	Namespaces (*)			Namespaces are used to create named groups of POU definitions.			
	Variables	External access of variables	Network variables	The function which allows access from the HMI, host computers, or other Controllers			
	Data types	Basic data types	Boolean	BOOL			
			Bit strings	BYTE, WORD, DWORD, and LWORD			
			Integers	INT, SINT, DINT, LINT, UINT, USINT, UDINT, and ULINT			
			Real numbers	REAL and LREAL			
			Durations	TIME			
			Dates	DATE			
			Times of day	TIME_OF_DAY			
			Dates and times	DATE_AND_TIME			
			Text strings	STRING			
		Derivative data types		Structures, Unions, and Enumerations			

<table>
<tr><th colspan="4">Item</th><th>NX701-□□□□</th><th>NJ501-□□□□</th><th>NJ301-□□□□</th><th>NJ101-□□□□</th></tr>
<tr><td rowspan="16">Pro-gram-ming</td><td rowspan="10">Data types</td><td rowspan="5">Struc-tures</td><td>Function</td><td colspan="4">A derivative data type that groups together data with different data types.</td></tr>
<tr><td>Maximum number of members</td><td colspan="4">2,048</td></tr>
<tr><td>Nesting maximum levels</td><td colspan="4">8</td></tr>
<tr><td>Member data types</td><td colspan="4">Basic data types, structures, unions, enumerations, or array variables</td></tr>
<tr><td>Specifying member offsets (*)</td><td colspan="4">You can use member offsets to place structure members at any memory locations.</td></tr>
<tr><td rowspan="3">Union</td><td>Function</td><td colspan="4">A derivative data type that enables access to the same data with different data types.</td></tr>
<tr><td>Maximum number of members</td><td colspan="4">4</td></tr>
<tr><td>Member data types</td><td colspan="4">BOOL, BYTE, WORD, DWORD, and LWORD</td></tr>
<tr><td>Enumeration</td><td>Function</td><td colspan="4">A derivative data type that uses text strings called enumerators to express variable values.</td></tr>
<tr><td rowspan="5">Data type attributes</td><td rowspan="4">Array specifi-cations</td><td>Function</td><td colspan="4">An array is a group of elements with the same data type. You specify the number (subscript) of the element from the first element to specify the element.</td></tr>
<tr><td>Maximum number of dimensions</td><td colspan="4">3</td></tr>
<tr><td>Maximum number of elements</td><td colspan="4">65,535</td></tr>
<tr><td>Array specifications for FB instances</td><td colspan="4">Supported</td></tr>
<tr><td colspan="2">Range specifications</td><td colspan="4">You can specify a range for a data type in advance. The data type can take only values that are in the specified range.</td></tr>
<tr><td colspan="3">Libraries (*)</td><td colspan="4">You can use user libraries.</td></tr>
<tr><td rowspan="10">Motion control *2</td><td colspan="3">Control modes</td><td colspan="4">Position control, Velocity control, and Torque control</td></tr>
<tr><td colspan="3">Axis types</td><td colspan="4">Servo axes, Virtual servo axes, Encoder axes, and Virtual encoder axes</td></tr>
<tr><td colspan="3">Positions that can be managed</td><td colspan="4">Command positions and actual positions</td></tr>
<tr><td rowspan="7">Single axes</td><td rowspan="4">Single-axis position control</td><td>Absolute positioning</td><td colspan="4">Positioning is performed for a target position that is specified with an absolute value.</td></tr>
<tr><td>Relative positioning</td><td colspan="4">Positioning is performed for a specified travel distance from the command current position.</td></tr>
<tr><td>Interrupt feeding</td><td colspan="4">Positioning is performed for a specified travel distance from the position where an interrupt input was received from an external input.</td></tr>
<tr><td>Cyclic synchronous absolute positioning (*)</td><td colspan="4">A positioning command is output each control period in Position Control Mode.</td></tr>
<tr><td rowspan="2">Single-axis velocity control</td><td>Velocity control</td><td colspan="4">Velocity control is performed in Position Control Mode.</td></tr>
<tr><td>Cyclic synchronous velocity control</td><td colspan="4">A velocity command is output each control period in Velocity Control Mode.</td></tr>
<tr><td>Single-axis torque control</td><td>Torque control</td><td colspan="4">The torque of the motor is controlled.</td></tr>
</table>

Item				NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Motion control *2	Single axes	Single-axis synchro-nized control	Starting cam operation	A cam motion is performed using the specified cam table.			
			Ending cam operation	The cam motion for the axis that is specified with the input parameter is ended.			
			Starting gear operation	A gear motion with the specified gear ratio is performed between a master axis and slave axis.			
			Positioning gear operation	A gear motion with the specified gear ratio and sync position is performed between a master axis and slave axis.			
			Ending gear operation	The specified gear motion or positioning gear motion is ended.			
			Synchronous positioning	Positioning is performed in sync with a specified master axis.			
			Master axis phase shift	The phase of a master axis in synchronized control is shifted.			
			Combining axes	The command positions of two axes are added or subtracted and the result is out-put as the command position.			
		Single-axis manual opera-tion	Powering the Servo	The Servo in the Servo Drive is turned ON to enable axis motion.			
			Jogging	An axis is jogged at a specified target velocity.			
		Auxil-iary func-tions for single-axis control	Resetting axis errors	Axes errors are cleared.			
			Homing	A motor is operated and the limit signals, home proximity signal, and home signal are used to define home.			
			Homing with specified parameters (*)	The parameters are specified, the motor is operated, and the limit signals, home proximity signal, and home signal are used to define home.			
			High-speed homing	Positioning is performed for an absolute target position of 0 to return to home.			
			Stopping	An axis is decelerated to a stop.			
			Immediately stopping	An axis is stopped immediately.			
			Setting over-ride factors	The target velocity of an axis can be changed.			
			Changing the current posi-tion	The command current position or actual current position of an axis can be changed to any position.			
			Enabling exter-nal latches	The position of an axis is recorded when a trigger occurs.			
			Disabling external latches	The current latch is disabled.			
			Zone monitor-ing	You can monitor the command position or actual position of an axis to see when it is within a specified range (zone).			
			Enabling digi-tal cam switches (*)	You can turn a digital output ON and OFF according to the position of an axis.			
			Monitoring axis following error	You can monitor whether the difference between the command positions or actual positions of two specified axes exceeds a threshold value.			
			Resetting the following error	The error between the command current position and actual current position is set to 0.			
			Torque limit	The torque control function of the Servo Drive can be enabled or disabled and the torque limits can be set to control the output torque.			
			Command position com-pensation	The function which compensate the position for the axis in operation.			
			Start velocity (*)	You can set the initial velocity when axis motion starts.			

Item				NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Motion control *2	Axes groups	Multi-axes coordinated control	Absolute linear interpolation	Linear interpolation is performed to a specified absolute position.			
			Relative linear interpolation	Linear interpolation is performed to a specified relative position.			
			Circular 2D interpolation	Circular interpolation is performed for two axes.			
			Axes group cyclic synchronous absolute positioning (*)	A positioning command is output each control period in Position Control Mode.			
		Auxiliary functions for multi-axes coordinated control	Resetting axes group errors	Axes group errors and axis errors are cleared.			
			Enabling axes groups	Motion of an axes group is enabled.			
			Disabling axes groups	Motion of an axes group is disabled.			
			Stopping axes groups	All axes in interpolated motion are decelerated to a stop.			
			Immediately stopping axes groups	All axes in interpolated motion are stopped immediately.			
			Setting axes group override factors	The blended target velocity is changed during interpolated motion.			
			Reading axes group positions (*)	The command current positions and actual current positions of an axes group can be read.			
			Changing the axes in an axes group (*)	The Composition Axes parameter in the axes group parameters can be overwritten temporarily.			
	Common items	Cams	Setting cam table properties	The end point index of the cam table that is specified in the input parameter is changed.			
			Saving cam tables	The cam table that is specified with the input parameter is saved in non-volatile memory in the CPU Unit.			
			Generating cam tables (*)	The cam table is generated from the cam property and cam node that is specified in input parameters.			
		Parameters	Writing MC settings	Some of the axis parameters or axes group parameters are overwritten temporarily.			
			Changing axis parameters (*)	Some of the axis parameters can be accessed or changed from the user program.			
	Auxiliary functions	Count modes		You can select either Linear Mode (finite length) or Rotary Mode (infinite length).			
		Unit conversions		You can set the display unit for each axis according to the machine.			
		Acceleration/deceleration control	Automatic acceleration/deceleration control	Jerk is set for the acceleration/deceleration curve for an axis motion or axes group motion.			
			Changing the acceleration and deceleration rates	You can change the acceleration or deceleration rate even during acceleration or deceleration.			
		In-position check		You can set an in-position range and in-position check time to confirm when positioning is completed.			
		Stop method		You can set the stop method to the immediate stop input signal or limit input signal.			
		Re-execution of motion control instructions		You can change the input variables for a motion control instruction during execution and execute the instruction again to change the target values during operation.			
		Multi-execution of motion control instructions (Buffer Mode)		You can specify when to start execution and how to connect the velocities between operations when another motion control instruction is executed during operation.			
		Continuous axes group motions (Transition Mode)		You can specify the Transition Mode for multi-execution of instructions for axes group operation.			

Item				NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Motion control *2	**Auxiliary functions**	**Monitoring functions**	**Software limits**	The movement range of an axis is monitored.			
			Following error	The error between the command current value and the actual current value is monitored for an axis.			
			Velocity, acceleration rate, deceleration rate, torque, interpolation velocity, interpolation acceleration rate, and interpolation deceleration rate	You can set and monitor warning values for each axis and each axes group.			
		Absolute encoder support		You can use an OMRON G5-series Servomotor with an Absolute Encoder to eliminate the need to perform homing at startup.			
		Input signal logic inversion (*)		You can inverse the logic of immediate stop input signal, positive limit input signal, negative limit input signal, or home proximity input signal.			
	External interface signals			The Servo Drive input signals listed on the right are used. Home signal, home proximity signal, positive limit signal, negative limit signal, immediate stop signal, and interrupt input signal			
Unit (I/O) management	**EtherCAT slaves**	**Maximum number of slaves**		512	192		64
	CJ-series Units	**Maximum number of Units**		---	40		
Communications	**Peripheral (USB) port**			A port for communications with various kinds of Support Software running on a personal computer.			
	EtherNet/IP port	**Communications protocol**		TCP/IP and UDP/IP			
		CIP communications services	**Tag data links**	Programless cyclic data exchange is performed with the devices on the EtherNet/IP network.			
			Message communications	CIP commands are sent to or received from the devices on the EtherNet/IP network.			
		TCP/IP applications	**Socket services**	Data is sent to and received from any node on Ethernet using the UDP or TCP protocol. Socket communications instructions are used.			
			FTP client (*)	Files are transferred via FTP from the CPU Unit to computers or Controllers at other Ethernet nodes. FTP client communications instructions are used.			
			FTP server	Files can be read from or written to the SD Memory Card in the CPU Unit from computers at other Ethernet nodes.			
			Automatic clock adjustment	Clock information is read from the NTP server at the specified time or at a specified interval after the power supply to the CPU Unit is turned ON. The internal clock time in the CPU Unit is updated with the read time.			
			SNMP agent	Built-in EtherNet/IP port internal status information is provided to network management software that uses an SNMP manager.			
	EtherCAT port	**Supported services**	**Process data communications**	A communications method to exchange control information in cyclic communications between the EtherCAT master and slaves. This communications method is defined by CoE.			
			SDO communications	A communications method to exchange control information in noncyclic event communications between EtherCAT master and slaves. This communications method is defined by CoE.			

<table>
<tr><th colspan="4">Item</th><th>NX701-□□□□</th><th>NJ501-□□□□</th><th>NJ301-□□□□</th><th>NJ101-□□□□</th></tr>
<tr><td rowspan="8">Commu-nica-tions</td><td rowspan="7">EtherCAT port</td><td colspan="2">Network scanning</td><td colspan="4">Information is read from connected slave devices and the slave configuration is automatically generated.</td></tr>
<tr><td colspan="2">DC (distributed clock)</td><td colspan="4">Time is synchronized by sharing the EtherCAT system time among all EtherCAT devices (including the master).</td></tr>
<tr><td colspan="2">Packet monitoring</td><td colspan="4">The frames that are sent by the master and the frames that are received by the master can be saved. The data that is saved can be viewed with WireShark or other applications. (*)</td></tr>
<tr><td colspan="2">Enable/disable settings for slaves</td><td colspan="4">The slaves can be enabled or disabled as communications targets.</td></tr>
<tr><td colspan="2">Disconnecting/reconnecting slaves</td><td colspan="4">Temporarily disconnects a slave from the EtherCAT network for maintenance, such as for replacement of the slave, and then connects the slave again.</td></tr>
<tr><td>Support application protocol</td><td>CoE</td><td colspan="4">SDO messages of the CAN application can be sent to slaves via EtherCAT</td></tr>
<tr><td colspan="3">Communications instructions</td><td>FTP client instructions, CIP communications instructions, socket communications instructions, and SDO message instructions</td><td colspan="3">FTP client instructions (*), CIP communications instructions, socket communications instructions, SDO message instructions, no-protocol communications instructions, and protocol macro instructions</td></tr>
<tr><td>Opera-tion man-age-ment</td><td colspan="3">RUN output contacts</td><td colspan="4">The output on the Power Supply Unit turns ON in RUN mode.</td></tr>
<tr><td rowspan="4">Sys-tem man-age-ment</td><td rowspan="4">Event logs</td><td colspan="2">Function</td><td colspan="4">Events are recorded in the logs</td></tr>
<tr><td rowspan="3">Maxi-mum number of events</td><td>System event log</td><td>2,048</td><td>1,024</td><td colspan="2">512</td></tr>
<tr><td>Access event log</td><td>1,024</td><td>1,024</td><td colspan="2">512</td></tr>
<tr><td>User-defined event log</td><td>1,024</td><td>1,024</td><td colspan="2">512</td></tr>
</table>

Item				NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Debugging	Online editing			Programs, function blocks, functions, and global variables can be changed online. More than one operators can change POUs individually via network.			
Debugging	Forced refreshing			The user can force specific variables to TRUE or FALSE.			
Debugging	Forced refreshing	Maximum number of forced variables	Device variables for EtherCAT slaves	64			
Debugging	Forced refreshing	Maximum number of forced variables	Device variables for CJ-series Units and variables with AT specifications	---	64		
Debugging	MC Test Run*2			Motor operation and wiring can be checked from the Sysmac Studio.			
Debugging	Synchronizing			The project file in the Sysmac Studio and the data in the CPU Unit can be made the same when online.			
Debugging	Differential monitoring (*)			You can monitor when a variable changes to TRUE or changes to FALSE.			
Debugging	Differential monitoring (*)	Maximum number of monitored variables		8			
Debugging	Data tracing	Types	Single triggered trace	When the trigger condition is met, the specified number of samples are taken and then tracing stops automatically.			
Debugging	Data tracing	Types	Continuous trace	Data tracing is executed continuously and the trace data is collected by the Sysmac Studio.			
Debugging	Data tracing	Maximum number of simultaneous data traces		4		2	
Debugging	Data tracing	Maximum number of records		10,000			
Debugging	Data tracing	Maximum number of sampled variables		192 variables		48 variables	
Debugging	Data tracing	Timing of sampling		Sampling is performed for the specified task period, at the specified time, or when a sampling instruction is executed.			
Debugging	Data tracing	Triggered traces		Trigger conditions are set to record data before and after an event.			
Debugging	Data tracing	Triggered traces	Trigger conditions	• When BOOL variable changes to TRUE or FALSE • Comparison of non-BOOL variable with a constant Comparison Method: Equals (=), Greater than (>), Greater than or equals (≥), Less Than (<), Less than or equals (≤), Not equal (≠)			
Debugging	Data tracing		Delay	Trigger position setting: A slider is used to set the percentage of sampling before and after the trigger condition is met.			
Debugging	Simulation			The operation of the CPU Unit is emulated in the Sysmac Studio.			
Reliability functions	Self-diagnosis	Controller errors	Levels	Major faults, partial faults, minor faults, observation, and information			
Reliability functions	Self-diagnosis	Controller errors	Maximum number of message languages	9 (Sysmac Studio) 2 (NS-series PT)			
Reliability functions	Self-diagnosis	User-defined errors	Function	User-defined errors are registered in advance and then records are created by executing instructions.			
Reliability functions	Self-diagnosis	User-defined errors	Levels	8			
Reliability functions	Self-diagnosis	User-defined errors	Maximum number of message languages	9			

<table>
<tr><th colspan="4">Item</th><th>NX701-□□□□</th><th>NJ501-□□□□</th><th>NJ301-□□□□</th><th>NJ101-□□□□</th></tr>
<tr><td rowspan="8">Secu-rity</td><td rowspan="8">Protecting software assets and preventing operating mistakes</td><td colspan="2">CPU Unit names and serial IDs</td><td colspan="4">When going online to a CPU Unit from the Sysmac Studio, the CPU Unit name in the project is compared to the name of the CPU Unit being connected to.</td></tr>
<tr><td rowspan="4">Protec-tion</td><td>User program transfer with no restoration information</td><td colspan="4">You can prevent reading data in the CPU Unit from the Sysmac Studio.</td></tr>
<tr><td>CPU Unit write-protection</td><td colspan="4">You can prevent writing data to the CPU Unit from the Sysmac Studio or SD Memory Card.</td></tr>
<tr><td>Overall Project file protection</td><td colspan="4">You can use passwords to protect .smc files from unauthorized opening on the Sysmac Studio.</td></tr>
<tr><td>Data protection (*)</td><td colspan="4">You can use passwords to protect POUs on the Sysmac Studio.</td></tr>
<tr><td colspan="2">Verification of operation authority</td><td colspan="4">Online operations can be restricted by operation rights to prevent damage to equipment or injuries that may be caused by operating mistakes.</td></tr>
<tr><td></td><td>Number of groups (*)</td><td colspan="4">5</td></tr>
<tr><td colspan="2">Verification of user program execution ID</td><td colspan="4">The user program cannot be executed without entering a user program execution ID from the Sysmac Studio for the specific hardware (CPU Unit).</td></tr>
<tr><td rowspan="5">SD Memory Card functions</td><td colspan="3">Storage type</td><td colspan="4">SD card or SDHC card</td></tr>
<tr><td rowspan="4">Application</td><td colspan="2">Automatic transfer from SD Memory Card (*)</td><td colspan="4">When the power supply to the Controller is turned ON, the data that is stored in the autoload directory of the SD Memory Card is transferred to the Controller.</td></tr>
<tr><td colspan="2">SD Memory Card operation instructions</td><td colspan="4">You can access SD Memory Cards from instructions in the user program.</td></tr>
<tr><td colspan="2">File operations from the Sysmac Studio</td><td colspan="4">You can perform file operations for Controller files in the SD Memory Card and read/write general-purpose document files on the computer.</td></tr>
<tr><td colspan="2">SD Memory Card life expiration detection</td><td colspan="4">Notification of the expiration of the life of the SD Memory Card is provided in a system-defined variable and event log.</td></tr>
<tr><td rowspan="6">Backing up data (*)</td><td rowspan="5">SD Memory Card backups</td><td rowspan="4">Operating methods</td><td>CPU Unit front-panel DIP switch</td><td colspan="4">Backup, verification, and restoration operations are performed by manipulating the front-panel DIP switch on the CPU Unit.</td></tr>
<tr><td>Specification with system-defined variables</td><td colspan="4">Backup and verification operations are performed by manipulating system-defined variables.</td></tr>
<tr><td>SD Memory Card Window in Sysmac Studio</td><td colspan="4">Backup and verification operations are performed from the SD Memory Card Window of the Sysmac Studio.</td></tr>
<tr><td>Special instruction (*)</td><td colspan="4">The special instruction is used to backup data.</td></tr>
<tr><td>Protection</td><td>Disabling backups to SD Memory Cards</td><td colspan="4">Backing up data to a SD Memory Card is prohibited.</td></tr>
<tr><td colspan="3">Sysmac Studio Controller backups</td><td colspan="4">The Sysmac Studio is used to backup, restore, or verify Controller data.</td></tr>
</table>

*1 Inline ST is supported. (Inline ST is ST that is written as an element in a ladder diagram.)
*2 This function is not available for NJ101-90□□ CPU Units.

Note Items that are marked with asterisks in the table were added for version upgrades. Refer to *A-13 Version Information for NX-series Controllers* and *A-14 Version Information for NJ-series Controllers* for information on version upgrades.

A-2 Calculating Guidelines for the Real Processing Times of Tasks for the NX-series System

This section describes how to calculate guidelines for the average real processing times of tasks on paper for the NX-series System.

You must use the physical Controller to check the real processing times of tasks and task execution times. For details, refer to *5-10 Task Design Methods and I/O Response Times*.

Precautions for Safe Use

The task execution times in the physical Controller depends on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors.

Before starting actual operation, you must test performance under all foreseeable conditions on the actual system and make sure that the task periods are not exceeded and that suitable communications performance is achieved.

Precautions for Correct Use

Calculation of the average real processing times of tasks for the priority-5 periodic task will require the value of PDO Communications Cycle 2: Transmission Delay Time, which is displayed in the EtherCAT Tab Page after the EtherCAT configuration is created with the Sysmac Studio.

Additional Information

Periodic tasks may be interrupted for the execution of tasks with higher execution priorities. The real processing time of a task does not include the time for which the task is interrupted. It is the task execution time that gives the actual time from when the task is started until it is finished, including the interrupted time. For a detailed description of the differences between the real processing times of tasks and the task execution times, refer to *Meaning of the Task Execution Time and the Real Processing Time of the Task* on page 5-89.

A-2-1 Calculating the Average Real Processing Times of Tasks

The average real processing time of a task is the total of the I/O refresh processing time, user program execution time, motion control processing time and common processing time.

Average real processing time of task = I/O refresh processing time + User program execution time + Motion control processing time + Common processing time

The following processing is performed.

Processing		Processing contents	Primary periodic task	Priority-5 periodic task	Priority-16 periodic task	Priority-17 and priority-18 periodic tasks
I/O refresh processing		I/O is refreshed for EtherCAT slaves.	Performed.	Performed.	Not performed.	Not performed.
User program execution		• Programs assigned to tasks are executed in the order that they are assigned.	Performed.	Performed.	Performed.	Performed.
Motion control processing		• Motion control commands from the user program are executed. • Motion output processing	Performed.	Performed.	Not performed.	Not performed.
Common processing time	System common processing 1	• Variable refresh processing (if there are accessing tasks) is performed. • Motion input processing • Data trace processing	Performed.	Performed.	Performed.	Performed.
	System common processing 2	• Variable refresh processing (if there are refreshing tasks) is performed. • Variable access processing external to the Controller to ensure concurrency with task execution	Performed.	Performed.	Performed.	Performed.
	System overhead time	Other system common processing	Performed.	Performed.	Performed.	Performed.

Guidelines are provided below for calculating the various processing times.

I/O Refresh Processing Time

Use the following formula for the I/O refresh processing time.

I/O refresh processing time = EtherCAT slave processing time

The formula for calculating the EtherCAT slave processing time is different between the primary periodic task and the priority-5 periodic task.

The following describes how to determine the EtherCAT slave processing time for each type of tasks.

Precautions for Correct Use

When calculating for the priority-5 periodic task, you need to determine the task period of the primary periodic task in advance.

Additional Information

The EtherCAT slave processing time is 0 in tasks to which EtherCAT slaves are not assigned.

● EtherCAT Slave Processing Time in Primary Periodic Task

Use the following formula to calculate the EtherCAT slave processing time in the primary periodic task.

EtherCAT slave processing time [μs] = $0.0006 \times \text{pDout} + 0.0001 \times \text{pDin} + 0.082 \times \text{pDinout}$
$+ (1.24 \times \text{Snum} + 0.01 \times \text{Clen})$

pDout	:	Total output processing data size [byte] of EtherCAT slaves assigned to the primary periodic task
pDin	:	Total input processing data size [byte] of EtherCAT slaves assigned to the primary periodic task
pDinout	:	Total of the larger of the input and output processing data size of each EtherCAT slave assigned to the primary periodic task [byte]
Snum	:	Total number of EtherCAT slaves connected to the built-in EtherCAT port
Clen	:	Total length of cables connected to the built-in EtherCAT port [m]

EtherCAT Slave Processing Time in Priority-5 Periodic Task

Use the following formula to calculate the EtherCAT slave processing time in the priority-5 periodic task.

EtherCAT slave processing time [μs] = 0.0006 × pDout + (Larger of the following A and B) + sTsend + 0.0001 × sDin + 16

A [μs] = ((0.0623 × sDout + 37) × pTcycle) ÷ (pTcycle − (0.0243 × pDout + 35))
B [μs] = 0.082 × pDinout

pDout	:	Total output processing data size [byte] of EtherCAT slaves assigned to the primary periodic task
sTsend*1	:	Transmission delay time of PDO communications cycle 2 [μs] displayed on the Sysmac Studio
sDin	:	Total input processing data size [byte] of EtherCAT slaves assigned to the priority-5 periodic task
sDout	:	Total output processing data size [byte] of EtherCAT slaves assigned to the priority-5 periodic task
pTcycle	:	Task period of the primary periodic task [μs]
pDinout	:	Total of the larger of the input and output processing data size of each EtherCAT slave assigned to the primary periodic task [byte]

*1 The value of sTsend (transmission delay time) cannot be calculated on paper. Assign the value of PDO Communications Cycle 2: Transmission Delay Time, which is displayed in the EtherCAT Tab Page after the EtherCAT configuration is created with the Sysmac Studio, to sTsend (transmission delay time).
Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No W504-E1-14 or later) for how to display the PDO Communications Cycle 2: Transmission Delay Time on the Sysmac Studio.

User Program Execution Time

The user program execution time depends on the specific instructions multiplied by the numbers of instructions used.

As a guideline, instructions are divided into three groups and the number of instructions in each group is used for measurements and estimates.

- Standard instructions
- Arithmetic instructions for LREAL data
- Trigonometric instructions for LREAL data

Different instructions are used in a ladder diagram and in ST. Refer to *Instruction Configuration for Standard Ladder Diagram Instructions* on page A-19 and *Instruction Configuration for Standard ST Instructions* on page A-21 for information on the instruction configuration.

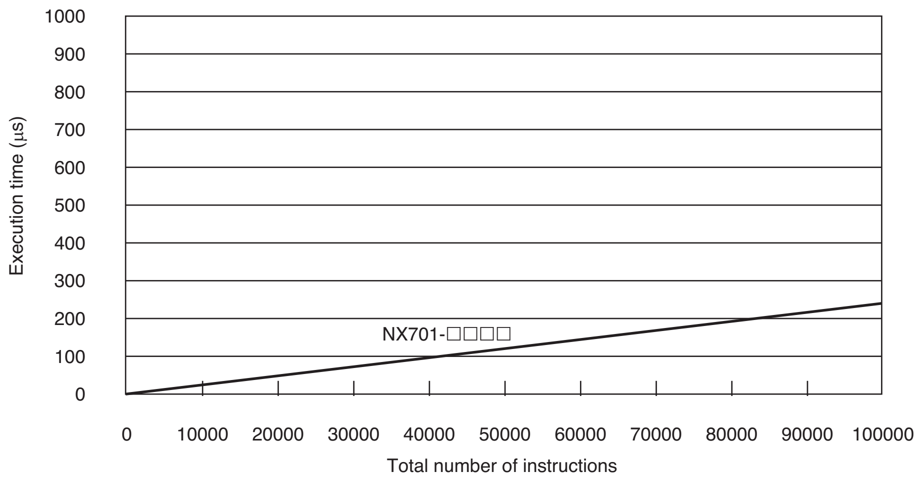
● Simple Estimate

For the number of instructions in each group, read the execution time for each instruction group from the following graphs and calculate the total.

- Execution time for standard instructions
- Execution time for arithmetic instructions for LREAL data
- Execution time for trigonometric instructions for LREAL data

This will allow you to estimate the execution time of the user program.

Execution Time for Standard Ladder Diagram Instructions



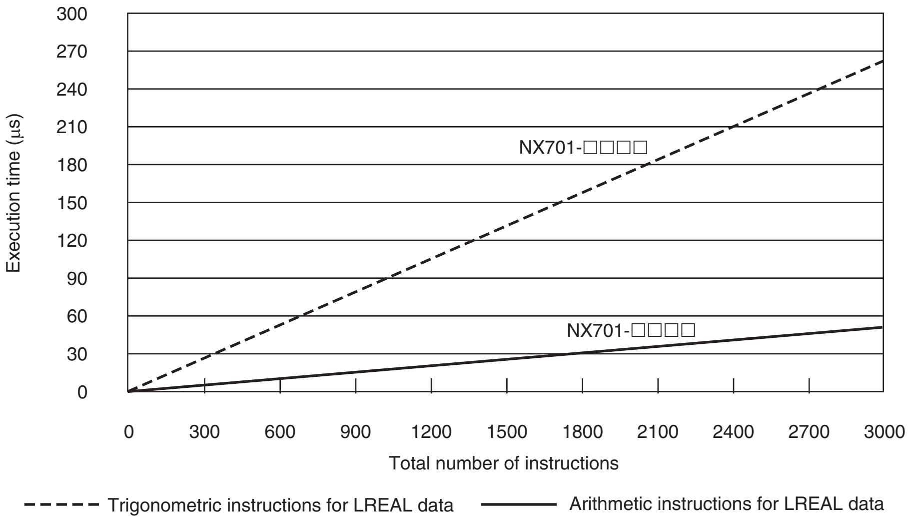
- Instruction Configuration for Standard Ladder Diagram Instructions

The instruction execution ratio for this configuration is 20%.

Types of instructions	Instructions	Percent of instructions [%]	Percent of execution time in instruction group [%]
Ladder diagram instructions	LD, AND, OUT, SET, and RESET	81.0%	40.2%
Comparison instructions	EQ and LT	4.1%	8.3%
Timer and counter instructions	Timer, TON/TOF, and CTU/CTD	1.6%	7.3%
Math instructions	+, -, *, /, ADD, SUB, MUL, and DIV	2.4%	6.5%
BCD conversion instructions and data conversion instructions	INT_TO_DINT and WORD_BCD_TO_UINT	0.2%	1.2%

Types of instructions	Instructions	Percent of instructions [%]	Percent of execution time in instruction group [%]
Bit string processing instructions	AND and OR	6.2%	13.0%
Data movement instructions	MOVE	4.6%	23.5%
Total		100.0%	100.0%

Execution Times for Ladder Diagram Arithmetic and Trigonometric Instructions for LREAL Data



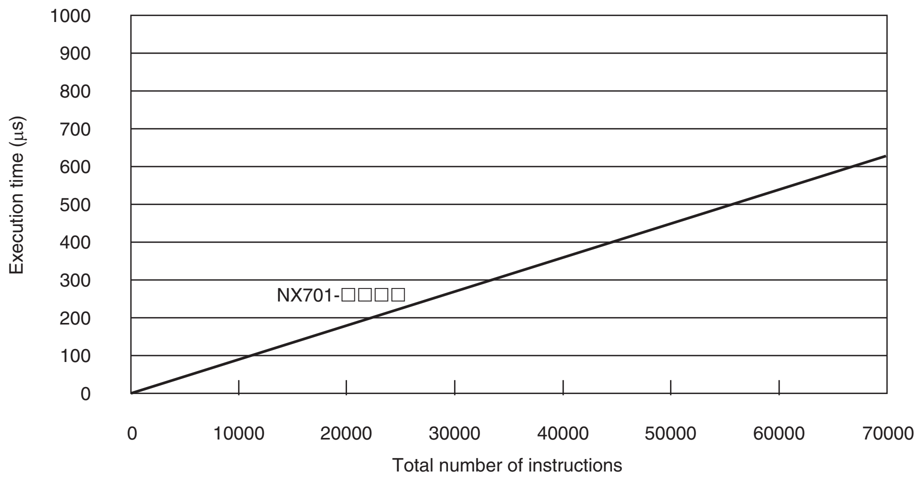
- Configuration of Arithmetic Instructions for LREAL Data

Instructions	Percent of instructions [%]
Addition instructions for LREAL data	20.0%
Subtraction instructions for LREAL data	20.0%
Multiplication instructions for LREAL data	30.0%
Division instructions for LREAL data	30.0%
Total	100.0%

- Configuration of Trigonometric Instructions for LREAL Data

Instructions	Percent of instructions [%]
Sin of LREAL data	16.7%
Cos of LREAL data	16.7%
Tan of LREAL data	16.7%
Sin^{-1} of LREAL data	16.7%
Cos^{-1} of LREAL data	16.7%
Tan^{-1} of LREAL data	16.7%
Total	100.0%

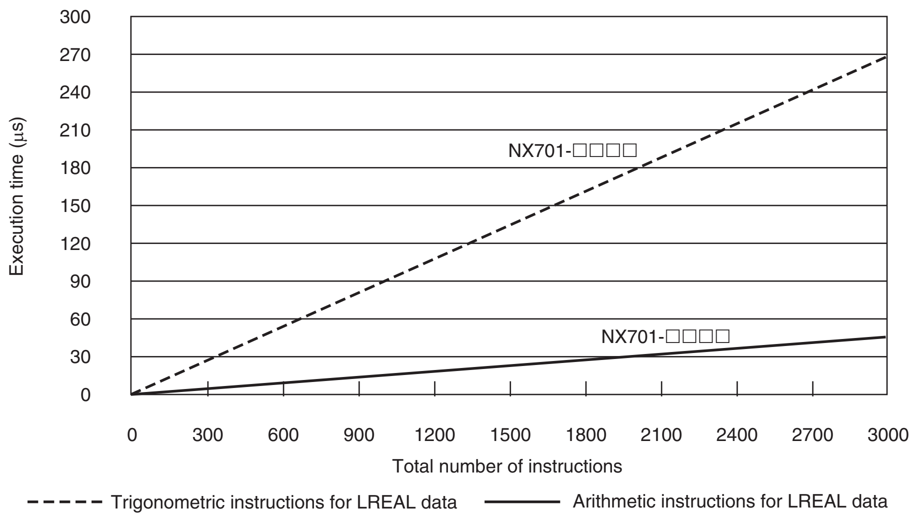
Execution Time for Standard ST Instructions



- Instruction Configuration for Standard ST Instructions

Types of instructions	Instructions	Percent of instructions [%]	Percent of execution time in instruction group [%]
ST constructs	IF ELSIF END_IF	75.4%	41.6%
Comparison instructions	EQ and LT	5.2%	8.7%
Timer and counter instructions	Timer, TON/TOF, and CTU/CTD	2.1%	18.8%
Math instructions	+, -, *, and /	3.1%	10.2%
BCD conversion instructions and data conversion instructions	INT_TO_DINT and WORD_BCD_TO_UINT	0.2%	1.6%
Bit string processing instructions	AND and OR	8.0%	11.7%
Data movement instructions	:=	5.9%	7.3%
Total		100.0%	100.0%

Execution Times for ST Arithmetic and Trigonometric Instructions for LREAL Data



- Configuration of Arithmetic Instructions for LREAL Data

Instructions	Percent of instructions [%]
Addition instructions for LREAL data	20.0%
Subtraction instructions for LREAL data	20.0%
Multiplication instructions for LREAL data	30.0%
Division instructions for LREAL data	30.0%
Total	100.0%

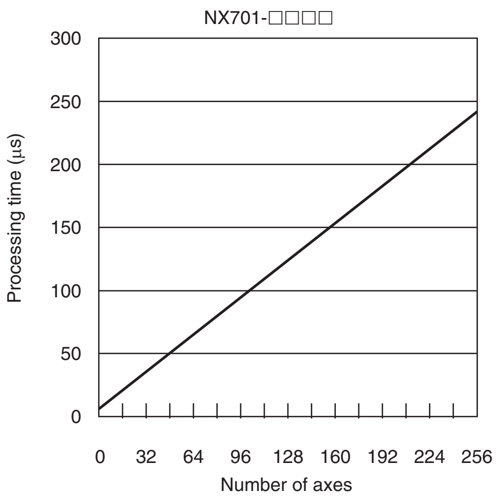
- Configuration of Trigonometric Instructions for LREAL Data

Instructions	Percent of instructions [%]
Sin of LREAL data	16.7%
Cos of LREAL data	16.7%
Tan of LREAL data	16.7%
Sin^{-1} of LREAL data	16.7%
Cos^{-1} of LREAL data	16.7%
Tan^{-1} of LREAL data	16.7%
Total	100.0%

Motion Control Processing Time

The motion control processing time depends on the number of servo axes and virtual servo axes that are used.

For the number of servo and virtual servo axes, read the motion control processing time from the following graph.



Common Processing Time

The common processing time is the following values by the total time for system overhead, system common processing 1, and system common processing 2. The common processing time depends on the type of task.

Type of task	Common processing times [μs] (reference values)
	NX701-□□□□
Primary periodic task	45
Priority-5 periodic task	55
Priority-16, 17, or 18 periodic task	10

A-2-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period

Example of Calculating the Average Real Processing Times of Tasks

If you are using an NX701-□□□□ CPU Unit with a unit version of 1.10, first find the average real processing time of the task for the following conditions.

The task is the primary periodic task.

Item		Conditions	
Slaves/Units that are used	EtherCAT slaves	GX-ID1611 (Ver. 1.1) Input Slave: 1	
		GX-OD1611 (Ver. 1.1) Output Slave: 1	
		R88D-□□ (Ver. 2.1) Servomotors: 4	
		EtherCAT Slave Terminal: 1	
			NX-ECC203 EtherCAT Coupler Unit: 1
			NX-ID5342 DC Input Unit: 1
			NX-OD5121 Transistor Output Unit: 1
			NX-AD3608 Analog Input Unit: 1
			NX-DA2605 Analog Output Unit: 1
			NX-CIF101 and NX-CIF105 Communications Interface Units: 1 per Unit
User program	Language	Ladder diagrams	
	Standard instruction configuration	Number of instructions: 5,000	
	Arithmetic instructions for LREAL data	Number of instructions: 200	
	Trigonometric instructions for LREAL data	Number of instructions: 100	
Motion control processing	Number of axes	4	

Note Total length of cables connected to the built-in EtherCAT port is 10 [m].

● I/O Refresh Time

EtherCAT slave processing time:

The following table gives the pDout (output processing data size), pDin (input processing data size), and pDinout (larger of the input and output data size) values of the GX-ID1611 (Ver. 1.1) Input Slave, GX-OD1611 (Ver. 1.1) Output Slave, R88D-□□ (Ver. 2.1) Servomotors, and EtherCAT Slave Terminal configured with Units shown above.

EtherCAT slave	pDout: Output processing data size in bytes	pDin: Input processing data size in bytes	pDinout: Larger of the input and output data size
GX-ID1611 (Ver. 1.1)	0	3	3
GX-OD1611 (Ver. 1.1)	2	1	2
R88D-□□ (Ver. 2.1)	23	26	26
EtherCAT Slave Terminal (Total)	62	88	88
NX-ECC203	0	18	---
NX-ID5342	0	2	
NX-OD5121	2	0	
NX-AD3608	0	8	
NX-DA2605	4	0	
NX-CIF101	28	30	
NX-CIF105	28	30	

Total number of bytes are given below for pDout, pDin and pDinout.

pDout = $2 + 23 \times 4 + 62 = 156$ [byte]

pDin = $3 + 1 + 26 \times 4 + 88 = 196$ [byte]

pDinout = $3 + 2 + 26 \times 4 + 88 = 197$ [byte]

From these values, the I/O refresh time is calculated by the following formula.

I/O refresh processing time

= EtherCAT slave processing time

= $0.0006 \times$ pDout + $0.0001 \times$ pDin + $0.082 \times$ pDinout + ($1.24 \times$ Snum + $0.01 \times$ Clen)

= $0.0006 \times 156 + 0.0001 \times 196 + 0.082 \times 197 + (1.24 \times 7 + 0.01 \times 10)$

= $0.0936 + 0.0196 + 16.154 + (8.68 + 0.1)$

≈ 25 [μs]

● User Program Execution Time

The graphs show the following values.

- Standard instruction configuration
  From the graph of the execution time for standard ladder diagram instructions, the user program execution time of 5,000 instructions for the NX701-□□□□ is 12 μs.
- Arithmetic instructions for LREAL data
  From the graph of the execution time for ladder diagram arithmetic and trigonometric instructions for LREAL data, the user program execution time of 200 instructions for the NX701-□□□□ is 4 μs.

- Trigonometric instructions for LREAL data
  From the graph of the execution time for ladder diagram arithmetic and trigonometric instructions for LREAL data, the user program execution time of 100 instructions for the NX701-□□□□ is 9 μs.

Therefore, the user program execution time is the total of the above values, which is given by the following formula.

User program execution time = 12 + 4 + 9
= 25 [μs]

● Motion Control Processing Time

From the graph of the execution time for motion control processing, the execution time of the motion control processing for four axes for the NX701-□□□□ with a unit version 1.10 is read as 10 μs.

● Common Processing Time

Because the task is the primary periodic task, the common processing time for the NX701-□□□□ is 45 μs.

Therefore, the average real processing time of the task is given by the following formula.

Average real processing time of task = I/O refresh processing time + User program execution time
+ Motion control processing time + Common processing time
= 25 + 25 + 10 + 45
= 105 [μs]

Setting the Task Period

The task period is set based on the average real processing time of the task that is calculated as above. The task is the primary periodic task.

The value of the task period must be larger than the average real processing time of the task that you calculated. More specifically, you should allow sufficient margin and set the task period value to at least 1.1 times as large as the average real processing time of the task.

Task period ≥ Average real processing time of task × 1.1

The task execution times in the physical Controller depends on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors. The task execution time for a periodic task depends on whether it is interrupted for the execution of tasks with higher execution priorities.

Use the physical Controller and verify the task execution time with the Task Execution Time Monitor.

A-3 Calculating Guidelines for the Real Processing Times of Tasks for the NJ-series System

This section describes how to calculate guidelines for the average real processing times of tasks on paper for the NJ-series System.

You must use the physical Controller to check the real processing times of tasks and task execution times. For details, refer to *5-10 Task Design Methods and I/O Response Times*.

Precautions for Safe Use

The task execution times in the physical Controller depends on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors.
Before starting actual operation, you must test performance under all foreseeable conditions on the actual system and make sure that the task periods are not exceeded and that suitable communications performance is achieved.

Additional Information

Periodic tasks will be interrupted for the execution of tasks with higher execution priorities. The real processing time of a task does not include the time for which the task is interrupted. It is the task execution time that gives the actual time from when the task is started until it is finished, including the interrupted time. For a detailed description of the differences between the real processing times of tasks and the task execution times, refer to *Meaning of the Task Execution Time and the Real Processing Time of the Task* on page 5-89.

A-3-1 Calculating the Average Real Processing Times of Tasks

The average real processing time of a task is the total of the I/O refresh processing time, user program execution time, motion control processing time and common processing time.

Average real processing time of task = I/O refresh processing time + User program execution time + Motion control processing time + Common processing time

The following processing is performed.

Processing		Processing contents	Primary periodic task	Priority-16 periodic task	Priority-17 and priority-18 periodic tasks
I/O refresh processing		I/O is refreshed for CJ-series Units (Basic I/O Units, Special I/O Units, and CPU Bus Units) and EtherCAT slaves.	Performed.	Performed.	Not performed.
User program execution		• Programs assigned to tasks are executed in the order that they are assigned.	Performed.	Performed.	Performed.
Motion control processing		• Motion control commands from the user program are executed. • Motion control outputs are processed.	Performed.	Not performed.	Not performed.
Common processing time	System common processing 1	• Variable refresh processing (if there are accessing tasks) is performed. • Motion input processing • Data trace processing	Performed.	Performed.	Performed.
	System common processing 2	• Variable refresh processing (if there are refreshing tasks) is performed. • Variable access processing external to the Controller to ensure concurrency with task execution	Performed.	Performed.	Performed.
	System overhead time	Other system common processing	Performed.	Performed.	Performed.

Guidelines are provided below for calculating the various processing times.

I/O Refresh Processing Time

Use the following formula for the I/O refresh processing time.

I/O refresh processing time = I/O refresh overhead time
+ (Larger of the EtherCAT slave processing time and the CJ-series Unit processing time)

The following describes how to determine the I/O refresh overhead time, EtherCAT slave processing time, and CJ-series Unit processing time used in the above formula.

● I/O Refresh Overhead Time

The I/O refresh overhead time is given by the following table, depending on whether there are EtherCAT slaves and CJ-series Units, and also depending on the models of the CPU Units.

EtherCAT slaves	CJ-series Unit	I/O refresh overhead time [μs]		
		NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Present	Present	65*1	95	135
Present	None	35*2	50	65
None	Present	30	45	70

*1 The value is 90 for a CPU Unit with unit version 1.01 or earlier.
*2 The value is 60 for a CPU Unit with unit version 1.01 or earlier.

● EtherCAT Slave Processing Time

Use the following formula for the EtherCAT slave processing time.

The calculation must include all EtherCAT slaves.

EtherCAT slave processing time [μs] = Tout × Dout + Tin × Din + Tref × Dinout
+ (1.24 × Snum + 0.01 × Clen - Tec)

Tout	: Output processing time per byte [μs]
Dout	: Total output processing data size [byte]
Tin	: Input processing time per byte [μs]
Din	: Total input processing data size [byte]
Snum	: Number of connected EtherCAT slaves
Tref	: Refresh processing time per byte [μs]
Dinout	: Larger of the total input and output process data size [byte]
Clen	: Total cable length [m]
Tovh	: I/O refresh overhead time [μs]
Tec	: EtherCAT communications adjustment time [μs]

The values of the output processing time, input processing time, refresh processing time, and EtherCAT communications adjustment time in the above formula are fixed. They are determined by the model of the CPU Unit as given in the following table.

CPU Unit	Tout: Output processing time per byte [μs]	Tin: Input processing time per byte [μs]	Tref: Refresh processing time per byte [μs]	Tec: EtherCAT communications adjustment time [μs]
NJ501-□□□□	0.004	0.011	0.082	55*1
NJ301-□□□□	0.005	0.013	0.082	90
NJ101-□□□□	0.010	0.004	0.082	145

*1 The value is 70 for a CPU Unit with unit version 1.01 or earlier.

If the result that is calculated from the part (1.24 × Snum + 0.01 × Clen - Tec) of the above formula is a negative number, the result is regarded as 0.

● CJ-series Unit Processing Time

Use the following formula for the CJ-series Unit processing time.

Σ (I/O refresh time for each CJ-series Unit × Number of Units) - Tcj [μs]

In the above formula, Σ represents the total processing time for all CJ-series Units.

If the result that is calculated from the above formula is a negative number, the CJ-series Unit processing time is regarded as 0 μs.

The method for calculating the I/O refresh time for each CJ-series Unit is provided later.

The value of Tcj depends on the model and the unit version of the CPU Unit.

Model	Unit version	Tcj [μs]
NJ501-□□□□	Ver. 1.01 or earlier	230
	Ver. 1.02 or later	110
NJ301-□□□□	Ver. 1.01 or earlier	230
	Ver. 1.02 or later	165
NJ101-□□□□	Ver. 1.10 or later*1	225

*1 There is no NJ101-□□□□ CPU Unit with unit version 1.09 or earlier.

If any of the following CJ-series Units is used, add Tcj μs to the result that is calculated from the above formula, regardless of the number of Units.

- CJ1W-PH41U Analog Input Unit with Universal Inputs
- CJ1W-AD04U Analog Input Unit with Universal Inputs
- CJ1W-PDC15 Analog Input Unit with Universal Inputs
- CJ1W-V680C11 ID Sensor Unit
- CJ1W-V680C12 ID Sensor Unit
- CJ1W-CRM21 CompoNet Master Unit

I/O Refresh Times for CJ-series Units

This section gives the I/O refresh times for CJ-series Units.

● Basic I/O Units

The following table gives the I/O refresh times for Basic I/O Units that are used on the CPU Rack. The I/O refresh times for these Units on Expansion Racks are approx. 1.5 times the values that are given in the table.

Unit name	Model numbers	I/O refresh time per Unit [µs]
8/16-point DC Input Units	CJ1W-ID201/211/212	1
32-point DC Input Units	CJ1W-ID231/232/233	2
64-point DC Input Units	CJ1W-ID261/262	4
8/16-point AC Input Units	CJ1W-IA201/111	1
16-point Interrupt Input Unit	CJ1W-INT01	1
16-point Quick-response Input Unit	CJ1W-IDP01	1
Relay Contact Output Units	CJ1W-OC201/211	1
Triac Output Unit	CJ1W-OA201	1
8/16-point Transistor Output Units	CJ1W-OD201/202/203/204//211/212/213	1
32-point Transistor Output Units	CJ1W-OD231/232/233/234	2
64-point Transistor Output Units	CJ1W-OD261/262/263	4
24-VDC Input/Transistor Output Units (16 inputs/16 outputs)	CJ1W-MD231/232/233	1
24-VDC Input/Transistor Output Units (32 inputs/32 outputs)	CJ1W-MD261/263	2
TTL I/O Unit	CJ1W-MD563	4
B7A Interface Units	CJ1W-B7A04	4
	CJ1W-B7A14	4
	CJ1W-B7A22	4

● Special I/O Units

The following table gives the I/O refresh times for Special I/O Units that are used on the CPU Rack. The I/O refresh times for these Units on Expansion Racks are approx. 1.5 times the values that are given in the table.

Unit name	Model numbers	I/O refresh time per Unit [µs]
Isolated-type Units with Universal Inputs	CJ1W-AD04U	66
Analog Input Units	CJ1W-AD041-V1/081-V1/042	24
Analog Output Units	CJ1W-DA021/041/042V/08V/08C	24
Analog I/O Units	CJ1W-MAD42	24
Isolated-type Units with Universal Inputs	CJ1W-PH41U	80 (180)*1
Isolated-type DC Input Unit	CJ1W-PDC15	60 (100)*1
Temperature Control Units	CJ1W-TC□□□	114
ID Sensor Units	CJ1W-V680C11	76
	CJ1W-V680C12	86
High-speed Counter Unit	CJ1W-CT021	54

Unit name	Model numbers		I/O refresh time per Unit [μs]
CompoNet Master Unit	CJ1W-CRM21	Communications mode 0	34
		Communications mode 1	52
		Communications mode 2	88
		Communications mode 3	84
		Communications mode 8	14 + (1.0 × Number of allocated words)

*1 The values in parentheses are the I/O refresh times when an expansion allocation area is used.

*2 The number of allocated words is the total number of I/O area words that are allocated to all of the slaves.

CPU Bus Units

The following table gives the I/O refresh times for CPU Bus Units. The times are the same regardless of whether the Units are used on the CPU Rack or an Expansion Rack.

Unit name	Model numbers	I/O refresh time per Unit [μs]
Serial Communications Units	CJ1W-SCU42 CJ1W-SCU32 CJ1W-SCU22	Coefficient*1 × 25*2
DeviceNet Unit	CJ1W-DRM21	Coefficient*1 × (Number of allocated words*3 + 25)
EtherNet/IP Unit	CJ1W-EIP21	7.0 + 0.8 × Number of tags*4

*1 The coefficient is determined by the model of the CPU Unit as follows.
NJ501-□□□□: 0.1
NJ301-□□□□: 0.2
NJ101-□□□□: 0.3

*2 The following maximum time is added if a protocol macro is executed:
Coefficient × Number of refresh words [μs]

*3 The number of allocated words is the total number of I/O area words that are allocated to all of the slaves.

*4 For a CPU Unit with unit version 1.03 or later, the I/O refresh times for EtherNet/IP Units are not added.

User Program Execution Time

The user program execution time depends on the specific instructions multiplied by the numbers of instructions used.

As a guideline, instructions are divided into three groups and the number of instructions in each group is used for measurements and estimates.

- Standard instructions
- Arithmetic instructions for LREAL data
- Trigonometric instructions for LREAL data

Different instructions are used in a ladder diagram and in ST. Refer to *Instruction Configuration for Standard Ladder Diagram Instructions* on page A-32 and *Instruction Configuration for Standard ST Instructions* on page A-33 for information on the instruction configuration.

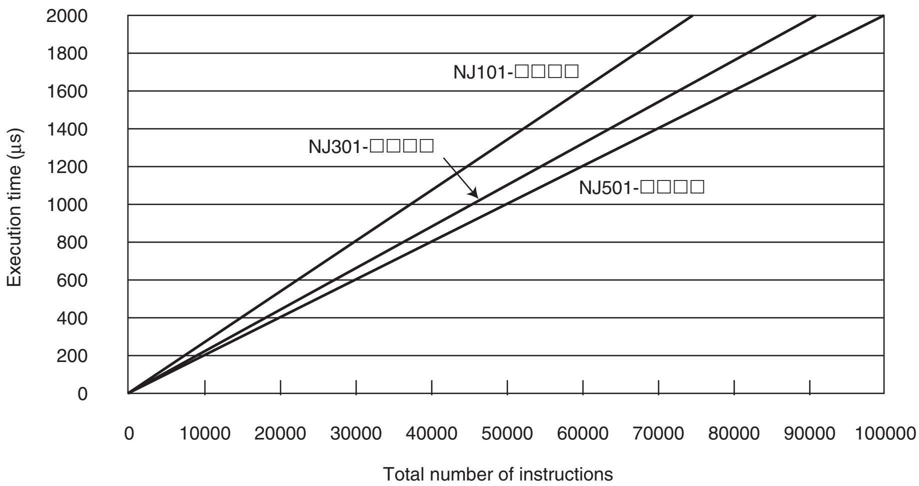
Simple Estimate

For the number of instructions in each group, read the execution time for each group from the following graphs and calculate the total.

- Execution time for standard instructions
- Execution time for arithmetic instructions for LREAL data
- Execution time for trigonometric instructions for LREAL data

This will allow you to estimate the execution time of the user program.

Execution Time for Standard Ladder Diagram Instructions

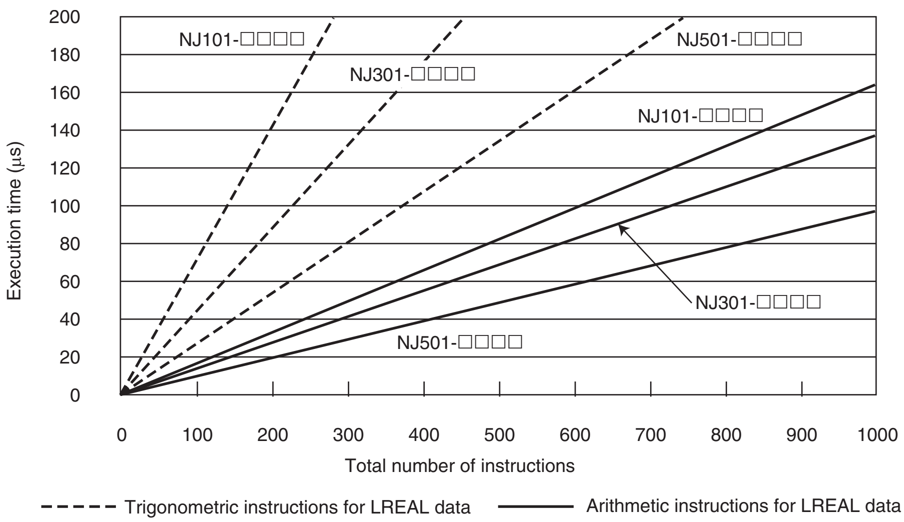


- Instruction Configuration for Standard Ladder Diagram Instructions

The instruction execution ratio for this configuration is 20%.

Types of instructions	Instructions	Percent of instruc-tions [%]	Percent of exe-cution time in instruction group [%]
Ladder diagram instructions	LD, AND, OUT, SET, and RESET	81.0%	40.2%
Comparison instructions	EQ and LT	4.1%	8.3%
Timer and counter instructions	Timer, TON/TOF, and CTU/CTD	1.6%	7.3%
Math instructions	+, -, *, /, ADD, SUB, MUL, and DIV	2.4%	6.5%
BCD conversion instructions and data conversion instructions	INT_TO_DINT and WORD_BCD_TO_UINT	0.2%	1.2%
Bit string processing instructions	AND and OR	6.2%	13.0%
Data movement instructions	MOVE	4.6%	23.5%
Total		100.0%	100.0%

Execution Times for Ladder Diagram Arithmetic and Trigonometric Instructions for LREAL Data



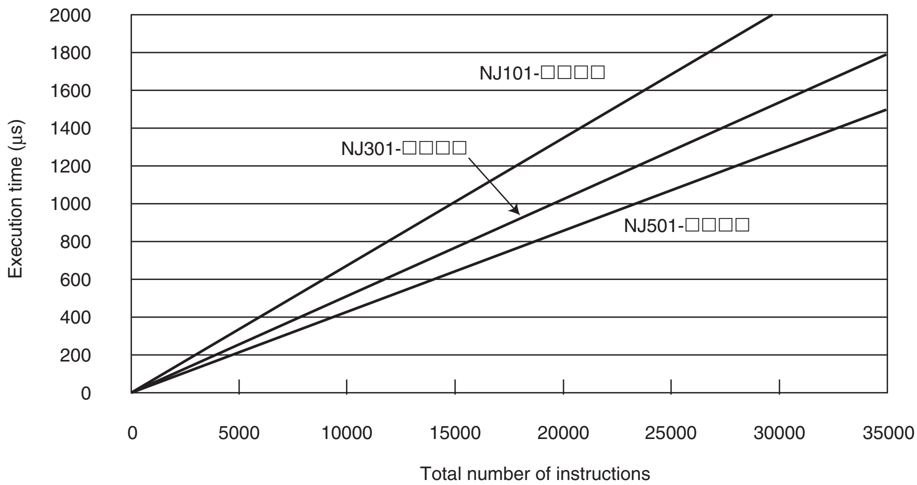
- Configuration of Arithmetic Instructions for LREAL Data

Instructions	Percent of instructions [%]
Addition instructions for LREAL data	20.0%
Subtraction instructions for LREAL data	20.0%
Multiplication instructions for LREAL data	30.0%
Division instructions for LREAL data	30.0%
Total	100.0%

- Configuration of Trigonometric Instructions for LREAL Data

Instructions	Percent of instructions [%]
Sin of LREAL data	16.7%
Cos of LREAL data	16.7%
Tan of LREAL data	16.7%
Sin^{-1} of LREAL data	16.7%
Cos^{-1} of LREAL data	16.7%
Tan^{-1} of LREAL data	16.7%
Total	100.0%

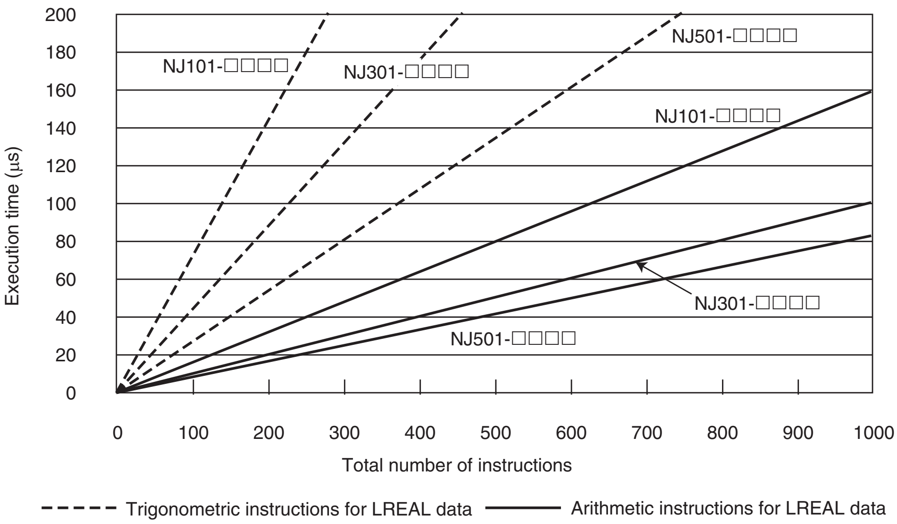
Execution Time for Standard ST Instructions



- Instruction Configuration for Standard ST Instructions

Types of instructions	Instructions	Percent of instructions [%]	Percent of execution time in instruction group [%]
ST constructs	IF ELSIF END_IF	75.4%	41.6%
Comparison instructions	EQ and LT	5.2%	8.7%
Timer and counter instructions	Timer, TON/TOF, and CTU/CTD	2.1%	18.8%
Math instructions	+, -, *, and /	3.1%	10.2%
BCD conversion instructions and data conversion instructions	INT_TO_DINT and WORD_BCD_TO_UINT	0.2%	1.6%
Bit string processing instructions	AND and OR	8.0%	11.7%
Data movement instructions	:=	5.9%	7.3%
Total		100.0%	100.0%

Execution Times for ST Arithmetic and Trigonometric Instructions for LREAL Data



- Configuration of Arithmetic Instructions for LREAL Data

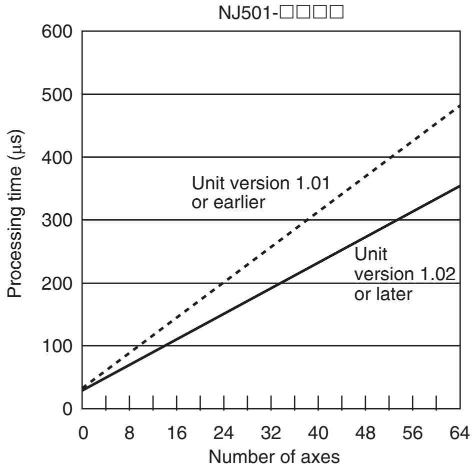
Instructions	Percent of instructions [%]
Addition instructions for LREAL data	20.0%
Subtraction instructions for LREAL data	20.0%
Multiplication instructions for LREAL data	30.0%
Division instructions for LREAL data	30.0%
Total	100.0%

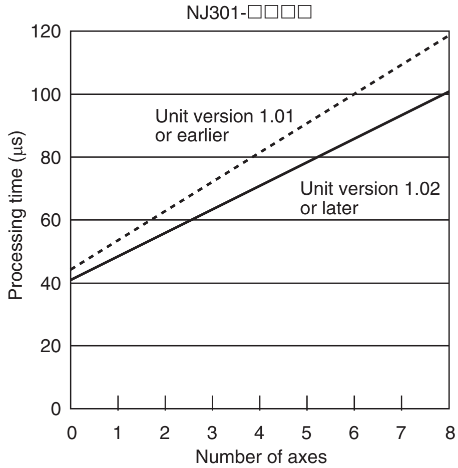
- Configuration of Trigonometric Instructions for LREAL Data

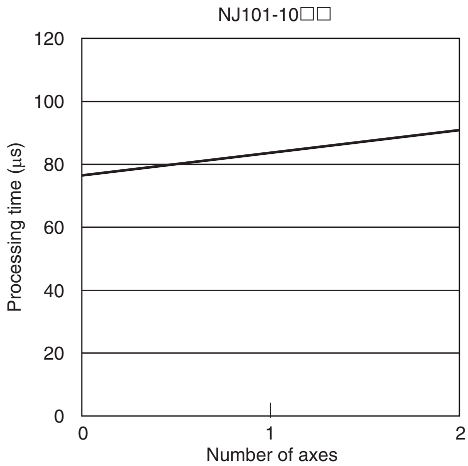
Instructions	Percent of instructions [%]
Sin of LREAL data	16.7%
Cos of LREAL data	16.7%
Tan of LREAL data	16.7%
Sin^{-1} of LREAL data	16.7%
Cos^{-1} of LREAL data	16.7%
Tan^{-1} of LREAL data	16.7%
Total	100.0%

Motion Control Processing Time

The motion control processing time depends on the number of servo axes and virtual servo axes that are used. For the number of servo and virtual servo axes, read the motion control processing time from the following graph.







Note You cannot use the motion control functions with an NJ101-90□□ CPU Unit.

Common Processing Time

The common processing time is the following values by the total time for system overhead, system common processing 1, and system common processing 2. The common processing time depends on the type of task.

Type of task	Common processing times [μs] (reference values)		
	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Primary periodic task	150*1	240*2	300
Periodic task	10	32	82

*1 The processing time is 265 μs for a CPU Unit with unit version 1.01 or earlier.

*2 The processing time is 360 μs for a CPU Unit with unit version 1.01 or earlier.

A-3-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period

Example of Calculating the Average Real Processing Times of Tasks

If you are using an NJ501-□□□ CPU Unit with a unit version of 1.02, first find the average real processing time of the task for the following conditions. The task is the primary periodic task.

Item		Conditions
Slaves/Units that are used	EtherCAT slaves	• GX-ID1611 (Ver. 1.1) Input Slave: 1 • GX-OD1611 (Ver. 1.1) Output Slave: 1 • R88D-□□ (Ver. 2.1) Servomotors: 4
	CJ-series Units (on CPU Rack)	• CJ1W-ID211 DC Input Unit: 1 • CJ1W-OD211 Transistor Output Unit: 1 • CJ1W-AD042 Analog Input Unit: 1 • CJ1W-DA021 Analog Output Unit: 1 • CJ1W-SCU42 Serial Communications Unit: 1 (Protocol macros are not used.)
User program	Language	Ladder diagrams
	Standard instruction configuration	Number of instructions: 5,000
	Arithmetic instructions for LREAL data	Number of instructions: 200
	Trigonometric instructions for LREAL data	Number of instructions: 100
Motion control processing	Number of axes	4

Note Total length of cables connected to the built-in EtherCAT port is 10 [m].

● I/O Refresh Time

I/O refresh overhead time:

The I/O refresh overhead time is 65 μs because the EtherCAT slaves and CJ-series Units are connected.

EtherCAT slave processing time:

The following table gives the Tout (output processing time per byte), Tin (input processing time per byte), and Tref (refresh processing time per byte) values when an NJ501-□□□□ CPU Unit is used.

Tout: Output processing time per byte [μs]	Tin: Input processing time per byte [μs]	Tref: Refresh processing time per byte [μs]	Tec: EtherCAT communications adjustment time [μs]
0.004	0.011	0.082	55

The following table gives the Dout (output processing data size), Din (input processing data size), and Dinout (larger of the input and output data size) values of the GX-ID1611 (Ver. 1.1) Input Slave, GX-OD1611 (Ver. 1.1) Output Slave, and R88D-□□ (Ver. 2.1) Servomotors.

EtherCAT slave	Dout: Output processing data size in bytes	Din: Input processing data size in bytes	Dinout: Larger of the input and output data size
GX-ID1611 (Ver. 1.1)	0	3	3
GX-OD1611 (Ver. 1.1)	2	1	2
R88D-□□ (Ver. 2.1)	23	26	26

Total number of bytes are given below for Dout, Din and Dinout.

Dout = 2 + 23 × 4 = 94 [byte]

Din = 3 + 1 + 26 × 4 = 108 [byte]

Dinout = 3 + 2 + 26 × 4 = 109 [byte]

From these values, the I/O refresh time is calculated by the following formula.

I/O refresh processing time

= EtherCAT slave processing time

= Tout × Dout + Tin × Din + Tref × Dinout + (1.24 × Snum + 0.01 × Clen − Tec)

= 0.004 × 94 + 0.011 × 108 + 0.082 × 109 + (1.24 × 6 + 0.01 × 10) − 55

= 0.376 + 1.188 + 8.938 + (7.44 + 0.1) − 55

= −36.958 [μs]

Because the result that is calculated from the above formula is a negative number, the EtherCAT slave processing time is regarded as 0 μs.

CJ-series Unit processing time:

The following table gives the I/O refresh time per Unit for the CJ1W-ID211 DC Input Unit, CJ1W-OD211 Transistor Output Unit, CJ1W-AD042 Analog Input Unit, CJ1W-DA021 Analog Output Unit, and CJ1W-SCU42 Serial Communications Unit that are used on the CPU Rack.

Model	I/O refresh time per Unit [μs]
CJ1W-ID211	1
CJ1W-OD211	1
CJ1W-AD042	24
CJ1W-DA021	24
CJ1W-SCU42	2.5

Because the number of Units is all one, the CJ-series Unit processing time is regarded as 0 μs.

CJ-series Unit processing time = Σ (I/O refresh time for each CJ-series Unit × Number of Units) - 110
= 1 × 1 + 1 × 1 + 24 × 1 + 24 × 1 + 2.5 × 1 - 110
= -57.5 [μs]

Because the result that is calculated from the above formula is a negative number, the CJ-series Unit processing time is regarded as 0 μs.

The following values of the I/O refresh overhead time, EtherCAT slave processing time, and CJ-series Unit processing time are found by the above calculations.

Item	Value [μs]
I/O refresh overhead time	65
EtherCAT slave processing time	0
CJ-series Unit processing time	0

From these values, the I/O refresh time is calculated by the following formula.

I/O refresh time = I/O refresh overhead time + (Larger of the EtherCAT slave processing time and the CJ-series Unit processing time)
= 65 + 0
= 65 [μs]

● User Program Execution Time

The graphs show the following values.

- Standard instruction configuration
  From the graph of the execution time for standard ladder diagram instructions, the user program execution time of 5,000 instructions for the NJ501-□□□□ is 100 μs.
- Arithmetic instructions for LREAL data
  From the graph of the execution time for ladder diagram arithmetic and trigonometric instructions for LREAL data, the user program execution time of 200 instructions for the NJ501-□□□□ is 20 μs.
- Trigonometric instructions for LREAL data
  From the graph of the execution time for ladder diagram arithmetic and trigonometric instructions for LREAL data, the user program execution time of 100 instructions for the NJ501-□□□□ is 27 μs.

Therefore, the user program execution time is the total of the above values, which is given by the following formula.

User program execution time = 100 + 20 + 27
= 147 [μs]

● Motion Control Processing Time

From the graph of the execution time for motion control processing, the execution time of the motion control processing for four axes for the NJ501-□□□□ with a unit version 1.02 is read as 46 μs.

● Common Processing Time

Because the task is the primary periodic task, the common processing time for the NJ501-□□□□ is 150 μs.

Therefore, the average real processing time of the task is given by the following formula.

Average real processing time of task = I/O refresh processing time + User program execution time + Motion control processing time + Common processing time

= 65 + 147 + 46 + 150

= 408 [µs]

Setting the Task Period

The task period is set based on the average real processing time of the task that is calculated as above. The task is the primary periodic task.

The value of the task period must be larger than the average real processing time of the task that you calculated. More specifically, you should allow sufficient margin and set the task period value to at least 1.1 times as large as the average real processing time of the task.

Task period ≥ Average real processing time of task × 1.1

Because the average real processing time of the task that is calculated above is 408 µs, the task period is set to 500 µs, which is larger than 1.1 times the average time.

The task execution times in the physical Controller depends on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors. The task execution time for a periodic task depends on whether it is interrupted for the execution of tasks with higher execution priorities.
Use the physical Controller and verify the task execution time with the Task Execution Time Monitor.

A-4 System-defined Variables

System-defined variables are assigned specific functions by the system. They are registered in the global variable table, or the local variable table for each POU, in advance.

These variables cannot be changed. Some of the variables start with an underbar and some start with ”P_”.

Some of the system-defined variables are read-only and some are read/write.

You read and write the variables with the user program, with communications from external devices, with the Sysmac Studio, or with an NS/NA-series PT.

Basically, system-defined variables are classified according to the function modules. The variables start with the following category names.

Function module	Category name
System-defined variables for the overall NJ/NX-series Controller	None
PLC Function Module	_PLC
	_CJB
Motion Control Function Module	_MC, _MC1, and _MC2
EtherCAT Master Function Module	_EC
EtherNet/IP Function Module	_EIP, _EIP1, and _EIP2

The variables are described in the tables of this appendix as shown below.

Variable name	Meaning	Function	Data type	Range of values	Reference
This is the system-defined variable name. The prefix gives the category name.	This is the meaning of the variable.	The function of the variable is described.	The data type of the variable is given.	The range of values that the variable can take is given.	The page of the individual system-defined variable specifications table is given.

A version in parentheses in the *Variable name* column is the unit version of the CPU Unit when the system-defined variable was added.

Precautions for Correct Use

There are system-defined variables that are not supported or differ in specifications such as the number of arrays. Refer to *A-5 Specifications for Individual System-defined Variables* for details on the specifications for individual system-defined variables.

A-4-1 System-defined Variables for the Overall NJ/NX-series Controller (No Category)

Functional Classification: Clock

Variable name	Meaning	Function	Data type	Range of values	Reference
_CurrentTime	System Time	Contains the CPU Unit's internal clock data.	DATE_ AND_ TIME	• NX-series CPU Units DT#1970-01-01-00:00:00 to DT#2069-12-31-23:59:59 • NJ-series CPU Units DT#1970-01-01-00:00:00 to DT#2106-02-06-23:59:59	page A-90

Functional Classification: Tasks

Variable name	Meaning	Function	Data type	Range of values	Reference
*TaskName* Active	Task Active Flag	TRUE during task execution. FALSE when task execution is not in progress. **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.	BOOL	TRUE or FALSE	page A-90
*TaskName* LastExecTime	Last Task Execution Time	Contains the task execution time the last time the task was executed (unit: 0.1 µs). **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.	TIME	Depends on data type.	page A-90
*TaskName* MaxExecTime	Maximum Task Execution Time	Contains the maximum value of the task execution time (unit: 0.1 µs). **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.	TIME	Depends on data type.	page A-90
*TaskName* MinExecTime	Minimum Task Execution Time	Contains the minimum value of the task execution time (unit: 0.1 µs). **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.	TIME	Depends on data type.	page A-91
*TaskName* ExecCount	Task Execution Count	Contains the number of executions of the task. If 4294967295 is exceeded, the value returns to 0 and counting is continued. **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.	UDINT	Depends on data type.	page A-91

Variable name	Meaning	Function	Data type	Range of values	Reference
*TaskName* Exceeded	Task Period Exceeded Flag	TRUE if the task period was exceeded. FALSE if task execution was completed within the task period. **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.	BOOL	TRUE or FALSE	page A-91
*TaskName* ExceedCount	Task Period Exceeded Count	Contains the number of times that the period was exceeded. If the present value exceeds the maximum value of the data type, the present value returns to 0 and the count is continued. If 4294967295 is exceeded, the value returns to 0 and counting is continued. **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.	UDINT	Depends on data type.	page A-91

● Functional Classification: Errors

Variable name	Meaning	Function	Data type	Range of values	Reference
_ErrSta	Controller Error Status	TRUE if there is a Controller error. FALSE if there is no Controller error. **Note** Do not use this variable in the user program. There may be a delay in updating it and concurrency problems in relation to the error status of the function module. Use this variable only to access status through communications from an external device. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#C0F0	page A-92
_AlarmFlag	User-defined Error Status	The bit corresponding to the event level is TRUE while there is a user-defined error. Bits 00 to 07 correspond to user fault levels 1 to 8. This variable contains 0000 hex when there is no user-defined error.	WORD	16#0000 to 16#00FF	page A-92

● Functional Classification: SD Memory Card

Variable name	Meaning	Function	Data type	Range of values	Reference
_Card1Ready	SD Memory Card Ready Flag	TRUE when the SD Memory Card is recognized. FALSE when the SD Memory Card is not recognized. TRUE: The Card can be used. FALSE: The Card cannot be used.	BOOL	TRUE or FALSE	page A-92
_Card1Protect	SD Memory Card Write Protected Flag	TRUE when the SD Memory Card is write-protected with the LOCK switch. TRUE: Write protected. FALSE: Not write protected.	BOOL	TRUE or FALSE	page A-93
_Card1Err	SD Memory Card Error Flag	TRUE when an unusable SD Memory Card is inserted or a format error occurs. TRUE: There is an error FALSE: There is no error	BOOL	TRUE or FALSE	page A-93
_Card1Access	SD Memory Card Access Flag	TRUE during SD Memory Card access. TRUE: Card is being accessed. FALSE: Card is not being accessed. The system updates the flag every 100 ms. Because of this, access to the SD Memory Card is shown by this flag with a delay of up to 100 ms. We therefore do not recommend the use of this variable in the user program.	BOOL	TRUE or FALSE	page A-93
_Card1Deteriorated	SD Memory Card Life Warning Flag	TRUE when the life of the SD Memory Card is exceeded. TRUE: The life of the Card has been exceeded. FALSE: The Card can still be used.	BOOL	TRUE or FALSE	page A-93
_Card1PowerFail	SD Memory Card Power Interruption Flag	TRUE when the power supply to the CPU Unit was interrupted during access to the SD Memory Card. TRUE: Power was interrupted during SD Memory Card access. FALSE: Normal	BOOL	TRUE or FALSE	page A-94

Variable name	Member name	Meaning	Function	Data type	Range of values	Reference
_Card1BkupCmd (Ver.1.03)		SD Memory Card Backup Commands		_sBKUP_CMD		page A-94
	ExecBkup	Execute Backup Flag	Change this variable to TRUE to back up Controller data to an SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-94
	CancelBkup	Cancel Backup Flag	Change this variable to TRUE to cancel backing up data to an SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-94
	ExecVefy	Execute Verify Flag	Change this variable to TRUE to compare the Controller data to a backup file in the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-94
	CancelVefy	Cancel Verify Flag	Change this variable to TRUE to cancel comparing the Controller data to a backup file in the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-95
	DirName	Directory Name	Used to specify the directory name in the SD Memory Card for which to back up or verify data. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	STRING(64)	Depends on data type.	page A-95

Variable name / Member name	Meaning	Function	Data type	Range of values	Reference
_Card1BkupSta (Ver. 1.03)	SD Memory Card Backup Status		_sBKUP_ STA		page A-95
Done	Done Flag	TRUE when a backup is completed. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-95
Active	Active Flag	TRUE when a backup is in progress. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-95
Err	Error Flag	TRUE when processing a backup ended in an error. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-96
_Card1VefySta (Ver. 1.03)	SD Memory Card Verify Status		_sVEFY_ STA		page A-96
Done	Done Flag	TRUE when a verification is completed. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-96
Active	Active Flag	TRUE when a verification is in progress. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-96
VefyRslt	Verify Result Flag	TRUE if the data was the same. FALSE if differences were found. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-96
Err	Error Flag	TRUE when processing a verification ended in an error. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-96

● Functional Classification: Backup

Variable name	Meaning	Function	Data type	Range of values	Reference
_BackupBusy (Ver. 1.03)	Backup Function Busy Flag	TRUE when a backup, restoration, or verification is in progress.	BOOL	TRUE or FALSE	page A-97

● Functional Classification: Power Supply

Variable name	Meaning	Function	Data type	Range of values	Reference
_PowerOnHour	Total Power ON Time	Contains the total time that the power has been ON. Contains the total time that the CPU Unit has been ON in 1-hour increments. To reset this value, overwrite the current value with 0. The value is not updated after it reaches 4294967295. This variable is not initialized at startup.	UDINT	0 to 4294967295	page A-97
_PowerOnCount	Power Interruption Count	Contains the number of times that the power supply has been interrupted. The value is incremented by 1 each time the power supply is interrupted after the first time that the power was turned ON. To reset this value, overwrite the current value with 0. The value is not updated after it reaches 4294967295. This variable is not initialized at startup.	UDINT	0 to 4294967295	page A-97
_RetainFail	Retention Failure Flag	TRUE at the following time (failure of retention during power interruptions). • When an error is detected in the battery-backup memory check at startup. FALSE at the following times (no failure of retention during power interruptions). • When no error is detected in the battery-backup memory check at startup. • When the user program is downloaded. • When the Clear All Memory operation is performed. **Note** When the encoder home offset data is not retained, the status is given in the error status of the axis variable, and not in this flag.	BOOL	TRUE or FALSE	page A-98

● Functional Classification: Programming

Variable name	Meaning	Function	Data type	Range of values	Reference
P_On	Always TRUE Flag	This flag is always TRUE.	BOOL	TRUE	page A-98
P_Off	Always FALSE Flag	This flag is always FALSE.	BOOL	FALSE	page A-98
P_CY	Carry Flag	This flag is updated by some instructions.	BOOL	TRUE or FALSE	page A-98
P_First_RunMode	First RUN Period Flag	This flag is TRUE for only one task period after the operating mode of the CPU Unit is changed from PROGRAM mode to RUN mode if execution of the program is in progress. This flag remains FALSE if execution of the program is not in progress. Use this flag to perform initial processing when the CPU Unit begins operation. **Note** You cannot use this system-defined variable inside functions.	BOOL	TRUE or FALSE	page A-98
P_First_Run (Ver.1.08)	First Program Period Flag	This flag is TRUE for one task period after execution of the program starts. Use this flag to perform initial processing when execution of a program starts. **Note** You cannot use this system-defined variable inside functions.	BOOL	TRUE or FALSE	page A-99
P_PRGER	Instruction Error Flag	This flag changes to and remains TRUE when an instruction error occurs in the program or in a function/function block called from the program. After this flag changes to TRUE, it stays TRUE until the user program changes it back to FALSE.	BOOL	TRUE or FALSE	page A-99

● Functional Classification: Communications

Variable name	Meaning	Function	Data type	Range of values	Reference
_Port_numUsingPort	Number of Used Ports	Gives the number of internal logical ports that are currently used. You can use this variable when you debug the user program.	USINT	0 to 32	page A-99
_Port_isAvailable	Network Communications Instruction Enabled Flag	Indicates whether there is an available internal logical port. TRUE when an internal logical port is available. Otherwise FALSE.	BOOL	FALSE or TRUE	page A-99
_FINSTCPConnSta	FINS/TCP Connection Status	Gives the FINS/TCP connection status.	WORD	16#0000 to 16#FFFF	page A-99

Functional Classification: Version

Variable name	Meaning	Function	Data type	Range of values	Reference
_UnitVersion (Ver.1.08)	Unit Version	The unit version of the CPU Unit is stored. The integer part of the unit version is stored in element number 0. The fractional part of the unit version is stored in element number 1. Example 1) If the unit version is 1.08, “1” is stored in element number 0 and “8” is stored in element number 1. Example 2) If the unit version is 1.10, “1” is stored in element number 0 and “10” is stored in element number 1.	ARRAY[0..1] OF USINT	0 to 99	page A-100

Functional Classification: Self-diagnosis

Variable name	Meaning	Function	Data type	Range of values	Reference
_SelfTest_HighTemperature (Ver.1.10)	CPU Unit High Temperature Flag	TRUE when the internal temperature of the CPU Unit is too high.	BOOL	TRUE or FALSE	page A-100
_SelfTest_LowBattery (Ver.1.10)	Low Battery Flag	TRUE when the battery is disconnected or the battery voltage is dropped.	BOOL	TRUE or FALSE	page A-100
_SelfTest_LowFanRevolution (Ver.1.10)	Low FAN Revolution Flag	TRUE when the fan is disconnected or the rotation speed of a fan is decreased. **Note** Always FALSE for an NJ-series CPU Unit.	BOOL	TRUE or FALSE	page A-100

A-4-2 PLC Function Module, Category Name: _PLC

● Functional Classification: Debugging

Variable name	Member	Meaning	Function	Data type	Range of values	Reference
_PLC_TraceSta[0..3]				_sTRACE_STA		page A-101
	.IsStart	Trace Busy Flag	TRUE when a trace starts. **Note** You cannot use this system-defined variable in the user program. It is used only to monitor the status of data tracing from the Sysmac Studio.	BOOL	TRUE or FALSE	page A-101
	.IsComplete	Trace Completed Flag	TRUE when a trace is completed. **Note** You cannot use this system-defined variable in the user program. It is used only to monitor the status of data tracing from the Sysmac Studio.	BOOL	TRUE or FALSE	page A-101
	.IsTrigger	Trace Trigger Monitor Flag	TRUE when the trigger condition is met. FALSE when the next trace starts. **Note** You cannot use this system-defined variable in the user program. It is used only to monitor the status of data tracing from the Sysmac Studio.	BOOL	TRUE or FALSE	page A-101
	.ParamErr	Trace Parameter Error Flag	TRUE when a trace starts, but there is an error in the trace settings. FALSE when the settings are normal. **Note** You cannot use this system-defined variable in the user program. It is used only to monitor the status of data tracing from the Sysmac Studio.	BOOL	TRUE or FALSE	page A-102

● Functional Classification: Errors

Variable name	Meaning	Function	Data type	Range of values	Reference
_PLC_ErrSta	PLC Function Module Error Status	TRUE when there is a Controller error that involves the PLC Function Module. FALSE when there is no Controller error that involves the PLC Function Module. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-102

A-4-3 PLC Function Module, Category Name: _CJB

● Functional Classification: I/O Bus Status

Variable name	Meaning	Function	Data type	Range of values	Reference
_CJB_MaxRackNo	Largest Rack Number	Contains the largest rack number of the Expansion Racks that are detected by the Controller. **Note** You can use this system-defined variable only for NJ-series CPU Units.	UINT	0 to 3 0: Only CPU Rack.	page A-103
_CJB_MaxSlotNo	Largest Slot Number	Contains one higher than the largest slot number with a CJ-series Unit on each of the Racks that are detected by the Controller. **Note** You can use this system-defined variable only for NJ-series CPU Units.	ARRAY [0..3] OF UINT	0 to 10 0: No CJ-series Unit mounted.	page A-103

● Functional Classification: I/O Bus Errors

Variable name	Meaning	Function	Data type	Range of values	Reference
_CJB_ErrSta	I/O Bus Error Status	Gives the I/O bus error status. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NJ-series CPU Units.	WORD	16#0000 to 16#C0F0	page A-103
_CJB_MstrErrSta	I/O Bus Master Error Status	Gives the I/O bus master error status. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NJ-series CPU Units.	WORD	16#0000 to 16#00F0	page A-104
_CJB_UnitErrSta	I/O Bus Unit Error Status	Gives the error status of the I/O Bus Unit. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NJ-series CPU Units.	ARRAY [0..3, 0..9] OF WORD	16#0000 to 16#80F0	page A-104
_CJB_InRespTm	Basic Input Unit Input Response Times	Contains the response times of the Basic Input Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.	ARRAY [0..3, 0..9] OF UNIT	0 to 320	page A-104

● Functional Classification: Auxiliary Area Bits for CJ-series Units

Variable name	Meaning	Function	Data type	Range of values	Reference
_CJB_IOUnitInfo	Basic I/O Unit Information	Shows the status of the Basic I/O Unit alarm output (load short-circuit protection). TRUE: Load short-circuit FALSE: No load short-circuit **Note** You can use this system-defined variable only for NJ-series CPU Units.	ARRAY [0..3, 0..9, 0..7] OF BOOL	TRUE or FALSE	page A-105
_CJB_CBU00InitSta to _CJB_CBU15InitSta	CPU Bus Unit Initializing Flags	The corresponding variable is TRUE during initialization of the CPU Bus Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-105
_CJB_SIO00InitSta to _CJB_SIO95InitSta	Special I/O Unit Initializing Flags	The corresponding variable is TRUE during initialization of the Special I/O Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-105
_CJB_CBU00Restart to _CJB_CBU15Restart	CPU Bus Unit Restart Bits	The CPU Bus Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the CPU Bus Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Flag to TRUE with an instruction, the restart process begins from refresh processing in the next task period. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-106
_CJB_SIO00Restart to _CJB_SIO95Restart	Special I/O Unit Restart Bits	The Special I/O Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the Special I/O Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Flag to TRUE with an instruction, the restart process begins from refresh processing in the next task period. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-106
_CJB_SCU00P1ChgSta to _CJB_SCU00P2ChgSta	Serial Communications Unit 0, Port 1/2 Settings Changing Flags	TRUE when the parameters of the specified port are being changed. FALSE after the parameters are changed. It is also possible for the user to indicate a change in serial port settings by turning ON the corresponding flag through the execution of an instruction or a user operation. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-107
_CJB_SCU15P1ChgSta to _CJB_SCU15P2ChgSta	Serial Communications Units 1 to 15, Port 1/2 Settings Changing Flags		BOOL	TRUE or FALSE	page A-107

A-4-4 Motion Control Function Module, Category Name: _MC

● Functional Classification: Motion Control Functions

Variable name	Meaning	Function	Data type	Range of values	Reference
_MC_ErrSta	Motion Control Function Module Error Status	Shows the status of errors that are detected in the Motion Control Function Module. You can use this variable directly in the user program. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#40F0	page A-108
_MC_ComErrSta	Common Error Status	Shows the status of errors that are detected in common processing for motion control. You can use this variable directly in the user program. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-108
_MC_AX_ErrSta	Axis Error Status	Shows the error status for each axis. The status of up to 256 axes[*1] is shown. You can use this variable directly in the user program. Refer to information on the meanings of the error status bits at the end of this appendix for details.	ARRAY [0..255] OF WORD[*1]	16#0000 to 16#00F0	page A-108
_MC_GRP_ErrSta	Axes Group Error Status	Shows the error status for each axes group. The error status for up to 64 axes groups[*2] is shown. You can use this variable directly in the user program. Refer to information on the meanings of the error status bits at the end of this appendix for details.	ARRAY [0..63] OF WORD[*2]	16#0000 to 16#00F0	page A-108
_MC_COM	Common Variable	Shows the status that is common to the Motion Control Function Module. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members.	_sCOMMON_REF	---	page A-109
_MC_GRP	Axes Group Variables	NX-series CPU Units: Used to specify axes groups and shows multiaxes coordinated control status, and multiaxes coordinated control settings for motion control instructions used for motion control 1. NJ-series CPU Units: Used to specify axes groups and shows multiaxes coordinated control status, and multiaxes coordinated control settings for motion control instructions. When you create an axes group on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members.	NX-series CPU Units: ARRAY[0..63] OF _sGROUP_REF NJ-series CPU Units: ARRAY[0..31] OF _sGROUP_REF	---	page A-109

Variable name	Meaning	Function	Data type	Range of values	Reference
_MC1_GRP	Axes Group Variables	Used to specify axes groups and shows multiaxes coordinated control status, and multiaxes coordinated control settings for motion control instructions used for motion control 1. When you create an axes group on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members. **Note** You can use this system-defined variable only for NX-series CPU Units. You can access the same values of *_MC1_GRP* and *_MC_GRP* if the array element numbers of them are the same.	ARRAY[0..63] OF _sGROUP_REF	---	page A-109
_MC2_GRP	Axes Group Variables	Used to specify axes groups and shows multiaxes coordinated control status, and multiaxes coordinated control settings for motion control instructions used for motion control 2. When you create an axes group on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY[0..63] OF _sGROUP_REF	---	page A-110
_MC_AX	Axis Variables	NX-series CPU Units: Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 1. NJ-series CPU Units: Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions. When you create an axis on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members.	NX-series CPU Units: ARRAY[0..255] OF _sAXIS_REF NJ-series CPU Units: ARRAY[0..63] OF _sAXIS_REF	---	page A-110

Variable name	Meaning	Function	Data type	Range of values	Reference
_MC1_AX	Axis Variables	Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 1. When you create an axis on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members. **Note** You can use this system-defined variable only for NX-series CPU Units. You can access the same values of *_MC1_AX* and *_MC_AX* if the array element numbers of them are the same.	ARRAY[0..255] OF _sAXIS_REF	---	page A-110
_MC2_AX	Axis Variables	Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 2. When you create an axis on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY[0..255] OF _sAXIS_REF	---	page A-111

*1 For NJ-series CPU Units, the error status of up to 64 axes is shown and the data type is ARRAY [0..63] OF WORD.
*2 For NJ-series CPU Units, the error status of up to 32 axes groups is shown and the data type is ARRAY [0..31] OF WORD.

A-4-5 EtherCAT Master Function Module, Category Name: _EC

● Functional Classification: EtherCAT Communications Errors

Variable name	Meaning	Function	Data type	Range of values	Reference
_EC_ErrSta	Built-in EtherCAT Error	This system-defined variable provides the collective status of errors in the EtherCAT Master Function Module. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-112
_EC_PortErr	Communications Port Error	This system-defined variable provides the collective status of errors in the communications ports for the EtherCAT master. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-112
_EC_MstrErr	Master Error	This system-defined variable provides the collective status of EtherCAT master errors and slave errors detected by the EtherCAT master. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-112
_EC_SlavErr	Slave Error	This system-defined variable provides the collective status of all the error status for EtherCAT slaves. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-112
_EC_SlavErrTbl	Slave Error Table	This system-defined variable gives the error status for each EtherCAT slave. The error status is given for each slave in the actual system configuration. This variable array indicates slaves in which there are errors. Status is provided for each EtherCAT slave node address (1 to 512)*1. Refer to information on the meanings of the error status bits at the end of this appendix for details.	Array [1..512] OF WORD*1	16#0000 to 16#00F0	page A-113
_EC_MacAdrErr	MAC Address Error	TRUE if there is an illegal MAC address.	BOOL	TRUE or FALSE	page A-113
_EC_LanHwErr	Communications Controller Error	TRUE if there is a communications controller hardware error.	BOOL	TRUE or FALSE	page A-113
_EC_LinkOffErr	Link OFF Error	TRUE if the communications controller link is not established.	BOOL	TRUE or FALSE	page A-113
_EC_NetCfgErr	Network Configuration Information Error	TRUE if there is illegal network configuration information.	BOOL	TRUE or FALSE	page A-113
_EC_NetCfgCmpErr	Network Configuration Verification Error	TRUE if the network configuration information does not match the actual network configuration.	BOOL	TRUE or FALSE	page A-114
_EC_NetTopologyErr	Network Configuration Error	TRUE if there is a network configuration error (too many devices connected or ring connection).	BOOL	TRUE or FALSE	page A-114
_EC_PDCommErr	Process Data Communications Error	TRUE if there is an unexpected slave disconnection or connection or if a slave WDT error is detected during process data communications.	BOOL	TRUE or FALSE	page A-114
_EC_PDTimeoutErr	Process Data Reception Timeout Error	TRUE if a timeout occurs while receiving process data.	BOOL	TRUE or FALSE	page A-114

Variable name	Meaning	Function	Data type	Range of values	Reference
_EC_PDSendErr	Process Data Transmission Error	TRUE if there is a process data transmission error (cannot send within the process data communications cycle or transmission jitter is over the limit).	BOOL	TRUE or FALSE	page A-114
_EC_SlavAdrDupErr	Slave Node Address Duplicated Error	TRUE if the same node address is set for more than one slave.	BOOL	TRUE or FALSE	page A-114
_EC_SlavInitErr	Slave Initialization Error	TRUE if there is an error in an initialization command addressed to a slave.	BOOL	TRUE or FALSE	page A-115
_EC_SlavAppErr	Slave Application Error	TRUE if there is an error in the slave's application status register.	BOOL	TRUE or FALSE	page A-115
_EC_MsgErr	EtherCAT Message Error	TRUE when a message is sent to a slave that does not support messages or when there is an error in the format of the response to a message that was sent to a slave.	BOOL	TRUE or FALSE	page A-115
_EC_SlavEmergErr	Emergency Message Detected	TRUE if the master detects an emergency message that was sent by a slave.	BOOL	TRUE or FALSE	page A-115
_EC_CommErrTbl	Communications Error Slave Table	Slaves are given in the table in the order of slave node addresses. The corresponding slave element is TRUE if the master detected an error for the slave.	Array [1..512] OF BOOL*2	TRUE or FALSE	page A-115
_EC_CycleExceeded	EtherCAT Communications Cycle Exceeded	TRUE if the CPU Unit cannot establish communications within the set communications period at startup. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-116

*1 For NJ-series CPU Units, the node address is 1 to 192 and the data type is Array [1..192] OF WORD.
*2 For NJ-series CPU Units, the data type is Array [1..192] OF BOOL.

Additional Information

Typical Relationships for the Built-in EtherCAT Error Flags

Variable Name	Meaning	Variable Name	Meaning	Variable Name	Meaning	Event level
_EC_ErrSta	Built-in EtherCAT Error	_EC_PortErr	Communi-cations Port Error	_EC_MacAdrErr	MAC Address Error	Partial fault level
				_EC_LanHwErr	Communications Controller Error	
				_EC_LinkOffErr	Link OFF Error	
		_EC_MstrErr	Master Error	_EC_NetCfgErr	Network Configura-tion Information Error	Minor fault level
				_EC_NetCfgCmpErr	Network Configura-tion Verification Error	
				_EC_NetTopologyErr	Network Configura-tion Error	
				_EC_PDCommErr	Process Data Com-munications Error	
				_EC_PDTimeoutErr	Process Data Recep-tion Timeout Error	
				_EC_PDSendErr	Process Data Trans-mission Error	
				_EC_SlavAdrDupErr	Slave Node Address Duplicated Error	
				_EC_SlavInitErr	Slave Initialization Error	
				_EC_SlavAppErr	Slave Application Error	
				_EC_CommErrTbl	Communications Error Slave Table	
				_EC_CycleExceeded	EtherCAT Communi-cations Cycle Exceeded	
				_EC_MsgErr	EtherCAT Message Error	Observation
				_EC_SlavEmergErr	Emergency Message Detected	
		_EC_SlavErr	Slave Error	_EC_SlavErrTbl	Slave Error Table	Defined by the slave.

Note The values of all system-defined variables that are related to errors in EtherCAT communications do not change until the cause of the error is removed and then the error in the Controller is reset with the trouble-shooting functions of the Sysmac Studio or the ResetECError instruction.

● Functional Classification: EtherCAT Communications Status

Variable name	Meaning	Function	Data type	Range of values	Reference
_EC_RegSlavTbl	Registered Slave Table	This table indicates the slaves that are registered in the network configuration information. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave is registered.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-116
_EC_EntrySlavTbl	Network Connected Slave Table	This table indicates which slaves are connected to the network. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave has entered the network.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-116
_EC_MBXSlavTbl	Message Communications Enabled Slave Table	This table indicates the slaves that can perform message communications. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if message communications are enabled for it (pre-operational, safe-operation, or operational state). **Note** Use this variable to confirm that message communications are possible for the relevant slave before you execute message communications with an EtherCAT slave.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-116
_EC_PDSlavTbl	Process Data Communicating Slave Table	This table indicates the slaves that are performing process data communications. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if process data of the corresponding slave is enabled (operational) for both slave inputs and outputs. **Note** Use this variable to confirm that the data for the relevant slave is valid before controlling an EtherCAT slave.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-117
_EC_DisconnSlavTbl	Disconnected Slave Table	Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave was disconnected.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-117
_EC_DisableSlavTbl	Disabled Slave Table	Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave is disabled.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-117
_EC_PDActive	Process Data Communications Status	TRUE when process data communications are performed with all slaves*. * Disabled slaves are not included.	BOOL	TRUE or FALSE	page A-117
_EC_PktMonStop	Packet Monitoring Stopped	TRUE when packet monitoring is stopped.	BOOL	TRUE or FALSE	page A-118
_EC_LinkStatus	Link Status	TRUE if the communications controller link status is Link ON.	BOOL	TRUE or FALSE	page A-118
_EC_PktSaving	Saving Packet Data File	Shows whether a packet data file is being saved. TRUE: Packet data file being saved. FALSE: Packet data file not being saved.	BOOL	TRUE or FALSE	page A-118
_EC_InDataInvalid	Input Data Invalid	TRUE when process data communications performed in the primary periodic task are not normal and the input data is not valid.	BOOL	TRUE or FALSE	page A-118

Variable name	Meaning	Function	Data type	Range of values	Reference
_EC_InData1Invalid	Input Data1 Invalid	TRUE when process data communications performed in the primary periodic task are not normal and the input data is not valid. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-118
_EC_InData2Invalid	Input Data2 Invalid	TRUE when process data communications performed in the priority-5 periodic task are not normal and the input data is not valid. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-119

*1 For NJ-series CPU Units, the data type is Array [1..192] OF BOOL.

Note All system-defined variables that are related to the status of EtherCAT communications give the current status.

A-4-6 EtherNet/IP Function Module, Category Name: _EIP

● Functional Classification: EtherNet/IP Communications Errors

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_ErrSta	Built-in EtherNet/IP Error	This is the error status variable for the built-in EtherNet/IP port. NX-series CPU Units: Represents the collective status of the following error flags. • *_EIP1_PortErr* (Communications Port1 Error) • *_EIP2_PortErr* (Communications Port2 Error) • *_EIP1_CipErr* (CIP Communications1 Error) • *_EIP2_CipErr* (CIP Communications2 Error) • *_EIP_TcpAppErr* (TCP Application Communications Error) NJ-series CPU Units: Represents the collective status of the following error flags. • *_EIP_PortErr* (Communications Port Error) • *_EIP_CipErr* (CIP Communications Error) • *_EIP_TcpAppErr* (TCP Application Communications Error) **Note** Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-120

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_PortErr	Communications Port Error	This is the error status variable for the communications port. NX-series CPU Units: Represents the collective status of the following error flags. • *_EIP1_MacAdrErr* (Port1 MAC Address Error) • *_EIP1_LanHwErr* (Port1 Communications Controller Error) • *_EIP1_EtnCfgErr* (Port1 Basic Ethernet Setting Error) • *_EIP1_IPAdrCfgErr* (Port1 IP Address Setting Error) • *_EIP1_IPAdrDupErr* (Port1 IP Address Duplication Error) • *_EIP1_BootpErr* (Port1 BOOTP Server Error) • *_EIP_DNSCfgErr* (DNS Setting Error) • *_EIP_DNSSrvErr* (DNS Server Connection Error) • *_EIP_IPRTblErr* (IP Route Table Error) NJ-series CPU Units: Represents the collective status of the following error flags. • *_EIP_MacAdrErr* (MAC Address Error) • *_EIP_LanHwErr* (Communications Controller Error) • *_EIP_EtnCfgErr* (Basic Ethernet Setting Error) • *_EIP_IPAdrCfgErr* (IP Address Setting Error) • *_EIP_IPAdrDupErr* (IP Address Duplication Error) • *_EIP_BootpErr* (BOOTP Server Error) • *_EIP_IPRTblErr* (IP Route Table Error) **Note** If a Link OFF Detected or Built-in EtherNet/IP Processing Error occurs, it is recorded in the event log and then the corresponding bit turns ON. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-120

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_PortErr	Communications Port1 Error	This is the error status variable for the communications port. It represents the collective status of the following error flags. • *_EIP1_MacAdrErr* (Port1 MAC Address Error) • *_EIP1_LanHwErr* (Port1 Communications Controller Error) • *_EIP1_EtnCfgErr* (Port1 Basic Ethernet Setting Error) • *_EIP1_IPAdrCfgErr* (Port1 IP Address Setting Error) • *_EIP1_IPAdrDupErr* (Port1 IP Address Duplication Error) • *_EIP1_BootpErr* (Port1 BOOTP Server Error) • *_EIP_DNSCfgErr* (DNS Setting Error) • *_EIP_DNSSrvErr* (DNS Server Connection Error) • *_EIP_IPRTblErr* (IP Route Table Error) **Note** If a Link OFF Detected or Built-in EtherNet/IP Processing Error occurs, it is recorded in the event log and then the corresponding bit turns ON. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NX-series CPU Units.	WORD	16#0000 to 16#00F0	page A-120
_EIP2_PortErr	Communications Port2 Error	This is the error status variable for the communications port. It represents the collective status of the following error flags. • *_EIP2_MacAdrErr* (Port2 MAC Address Error) • *_EIP2_LanHwErr* (Port2 Communications Controller Error) • *_EIP2_EtnCfgErr* (Port2 Basic Ethernet Setting Error) • *_EIP2_IPAdrCfgErr* (Port2 IP Address Setting Error) • *_EIP2_IPAdrDupErr* (Port2 IP Address Duplication Error) • *_EIP2_BootpErr* (Port2 BOOTP Server Error) • *_EIP_DNSCfgErr* (DNS Setting Error) • *_EIP_DNSSrvErr* (DNS Server Connection Error) • *_EIP_IPRTblErr* (IP Route Table Error) **Note** If a Link OFF Detected or Built-in EtherNet/IP Processing Error occurs, it is recorded in the event log and then the corresponding bit turns ON. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NX-series CPU Units.	WORD	16#0000 to 16#00F0	page A-120

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_CipErr	CIP Communications Error	This is the error status variable for CIP communications. NX-series CPU Units: Represents the collective status of the following error flags. • *_EIP1_IdentityErr* (CIP Communications1 Identity Error) • *_EIP1_TDLinkCfgErr* (CIP Communications1 Tag Data Link Setting Error) • *_EIP1_TDLinkOpnErr* (CIP Communications1 Tag Data Link Connection Failed) • *_EIP1_TDLinkErr* (CIP Communications1 Tag Data Link Communications Error) • *_EIP1_TagAdrErr* (CIP Communications1 Tag Name Resolution Error) • *_EIP1_MultiSwONErr* (CIP Communications1 Multiple Switches ON Error) NJ-series CPU Units: Represents the collective status of the following error flags. • *_EIP_IdentityErr* (Identity Error) • *_EIP_TDLinkCfgErr* (Tag Data Link Setting Error) • *_EIP_TDLinkOpnErr* (Tag Data Link Connection Failed) • *_EIP_TDLinkErr* (Tag Data Link Communications Error) • *_EIP_TagAdrErr* (Tag Name Resolution Error) • *_EIP_MultiSwONErr* (Multiple Switches ON Error) **Note** If a Tag Name Resolution Error occurs, it is recorded in the event log and this variable changes to TRUE. Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-122

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_CipErr	CIP Communications1 Error	This is the error status variable for CIP communications 1. It represents the collective status of the following error flags. • *_EIP1_IdentityErr* (CIP Communications1 Identity Error) • *_EIP1_TDLinkCfgErr* (CIP Communications1 Tag Data Link Setting Error) • *_EIP1_TDLinkOpnErr* (CIP Communications1 Tag Data Link Connection Failed) • *_EIP1_TDLinkErr* (CIP Communications1 Tag Data Link Communications Error) • *_EIP1_TagAdrErr* (CIP Communications1 Tag Name Resolution Error) • *_EIP1_MultiSwONErr* (CIP Communications1 Multiple Switches ON Error) **Note** If a Tag Name Resolution Error occurs, it is recorded in the event log and this variable changes to TRUE. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NX-series CPU Units.	WORD	16#0000 to 16#00F0	page A-122
_EIP2_CipErr	CIP Communications2 Error	This is the error status variable for CIP communications 2. It represents the collective status of the following error flags. • *_EIP2_IdentityErr* (CIP Communications2 Identity Error) • *_EIP2_TDLinkCfgErr* (CIP Communications2 Tag Data Link Setting Error) • *_EIP2_TDLinkOpnErr* (CIP Communications2 Tag Data Link Connection Failed) • *_EIP2_TDLinkErr* (CIP Communications2 Tag Data Link Communications Error) • *_EIP2_TagAdrErr* (CIP Communications2 Tag Name Resolution Error) • *_EIP2_MultiSwONErr* (CIP Communications2 Multiple Switches ON Error) **Note** If a Tag Name Resolution Error occurs, it is recorded in the event log and this variable changes to TRUE. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NX-series CPU Units.	WORD	16#0000 to 16#00F0	page A-123

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TcpAppErr	TCP Application Communications Error	This is the error status variable for TCP application communications. It represents the collective status of the following error flags. • *_EIP_TcpAppCfgErr* (TCP/IP Setting Error) • *_EIP_NTPSrvErr* (NTP Server Connection Error) **Note** Refer to information on the meanings of the error status bits at the end of this appendix for details.	WORD	16#0000 to 16#00F0	page A-123
_EIP_MacAdrErr	MAC Address Error	NX-series CPU Units: Indicates that an error occurred when the MAC address was read on the communications port 1 at startup. TRUE: Error FALSE: Normal NJ-series CPU Units: Indicates that an error occurred when the MAC address was read at startup. TRUE: Error FALSE: Normal	BOOL	TRUE or FALSE	page A-123
_EIP1_MacAdrErr	Port1 MAC Address Error	Indicates that an error occurred when the MAC address was read on the communications port 1 at startup. TRUE: Error FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-124
_EIP2_MacAdrErr	Port2 MAC Address Error	Indicates that an error occurred when the MAC address was read on the communications port 2 at startup. TRUE: Error FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-124
_EIP_LanHwErr	Communications Controller Error	NX-series CPU Units: Indicates that a communications controller failure occurred on the communications port 1. TRUE: Failure FALSE: Normal NJ-series CPU Units: Indicates that a communications controller failure occurred. TRUE: Failure FALSE: Normal	BOOL	TRUE or FALSE	page A-124
_EIP1_LanHwErr	Port1 Communications Controller Error	Indicates that a communications controller failure occurred on the communications port 1. TRUE: Failure FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-124

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_LanHwErr	Port2 Communications Controller Error	Indicates that a communications controller failure occurred on the communications port 2. TRUE: Failure FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-125
_EIP_EtnCfgErr	Basic Ethernet Setting Error	NX-series CPU Units: Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 1 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal NJ-series CPU Units: Indicates that the Ethernet communications speed setting (Speed/Duplex) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal	BOOL	TRUE or FALSE	page A-125
_EIP1_EtnCfgErr	Port1 Basic Ethernet Setting Error	Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 1 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-125
_EIP2_EtnCfgErr	Port2 Basic Ethernet Setting Error	Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 2 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-125

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_IPAdrCfgErr	IP Address Setting Error	NX-series CPU Units: Indicates the IP address setting errors for the communications port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The DNS settings are not correct. FALSE: Normal NJ-series CPU Units: Indicates the IP address setting errors. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The DNS settings are not correct. FALSE: Normal	BOOL	TRUE or FALSE	page A-126
_EIP1_IPAdrCfgErr	Port1 IP Address Setting Error	Indicates the IP address setting errors for the communications port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The DNS settings are not correct. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-126
_EIP2_IPAdrCfgErr	Port2 IP Address Setting Error	Indicates the IP address setting errors for the communications port 2. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The DNS settings are not correct. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-126

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_IPAdrDupErr	IP Address Duplication Error	NX-series CPU Units: Indicates that the same IP address is assigned to more than one node for the communications port 1. TRUE: Duplication occurred. FALSE: Other than the above. NJ-series CPU Units: Indicates that the same IP address is assigned to more than one node. TRUE: Duplication occurred. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-127
_EIP1_IPAdrDupErr	Port1 IP Address Duplication Error	Indicates that the same IP address is assigned to more than one node for the communications port 1. TRUE: Duplication occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-127
_EIP2_IPAdrDupErr	Port2 IP Address Duplication Error	Indicates that the same IP address is assigned to more than one node for the communications port 2. TRUE: Duplication occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-127
_EIP_DNSCfgErr	DNS Setting Error	Indicates that the DNS or hosts settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-127
_EIP_BootpErr	BOOTP Server Error	NX-series CPU Units: Indicates that a BOOTP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. NJ-series CPU Units: Indicates that a BOOTP server connection failure occurred. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server.	BOOL	TRUE or FALSE	page A-128
_EIP1_BootpErr	Port1 BOOTP Server Error	Indicates that a BOOTP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-128

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_BootpErr	Port2 BOOTP Server Error	Indicates that a BOOTP server connection failure occurred on the communications port 2. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-128
_EIP_IPRTblErr	IP Route Table Error	NX-series CPU Units: Indicates that the default gateway settings or IP router table settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** For NX-series CPU Units, the host setting errors will be indicated to *_EIP_DNSCfgErr*. NJ-series CPU Units: Indicates that the IP router table or hosts settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** For NJ-series CPU Units, the default gateway setting errors will be indicated to *_EIP_IPAdrCfgErr*.	BOOL	TRUE or FALSE	page A-129
_EIP_IdentityErr	Identity Error	NX-series CPU Units: Indicates that the identity information for CIP communications 1 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal NJ-series CPU Units: Indicates that the identity information (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal	BOOL	TRUE or FALSE	page A-129
_EIP1_IdentityErr	CIP Communications1 Identity Error	Indicates that the identity information for CIP communications 1 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-129
_EIP2_IdentityErr	CIP Communications2 Identity Error	Indicates that the identity information for CIP communications 2 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-130

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TDLinkCfgErr	Tag Data Link Setting Error	NX-series CPU Units: Indicates that the tag data link settings for CIP communications 1 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal NJ-series CPU Units: Indicates that the tag data link settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal	BOOL	TRUE or FALSE	page A-130
_EIP1_TDLinkCfgErr	CIP Communications1 Tag Data Link Setting Error	Indicates that the tag data link settings for CIP communications 1 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-130
_EIP2_TDLinkCfgErr	CIP Communications2 Tag Data Link Setting Error	Indicates that the tag data link settings for CIP communications 2 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-130
_EIP_TDLinkOpnErr	Tag Data Link Connection Failed	NX-series CPU Units: Indicates that establishing a tag data link connection for CIP communications 1 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above. NJ-series CPU Units: Indicates that establishing a tag data link connection failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-131

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_TDLinkOpnErr	CIP Communications1 Tag Data Link Connection Failed	Indicates that establishing a tag data link connection for CIP communications 1 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-131
_EIP2_TDLinkOpnErr	CIP Communications2 Tag Data Link Connection Failed	Indicates that establishing a tag data link connection for CIP communications 2 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-131
_EIP_TDLinkErr	Tag Data Link Communications Error	NX-series CPU Units: Indicates that a timeout occurred in a tag data link connection for CIP communications 1. TRUE: A timeout occurred. FALSE: Other than the above. NJ-series CPU Units: Indicates that a timeout occurred in a tag data link connection. TRUE: A timeout occurred. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-132
_EIP1_TDLinkErr	CIP Communications1 Tag Data Link Communications Error	Indicates that a timeout occurred in a tag data link connection for CIP communications 1. TRUE: A timeout occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-132
_EIP2_TDLinkErr	CIP Communications2 Tag Data Link Communications Error	Indicates that a timeout occurred in a tag data link connection for CIP communications 2. TRUE: A timeout occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-132

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TagAdrErr	Tag Name Resolution Error	NX-series CPU Units: Indicates that tag resolution for CIP communications 1 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. NJ-series CPU Units: Indicates that tag resolution failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-133
_EIP1_TagAdrErr	CIP Communications1 Tag Name Resolution Error	Indicates that tag resolution for CIP communications 1 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-133

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_TagAdrErr	CIP Communications2 Tag Name Resolution Error	Indicates that tag resolution for CIP communications 2 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-134
_EIP_MultiSwONErr	Multiple Switches ON Error	NX-series CPU Units: Indicates that more than one switch turned ON at the same time in CIP communications 1. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. NJ-series CPU Units: Indicates that more than one switch turned ON at the same time. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-134
_EIP1_MultiSwONErr	CIP Communications1 Multiple Switches ON Error	Indicates that more than one switch turned ON at the same time in CIP communications 1. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-134
_EIP2_MultiSwONErr	CIP Communications2 Multiple Switches ON Error	Indicates that more than one switch turned ON at the same time in CIP communications 2. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-134
_EIP_TcpAppCfgErr	TCP/IP Setting Error	TRUE: At least one of the set values for a TCP/IP application (FTP, NTP, SNMP) is incorrect. Or, a read operation failed. FALSE: Normal	BOOL	TRUE or FALSE	page A-135
_EIP_NTPSrvErr	NTP Server Connection Error	TRUE: The NTP client failed to connect to the server (timeout). FALSE: NTP is not set or the connection was successful.	BOOL	TRUE or FALSE	page A-135

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_DNSSrvErr	DNS Server Connection Error	TRUE: The DNS client failed to connect to the server (timeout). FALSE: DNS is not enabled. Or, DNS is enabled and the connection was successful.	BOOL	TRUE or FALSE	page A-135

Hierarchical Relationship of System-defined Variables Related to EtherNet/IP Errors in the NJ-series CPU Unit

The system-defined variables that are related to EtherNet/IP errors have the following hierarchical relationship. For example, if the value of any of the *_EIP_PortErr*, *_EIP_CipErr*, or *_EIP_TcpAppErr* variables in the second level is TRUE, then the *_EIP_ErrSta* variable in the first level also changes to TRUE. Therefore, you can check the values of system-defined variables in a higher level to see if an error has occurred for a variable in a lower level.

Level 1		Level 2		Level 3	
Variable	Name	Variable	Name	Variable	Name
_EIP_ErrSta	Built-in EtherNet/IP Error	_EIP_PortErr	Communications Port Error	_EIP_MacAdrErr	MAC Address Error
				_EIP_LanHwErr	Communications Controller Error
				_EIP_EtnCfgErr	Basic Ethernet Settings Error
				_EIP_IPAdrCfgErr	IP Address Setting Error
				_EIP_IPAdrDupErr	IP Address Duplication Error
				_EIP_BootpErr	BOOTP Server Error
				_EIP_DNSSrvErr	DNS Server Connection Error
				_EIP_IPRTblErr	IP Route Table Error
		_EIP_CipErr	CIP Communications Error	_EIP_IdentityErr	Identity Error
				_EIP_TDLinkCfgErr	Tag Data Link Setting Error
				_EIP_TDLinkOpnErr	Tag Data Link Connection Failed
				_EIP_TDLinkErr	Tag Data Link Communications Error
				_ EIP_TagAdrErr	Tag Name Resolution Error
				_EIP_MultiSwONErr	Multiple Switches ON Error
		_EIP_TcpAppErr	TCP Application Communications Error	_EIP_TcpAppCfgErr	TCP/IP Setting Error
				_EIP_NTPSrvErr	NTP Server Connection Error

Hierarchical Relationship of System-defined Variables Related to EtherNet/IP Errors in the NX-series CPU Unit

The system-defined variables that are related to EtherNet/IP errors have the following hierarchical relationship. For example, if the value of any of the *_EIP1_PortErr*, *_EIP2_PortErr*, *EIP1_CipErr*, *_EIP2_CipErr*, and *_EIP_TcpAppErr* variables in the second level is TRUE, then the *_EIP_ErrSta* variable in the first level also changes to TRUE. Therefore, you can check the values of system-defined variables in a higher level to see if an error has occurred for a variable in a lower level.

Level 1		Level 2		Level 3	
Variable	**Name**	**Variable**	**Name**	**Variable**	**Name**
_EIP_ErrSta	Built-in EtherNet/IP Error	_EIP1_PortErr	Communications Port1 Error	_EIP1_MacAdrErr	Port1 MAC Address Error
				_EIP1_LanHwErr	Port1 Communications Controller Error
				_EIP1_EtnCfgErr	Port1 Basic Ethernet Setting Error
				_EIP1_IPAdrCfgErr	Port1 IP Address Setting Error
				_EIP1_IPAdrDupErr	Port1 IP Address Duplication Error
				_EIP1_BootpErr	Port1 BOOTP Server Error
				_EIP_DNSCfgErr	DNS Setting Error
				_EIP_DNSSrvErr	DNS Server Connection Error
				_EIP_IPRTblErr	IP Route Table Error
		_EIP2_PortErr	Communications Port2 Error	_EIP2_MacAdrErr	Port2 MAC Address Error
				_EIP2_LanHwErr	Port2 Communications Controller Error
				_EIP2_EtnCfgErr	Port2 Basic Ethernet Setting Error
				_EIP2_IPAdrCfgErr	Port2 IP Address Setting Error
				_EIP2_IPAdrDupErr	Port2 IP Address Duplication Error
				_EIP2_BootpErr	Port2 BOOTP Server Error
				_EIP_DNSCfgErr	DNS Setting Error
				_EIP_DNSSrvErr	DNS Server Connection Error
				_EIP_IPRTblErr	IP Route Table Error
		_EIP1_CipErr	CIP Communications1 Error	_EIP1_IdentityErr	CIP Communications1 Identity Error
				_EIP1_TDLinkCfgErr	CIP Communications1 Tag Data Link Setting Error
				_EIP1_TDLinkOpnErr	CIP Communications1 Tag Data Link Connection Failed
				_EIP1_TDLinkErr	CIP Communications1 Tag Data Link Communications Error
				_EIP1_TagAdrErr	CIP Communications1 Tag Name Resolution Error
				_EIP1_MultiSwONErr	CIP Communications1 Multiple Switches ON Error
		_EIP2_CipErr	CIP Communications2 Error	_EIP2_IdentityErr	CIP Communications2 Identity Error
				_EIP2_TDLinkCfgErr	CIP Communications2 Tag Data Link Setting Error
				_EIP2_TDLinkOpnErr	CIP Communications2 Tag Data Link Connection Failed
				_EIP2_TDLinkErr	CIP Communications2 Tag Data Link Communications Error
				_EIP2_TagAdrErr	CIP Communications2 Tag Name Resolution Error
				_EIP2_MultiSwONErr	CIP Communications2 Multiple Switches ON Error
		_EIP_TcpAppErr	TCP Application Communications Error	_EIP_TcpAppCfgErr	TCP/IP Setting Error
				_EIP_NTPSrvErr	NTP Server Connection Error

Note You can access the same values of the system-defined variables whose variable names with *_EIP1* and the system-defined variables whose variable names with *_EIP*. For example, you can access the same values of *_EIP1_PortErr* (Communications Port1 Error) and *_EIP_PortErr* (Communcations Port Error).

● Functional Classification: EtherNet/IP Communications Status

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_EtnOnlineSta	Online	NX-series CPU Units: Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 1 (that is, the link is ON, IP address is defined, and there are no errors). TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. NJ-series CPU Units: Indicates that the built-in EtherNet/IP port's communications can be used via the communications port (that is, the link is ON, IP address is defined, and there are no errors). TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status.	BOOL	TRUE or FALSE	page A-135
_EIP1_EtnOnlineSta	Port1 Online	Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 1 (that is, the link is ON, IP address is defined, and there are no errors). TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-136
_EIP2_EtnOnlineSta	Port2 Online	Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 2 (that is, the link is ON, IP address is defined, and there are no errors). TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-136

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TDLinkRunSta	Tag Data Link Communications Status	NX-series CPU Units: Indicates that at least one connection is in normal operation in CIP communications 1. TRUE: Normal operation FALSE: Other than the above. NJ-series CPU Units: Indicates that at least one connection is in normal operation. TRUE: Normal operation FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-136
_EIP1_TDLinkRunSta	CIP Communications1 Tag Data Link Communications Status	Indicates that at least one connection is in normal operation in CIP communications 1. TRUE: Normal operation FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-136
_EIP2_TDLinkRunSta	CIP Communications2 Tag Data Link Communications Status	Indicates that at least one connection is in normal operation in CIP communications 2. TRUE: Normal operation FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-137
_EIP_TDLinkAllRunSta	All Tag Data Link Communications Status	NX-series CPU Units: Indicates that all tag data links are communicating in CIP communications 1. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. NJ-series CPU Units: Indicates that all tag data links are communicating. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection.	BOOL	TRUE or FALSE	page A-137
_EIP1_TDLinkAllRunSta	CIP Communications1 All Tag Data Link Communications Status	Indicates that all tag data links are communicating in CIP communications 1. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-137
_EIP2_TDLinkAllRunSta	CIP Communications2 All Tag Data Link Communications Status	Indicates that all tag data links are communicating in CIP communications 2. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-137

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_RegTargetSta [255]	Registered Target Node Information	NX-series CPU Units: Gives a list of nodes for which built-in EtherNet/IP connections are registered for CIP communications 1. This variable is valid only when the built-in EtherNet/IP port is the originator. *Array[x]* is TRUE: The connection to the node with a target node ID of x is registered. *Array[x]* is FALSE: The connection to the node with a target node ID of x is not registered. NJ-series CPU Units: Gives a list of nodes for which built-in EtherNet/IP connections are registered. This variable is valid only when the built-in EtherNet/IP port is the originator. *Array[x]* is TRUE: The connection to the node with a target node ID of x is registered. *Array[x]* is FALSE: The connection to the node with a target node ID of x is not registered.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-138
_EIP1_RegTargetSta [255]	CIP Communications1 Registered Target Node Information	Gives a list of nodes for which built-in EtherNet/IP connections are registered for CIP communications 1. This variable is valid only when the built-in EtherNet/IP port is the originator. *Array[x]* is TRUE: The connection to the node with a target node ID of x is registered. *Array[x]* is FALSE: The connection to the node with a target node ID of x is not registered. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-138
_EIP2_RegTargetSta [255]	CIP Communications2 Registered Target Node Information	Gives a list of nodes for which built-in EtherNet/IP connections are registered for CIP communications 2. This variable is valid only when the built-in EtherNet/IP port is the originator. *Array[x]* is TRUE: The connection to the node with a target node ID of x is registered. *Array[x]* is FALSE: The connection to the node with a target node ID of x is not registered. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-138

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_EstbTargetSta [255]	Normal Target Node Information	NX-series CPU Units: Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 1. *Array[x]* is TRUE: The connection to the node with a target node ID of x was established normally. *Array[x]* is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. NJ-series CPU Units: Gives a list of nodes that have normally established EtherNet/IP connections. *Array[x]* is TRUE: The connection to the node with a target node ID of x was established normally. *Array[x]* is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-139
_EIP1_EstbTargetSta [255]	CIP Communications1 Normal Target Node Information	Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 1. *Array[x]* is TRUE: The connection to the node with a target node ID of x was established normally. *Array[x]* is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-139
_EIP2_EstbTargetSta [255]	CIP Communications2 Normal Target Node Information	Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 2. *Array[x]* is TRUE: The connection to the node with a target node ID of x was established normally. *Array[x]* is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-139

Variable name	Meaning	Function	Data type	Range of val-ues	Reference
_EIP_TargetPLCModeSta [255]	Target PLC Operating Mode	NX-series CPU Units: Shows the operating status of the target node Controllers that are connected for CIP communications 1, with the built-in EtherNet/IP port as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. If the corresponding Normal Target Node Information is FALSE, the Target Node Controller Operating Information indicates the previous operating status. *Array[x]* is TRUE: This is the operating state of the target Controller with a node address of x. *Array[x]* is FALSE: Other than the above. NJ-series CPU Units: Shows the operating status of the target node Controllers that are connected with the built-in EtherNet/IP port as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. If the corresponding Normal Target Node Information is FALSE, the Target Node Controller Operating Information indicates the previous operating status. *Array[x]* is TRUE: This is the operating state of the target Controller with a node address of x. *Array[x]* is FALSE: Other than the above.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-140
_EIP1_TargetPLCModeSta [255]	CIP Communications1 Target PLC Operating Mode	Shows the operating status of the target node Controllers that are connected for CIP communications 1, with the built-in EtherNet/IP port as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. If the corresponding Normal Target Node Information is FALSE, the Target Node Controller Operating Information indicates the previous operating status. *Array[x]* is TRUE: This is the operating state of the target Controller with a node address of x. *Array[x]* is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-140

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_TargetPLCModeSta [255]	CIP Communications2 Target PLC Operating Mode	Shows the operating status of the target node Controllers that are connected for CIP communications 2, with the built-in EtherNet/IP port as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. If the corresponding Normal Target Node Information is FALSE, the Target Node Controller Operating Information indicates the previous operating status. *Array[x]* is TRUE: This is the operating state of the target Controller with a node address of x. *Array[x]* is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-140
_EIP_TargetPLCErr [255]	Target PLC Error Information	NX-series CPU Units: Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 1, with the built-in EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. *Array[x]* is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. *Array[x]* is FALSE: Other than the above. NJ-series CPU Units: Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected with the built-in EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. *Array[x]* is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. *Array[x]* is FALSE: Other than the above.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-141

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_TargetPLCErr [255]	CIP Communications1 Target PLC Error Information	Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 1, with the built-in EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. *Array[x]* is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. *Array[x]* is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-141
_EIP2_TargetPLCErr [255]	CIP Communications2 Target PLC Error Information	Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 2, with the built-in EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. *Array[x]* is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. *Array[x]* is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-141

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TargetNodeErr [255]	Target Node Error Information	NX-series CPU Units: Indicates that the connection for the Registered Target Node Information for CIP communications 1 was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. *Array[x]* is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. *Array[x]* is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE). NJ-series CPU Units: Indicates that the connection for the Registered Target Node Information was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. *Array[x]* is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. *Array[x]* is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE).	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-142

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_TargetNodeErr [255]	CIP Communications1 Target Node Error Information	Indicates that the connection for the Registered Target Node Information for CIP communications 1 was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. *Array[x]* is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. *Array[x]* is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE). **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-142
_EIP2_TargetNodeErr [255]	CIP Communications2 Target Node Error Information	Indicates that the connection for the Registered Target Node Information for CIP communications 2 was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. *Array[x]* is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. *Array[x]* is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE). **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-143

Variable name		Meaning	Function	Data type	Range of values	Reference
_EIP_NTPResult		NTP Operation Information	Use the GetNTPStatus instruction to read the NTP operation information from the user program. Direct access is not possible.	_sNTP_RESULT		page A-143
	.ExecTime	NTP Last Operation Time	Gives the last time that NTP processing ended normally. The time that was obtained from the NTP server is stored when the time is obtained normally. The time is not stored if it is not obtained from the NTP server normally. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device.	DATE_AND_TIME	Depends on data type.	page A-143
	.ExecNormal	NTP Operation Result	TRUE: Indicates an NTP normal end. FALSE: Indicates that NTP operation ended in an error or has not been executed even once. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device.	BOOL	TRUE or FALSE	page A-143

Additional Information

Communications Status with Target Node

The communications status with the target node of an NJ/NX-series Controller is shown by the combination of the values of four system-defined variables.

- *_EIP_RegTargetSta* (Registered Target Node Information)
- *_EIP_EstbTargetSta* (Normal Target Node Information)
- *_EIP_TargetPLCErr* (Target PLC Error Information)
- *_EIP_TargetNodeErr* (Target Node Error Information)

Value of *_EIP_RegTargetSta*	Value of *_EIP_EstbTargetSta*	Value of *_EIP_TargetPLCErr*	Value of *_EIP_TargetNodeErr*	Communications status with target node
TRUE	TRUE	FALSE	FALSE	A connection with the target node was established normally and there is no error in the target PLC.
		TRUE	TRUE	A connection with the target node was established but there is an error in the target PLC.
	FALSE	Disabled	TRUE	A connection with the target node was not established normally.
FALSE	Disabled	Disabled	Disabled	The information is not valid because the target node is not registered.

For the NX-series Controller, the communications status of CIP communications 1 and CIP communications 2 is shown by the combination of the values of four system-defined variables in the same way as shown in the above table.

- CIP Communications 1
  - *_EIP1_RegTargetSta* (CIP Communications1 Registered Target Node Information)
  - *_EIP1_EstbTargetSta* (CIP Communications1 Normal Target Node Information)
  - *_EIP1_TargetPLCErr* (CIP Communications1 Target PLC Error Information)
  - *_EIP_TargetNodeErr* (Target Node Error Information)
- CIP Communications 2
  - *_EIP2_RegTargetSta* (CIP Communications2 Registered Target Node Information)
  - *_EIP2_EstbTargetSta* (CIP Communications2 Normal Target Node Information)
  - *_EIP2_TargetPLCErr* (CIP Communications2 Target PLC Error Information)
  - *_EIP_TargetNodeErr* (Target Node Error Information)

- **Functional Classification: EtherNet/IP Communications Switches**

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TDLinkStartCmd	Tag Data Link Communications Start Switch	NX-series CPU Units: Change this variable to TRUE to start tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation starts. NJ-series CPU Units: Change this variable to TRUE to start tag data links. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically.	BOOL	TRUE or FALSE	page A-144
_EIP1_TDLinkStartCmd	CIP Communications1 Tag Data Link Communications Start Switch	Change this variable to TRUE to start tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-144
_EIP2_TDLinkStartCmd	CIP Communications2 Tag Data Link Communications Start Switch	Change this variable to TRUE to start tag data links for CIP communications 2. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-144
_EIP_TDLinkStopCmd	Tag Data Link Communications Stop Switch	NX-series CPU Units: Change this variable to TRUE to stop tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation stops. NJ-series CPU Units: Change this variable to TRUE to stop tag data links. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically.	BOOL	TRUE or FALSE	page A-145
_EIP1_TDLinkStopCmd	CIP Communications1 Tag Data Link Communications Stop Switch	Change this variable to TRUE to stop tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-145

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_TDLinkStopCmd	CIP Communications2 Tag Data Link Communications Stop Switch	Change this variable to TRUE to stop tag data links for CIP communications 2. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-145

A-4-7 Meanings of Error Status Bits

The meanings of the individual bits in the following error status are the same.

- *_ErrSta* (Controller Error Status)
- *_PLC_ErrSta* (PLC Function Module Error Status)
- *_CJB_ErrSta* (I/O Bus Error Status)
- *_CJB_MstrErrSta* (I/O Bus Master Error Status)
- *_CJB_UnitErrSta* (I/O Bus Unit Error Status)
- *_MC_ErrSta* (Motion Control Function Module Error Status)
- *_MC_ComErrSta* (MC Common Error Status)
- *_MC_AX_ErrSta* (Axis Error Status)
- *_MC_GRP_ErrSta* (Axes Group Error Status)
- *_EC_ErrSta* (Built-in EtherCAT Error)
- *_EC_PortErr* (Communications Port Error)
- *_EC_MstrErr* (Master Error)
- *_EC_SlavErr* (Slave Error)
- *_EC_SlavErrTbl* (Slave Error Table)
- *_EIP_ErrSta* (Built-in EtherNet/IP Error)
- *EIP_PortErr* (Communications Port Error), *_EIP1_PortErr* (Communications Port1 Error), and *_EIP2_PortErr* (Communications Port2 Error)
- *_EIP_CipErr* (CIP Communications Error), *_EIP1_CipErr* (CIP Communications1 Error), and *_EIP2_CipErr* (CIP Communications2 Error)
- *_EIP_TcpAppErr* (TCP Application Communications Error)

The meaning of the bits are shown in the following table.
However, do not use the following variables in the user program:*_ErrSta* (Controller Error Status), *_CJB_ErrSta* (I/O Bus Error Status), *_CJB_MstrErrSta* (I/O Bus Master Error Status), and *_CJB_UnitErrSta* (I/O Bus Master Unit Status). There may be a delay in updating them and concurrency problems in relation to the error status of the function module.

Use these variables only to access status through communications from an external device.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WORD			–	–	–	–	–	–					–	–	–	–

Bit	Meaning
15	Master-detected error: This bit indicates whether the master detected a Controller error in the Unit/slave for the error status of the Controller error. TRUE: The master detected a Controller error. FALSE: The master has not detected a Controller error. (Valid for *_CJB_U_ErrSta* and *_EC_SlvErrTbl*.)
14	Collective slave error status: This bit indicates if a Controller error was detected for levels (e.g., a Unit, slave, axis, or axes group) that are lower than the event source (i.e., for a function module). TRUE: A Controller error has occurred at a lower level. FALSE: A Controller error has not occurred at a lower level. (Valid for *_CJB_ErrSta, _MC_ErrSta,* and *_EC_ErrSta*.)
8 to 13	Reserved.
7	This bit indicates whether a major fault level Controller error has occurred. TRUE: A major fault level Controller error has occurred. FALSE: A major fault level Controller error has not occurred.
6	This bit indicates whether a partial fault level Controller error has occurred. TRUE: A partial fault level Controller error has occurred. FALSE: A partial fault level Controller error has not occurred.
5	This bit indicates whether a minor fault level Controller error has occurred. TRUE: A minor fault level Controller error has occurred. FALSE: A minor fault level Controller error has not occurred.
4	This bit indicates whether an observation level Controller error has occurred. TRUE: An observation level Controller error has occurred. FALSE: An observation level Controller error has not occurred.
0 to 3	Reserved.

A-5 Specifications for Individual System-defined Variables

The specifications for each system-defined variable are given as described below.

<table>
<tr><td>Variable name</td><td colspan="3">This is the system-defined variable name. The prefix gives the category name.</td><td>Members (for structures)</td><td>The member names are given for structure only.</td></tr>
<tr><td>Meaning</td><td colspan="3">This is the meaning of the variable.</td><td>Global/local</td><td>Global: Global variable, Local: Local variable</td></tr>
<tr><td>Function</td><td colspan="5">The function of the variable is described.</td></tr>
<tr><td>Data type</td><td colspan="3">The data type of the variable is given.</td><td>Range of values</td><td>The range of values that the variable can take is given.</td></tr>
<tr><td>R/W access</td><td>R: Read only,
RW: Read/write</td><td>Retained</td><td>The Retain attribute of the variable is given.</td><td>Network Publish</td><td>The Network Publish attribute of the variable is given.</td></tr>
<tr><td>Usage in user program</td><td>Whether you can use the variable directly in the user program is specified.</td><td>Related instructions</td><td colspan="3">The instructions that are related to the variable are given.
If you cannot use the variable directly in the user program, the instructions that access the variable are given.</td></tr>
</table>

A-5-1 System-defined Variables for the Overall NJ/NX-series Controller (No Category)

● Functional Classification: Clock

Variable name	_CurrentTime				
Meaning	System Time			Global/local	Global
Function	This variable contains the CPU Unit's internal clock data.				
Data type	DATE_AND_TIME			Range of values	• NX-series CPU Units DT#1970-01-01-00:00:00 to DT#2069-12-31-23:59:59 • NJ-series CPU Units DT#1970-01-01-00:00:00 to DT#2106-02-06-23:59:59
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Clock instructions		

● Functional Classification: Tasks

Variable name	_*TaskName*_Active				
Meaning	Task Active Flag			Global/local	Global
Function	TRUE during task execution. FALSE when task execution is not in progress. **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instructions	ActEventTask You can access this variable from the user program only with the following instruction. • Task_IsActive		

Variable name	_*TaskName*_LastExecTime				
Meaning	Last Task Execution Time			Global/local	Global
Function	Contains the task execution time the last time the task was executed (unit: 0.1 µs). **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.				
Data type	TIME			Range of values	Depends on data type.
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instructions	You can access this variable from the user program only with the following instruction. • GetMyTaskStatus		

Variable name	_*TaskName*_MaxExecTime		
Meaning	Maximum Task Execution Time	Global/local	Global
Function	Contains the maximum value of the task execution time (unit: 0.1 µs). **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.		

Data type	TIME			Range of values	Depends on data type.
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not supported.	Related instructions	You can access this variable from the user program only with the following instruction. • GetMyTaskStatus		

Variable name	_*TaskName*_MinExecTime				
Meaning	Minimum Task Execution Time			Global/local	Global
Function	Contains the minimum value of the task execution time (unit: 0.1 μs). **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.				
Data type	TIME			Range of values	Depends on data type.
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instructions	You can access this variable from the user program only with the following instruction. • GetMyTaskStatus		

Variable name	_*TaskName*_ExecCount				
Meaning	Task Execution Count			Global/local	Global
Function	Contains the number of executions of the task. If 4294967295 is exceeded, the value returns to 0 and counting is continued. **Note** You cannot use these system-defined variables in the user program. It is used only to access task status for data tracing from the Sysmac Studio.				
Data type	UDINT			Range of values	Depends on data type.
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instructions	You can access this variable from the user program only with the following instruction. • GetMyTaskStatus		

Variable name	_*TaskName*_Exceeded				
Meaning	Task Period Exceeded Flag			Global/local	Global
Function	TRUE if the task period was exceeded. FALSE if task execution was completed within the task period. **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instructions	You can access this variable from the user program only with the following instruction. • GetMyTaskStatus		

Variable name	_*TaskName*_ExceedCount		
Meaning	Task Period Exceeded Count	Global/local	Global
Function	Contains the number of times that the period was exceeded. If the present value exceeds the maximum value of the data type, the present value returns to 0 and the count is continued. If 4294967295 is exceeded, the value returns to 0 and counting is continued. **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.		

Data type	UDINT			Range of values	Depends on data type.
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instructions	You can access this variable from the user program only with the following instruction. • GetMyTaskStatus		

● Functional Classification: Errors

Variable name	_ErrSta				
Meaning	Controller Error Status			Global/local	Global
Function	TRUE if there is a Controller error. FALSE if there is no Controller error. **Note** Do not use this variable in the user program. There may be a delay in updating it and concurrency problems in relation to the status of the function module. Use this variable only to access status through communications from an external device. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#C0F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	• ResetPLCError • ResetCJBError • ResetECError • ResetMCError • MC_Reset • MC_GroupReset You can access this variable from the user program only with the following instructions. • GetPLCError • GetCJBError • GetECError • GetMCError • GetEIPError			

Variable name	_AlarmFlag				
Meaning	User-defined Error Status			Global/local	Global
Function	The bit corresponding to the event level is TRUE while there is a user-defined error. Bits 00 to 07 correspond to user fault levels 1 to 8. This variable contains 0000 hex when there is no user-defined error.				
Data type	WORD			Range of values	16#0000 to 16#00FF
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• SetAlarm • ResetAlarm • GetAlarm		

● Functional Classification: SD Memory Card

Variable name	_Card1Ready				
Meaning	SD Memory Card Ready Flag			Global/local	Global
Function	TRUE when the SD Memory Card is recognized. FALSE when an SD Memory Card is not recognized. TRUE: The Card can be used. FALSE: The Card cannot be used.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1Protect				
Meaning	SD Memory Card Write Protected Flag			Global/local	Global
Function	TRUE when the SD Memory Card is write-protected with the LOCK switch. TRUE: Write protected. FALSE: Not write protected.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1Err				
Meaning	SD Memory Card Error Flag			Global/local	Global
Function	TRUE when an unusable SD Memory Card is inserted or a format error occurs. TRUE: There is an error FALSE: There is no error				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1Access				
Meaning	SD Memory Card Access Flag			Global/local	Global
Function	TRUE during SD Memory Card access. TRUE: Card is being accessed. FALSE: Card is not being accessed. The system updates the flag every 100 ms. Because of this, access to the SD Memory Card is shown by this flag with a delay of up to 100 ms. We therefore do not recommend the use of this variable in the user program.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1Deteriorated				
Meaning	SD Memory Card Life Warning Flag			Global/local	Global
Function	TRUE when the life of the SD Memory Card is exceeded. If this variable changed to TRUE, replace the SD Memory Card. Read/write operation may fail if the SD Memory Card is not replaced. TRUE: The life of the Card has been exceeded. FALSE: The Card can still be used.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1PowerFail				
Meaning	SD Memory Card Power Interruption Flag			Global/local	Global
Function	TRUE when the power supply to the CPU Unit was interrupted during access to the SD Memory Card. TRUE: Power was interrupted during SD Memory Card access. FALSE: Normal.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*1	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1BkupCmd*1			Member name	.ExecBkup
Meaning	Execute Backup Flag			Global/local	Global
Function	Change this variable to TRUE to back up Controller data to an SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupCmd*1			Member name	.CancelBkup
Meaning	Cancel Backup Flag			Global/local	Global
Function	Change this variable to TRUE to cancel backing up data to an SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupCmd*1			Member name	.ExecVefy
Meaning	Execute Verify Flag			Global/local	Global
Function	Change this variable to TRUE to compare the Controller data to a backup file in the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupCmd[*1]			Member name	.CancelVefy
Meaning	Cancel Verify Flag			Global/local	Global
Function	Change this variable to TRUE to cancel comparing the Controller data to a backup file in the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupCmd[*1]			Member name	.DirName
Meaning	Directory Name			Global/local	Global
Function	Used to specify the directory name in the SD Memory Card for which to back up data. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: STRING(64)			Range of values	Depends on data type.
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupSta[*1]			Member name	.Done
Meaning	Done Flag			Global/local	Global
Function	TRUE when a backup is completed. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sBKUP_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	Read	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupSta[*1]			Member name	.Active
Meaning	Active Flag			Global/local	Global
Function	TRUE when a backup is in progress. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sBKUP_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	Read	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupSta[*1]			Member name	.Err
Meaning	Error Flag			Global/local	Global
Function	TRUE when processing a backup ended in an error. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sBKUP_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	Read	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1VefySta[*1]			Member name	.Done
Meaning	Done Flag			Global/local	Global
Function	TRUE when a verification is completed. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sVEFY_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	Read	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1VefySta[*1]			Member name	.Active
Meaning	Active Flag			Global/local	Global
Function	TRUE when a verification is in progress. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sVEFY_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	Read	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1VefySta[*1]			Member name	.VefyRslt
Meaning	Verify Result Flag			Global/local	Global
Function	TRUE if the data was the same. FALSE if differences were found. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sVEFY_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	Read	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1VefySta[*1]			Member name	.Err
Meaning	Error Flag			Global/local	Global
Function	TRUE when processing a verification ended in an error. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				

Data type	Structure: _sVEFY_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	Read	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

● Functional Classification: Backup

Variable name	_BackupBusy[*1]				
Meaning	Backup Function Busy Flag			Global/local	Global
Function	TRUE when a backup, restoration, or verification is in progress.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	Read	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.03 of the CPU Unit.

● Functional Classification: Power Supply

Variable name	_PowerOnHour				
Meaning	Total Power ON Time			Global/local	Global
Function	Contains the total time that the power has been ON. Contains the total time that the CPU Unit has been ON in 1-hour increments. To reset this value, overwrite the current value with 0. The value is not updated after it reaches 4294967295. This variable is not initialized at startup.				
Data type	UDINT			Range of values	0 to 4294967295
R/W access	RW	Retained	Retained.[*1]	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_PowerOnCount				
Meaning	Power Interruption Count			Global/local	Global
Function	Contains the number of times that the power supply has been interrupted. The value is incremented by 1 each time the power supply is interrupted after the first time that the power was turned ON. To reset this value, overwrite the current value with 0. The value is not updated after it reaches 4294967295. This variable is not initialized at startup.				
Data type	UDINT			Range of values	0 to 4294967295
R/W access	R/W	Retained	Retained.[*1]	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_RetainFail				
Meaning	Retention Failure Flag			Global/local	Global
Function	TRUE at the following times (failure of retention during power interruptions). • When an error is detected in the battery-backup memory check at startup. FALSE at the following times (no failure of retention during power interruptions). • When no error is detected in the battery-backup memory check at startup. • When the user program is downloaded. • When the Clear All Memory operation is performed. **Note** When the encoder home offset data is not retained, the status is given in the error status of the axis variable, and not in this flag.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: Programming

Variable name	P_On				
Meaning	Always TRUE Flag			Global/local	Global
Function	This flag is always TRUE.				
Data type	BOOL			Range of values	TRUE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

Variable name	P_Off				
Meaning	Always FALSE Flag			Global/local	Global
Function	This flag is always FALSE.				
Data type	BOOL			Range of values	FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

Variable name	P_CY				
Meaning	Carry Flag			Global/local	Local
Function	This flag is updated by some instructions.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

Variable name	P_First_RunMode				
Meaning	First RUN Period Flag			Global/local	Local
Function	This flag is TRUE for only one task period after the operating mode of the CPU Unit is changed from PROGRAM mode to RUN mode if execution of the program is in progress. This flag remains FALSE if execution of the program is not in progress. Use this flag to perform initial processing when the CPU Unit begins operation. **Note** You cannot use this system-defined variable inside functions.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

Variable name	P_First_Run*1				
Meaning	First Program Period Flag			Global/local	Local
Function	This flag is TRUE for one task period after execution of the program starts. Use this flag to perform initial processing when execution of a program starts. **Note** You cannot use this system-defined variable inside functions.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.08 of the CPU Unit.

Variable name	P_PRGER				
Meaning	Instruction Error Flag			Global/local	Local
Function	This flag changes to and remains TRUE when an instruction error occurs in the program or in a function/function block called from the program. After this flag changes to TRUE, it stays TRUE until the user program changes it back to FALSE.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: Communications

Variable name	_Port_numUsingPort				
Meaning	Number of Used Ports			Global/local	Global
Function	Gives the number of internal logical ports that are currently used. You can use this variable when you debug the user program.				
Data type	USINT			Range of values	0 to 32
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	Communications instructions (ExecPMCR, SerialSend, SerialRcv, Send, Rcv, and SendCmd)		

Variable name	_Port_isAvailable				
Meaning	Network Communications Instruction Enabled Flag			Global/local	Global
Function	Indicates whether there is an available internal logical port. TRUE when an internal logical port is available. Otherwise FALSE.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	Communications instructions (ExecPMCR, SerialSend, SerialRcv, Send, Rcv, and SendCmd)		

Variable name	_FINSTCPConnSta				
Meaning	FINS/TCP Connection Status			Global/local	Global
Function	Gives the FINS/TCP connection status.				
Data type	WORD			Range of values	16#0000 to 16#FFFF
R/W access	R	Retained	Not retained.	Network Publish	Published.*1
Usage in user program	Possible.	Related instructions	---		

*1 The network for CPU Units with unit version 1.07 or earlier is not published.

Functional Classification: Version

Variable name	_UnitVersion[*1]				
Meaning	Unit Version			Global/local	Global
Function	The unit version of the CPU Unit is stored. The integer part of the unit version is stored in element number 0. The fractional part of the unit version is stored in element number 1. Example 1) If the unit version is 1.08, “1” is stored in element number 0 and “8” is stored in element number 1. Example 2) If the unit version is 1.10, “1” is stored in element number 0 and “10” is stored in element number 1.				
Data type	ARRAY[0..1] OF USINT			Range of values	0 to 99
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.08 of the CPU Unit.

Functional Classification: Self-diagnosis

Variable name	_SelfTest_HighTemperature[*1]				
Meaning	CPU Unit High Temperature Flag			Global/local	Global
Function	TRUE when the internal temperature of the CPU Unit is too high.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.10 of the CPU Unit.

Variable name	_SelfTest_LowBattery[*1]				
Meaning	Low Battery Flag			Global/local	Global
Function	TRUE when the battery is disconnected or the battery voltage is dropped.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.10 of the CPU Unit.

Variable name	_SelfTest_LowFanRevolution[*1]				
Meaning	Low FAN Revolution Flag			Global/local	Global
Function	TRUE when the fan is disconnected or the rotation speed of a fan is decreased. **Note** Always FALSE for an NJ-series CPU Unit.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.10 of the CPU Unit.

A-5-2 PLC Function Module, Category Name: _PLC

● Functional Classification: Debugging

Variable name	_PLC_TraceSta[0..3]			Members	.IsStart
Meaning	Trace Busy Flag			Global/local	Global
Function	TRUE when a trace starts. **Note** You cannot use these system-defined variables in the user program. It is used only to monitor the status of data tracing from the Sysmac Studio.				
Data type	Structure: _sTRACE_STA, Members: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instruc-tions	• TraceTrig • TraceSamp You can access this variable from the user program only with the following instruction. • GetTraceStatus		

Variable name	_PLC_TraceSta[0..3]			Members	.IsComplete
Meaning	Trace Completed Flag			Global/local	Global
Function	TRUE when a trace is completed. **Note** You cannot use this system-defined variable in the user program. It is used only to monitor the status of data tracing from the Sysmac Studio.				
Data type	Structure: _sTRACE_STA, Members: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instruc-tions	• TraceTrig • TraceSamp You can access this variable from the user program only with the following instruction. • GetTraceStatus		

Variable name	_PLC_TraceSta[0..3]			Members	.IsTrigger
Meaning	Trace Trigger Monitor Flag			Global/local	Global
Function	TRUE when the trigger condition is met. FALSE when the next trace starts. **Note** You cannot use these system-defined variables in the user program. It is used only to monitor the status of data tracing from the Sysmac Studio.				
Data type	Structure: _sTRACE_STA, Members: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instruc-tions	• TraceTrig • TraceSamp You can access this variable from the user program only with the following instruction. • GetTraceStatus		

<table>
<tr><td>Variable name</td><td colspan="3">_PLC_TraceSta[0..3]</td><td>Members</td><td>.ParamErr</td></tr>
<tr><td>Meaning</td><td colspan="3">Trace Parameter Error Flag</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">TRUE when a trace starts, but there is an error in the trace settings.
FALSE when the settings are normal.
Note You cannot use these system-defined variables in the user program. It is used only to monitor the status of data tracing from the Sysmac Studio.</td></tr>
<tr><td>Data type</td><td colspan="3">Structure: _sTRACE_STA, Members: BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Not published.</td></tr>
<tr><td>Usage in user program</td><td>Not possible.</td><td>Related instructions</td><td colspan="3">You can access this variable from the user program only with the following instruction.
• GetTraceStatus</td></tr>
</table>

● Functional Classification: Errors

<table>
<tr><td>Variable name</td><td colspan="5">_PLC_ErrSta</td></tr>
<tr><td>Meaning</td><td colspan="3">PLC Function Module Error Status</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">TRUE when there is a Controller error that involves the PLC Function Module.
FALSE when there is no Controller error that involves the PLC Function Module.
Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.</td></tr>
<tr><td>Data type</td><td colspan="3">WORD</td><td>Range of values</td><td>16#0000 to 16#00F0</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">• GetPLCError
You can use the following instruction to clear this variable.
• ResetPLCError</td></tr>
</table>

A-5-3 PLC Function Module, Category Name: _CJB

● Functional Classification: I/O Bus Status

Variable name	_CJB_MaxRackNo				
Meaning	Largest Rack Number			Global/local	Global
Function	Contains the largest rack number of the Expansion Racks that are detected by the Controller. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	UINT			Range of values	0 to 3 "0" means there are no Expansion Racks.
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_CJB_MaxSlotNo				
Meaning	Largest Slot Number			Global/local	Global
Function	Contains one higher than the largest slot number with a CJ-series Unit on each of the Racks that are detected by the Controller. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	ARRAY [0..3] OF UINT			Range of values	0 to 10 0: No CJ-series Unit mounted.
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: I/O Bus Errors

Variable name	_CJB_ErrSta				
Meaning	I/O Bus Error Status			Global/local	Global
Function	Gives the I/O bus error status. **Note** Do not use this variable in the user program. There may be a delay in updating it and concurrency problems may occur. Use this variable only to access status through communications from an external device. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	You can access this variable from the user program only with the following instruction. • GetCJBError You can use the following instruction to clear this variable. • ResetCJBError		

<table>
<tr><td>Variable name</td><td colspan="5">_CJB_MstrErrSta</td></tr>
<tr><td>Meaning</td><td colspan="3">I/O Bus Master Error Status</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Gives the I/O bus master error status.
Note Do not use this variable in the user program. There may be a delay in updating it and concurrency problems may occur. Use these variables only to access status through communications from an external device. Refer to A-4-7 Meanings of Error Status Bits for the meanings of the error status bits.
Note You can use this system-defined variable only for NJ-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">WORD</td><td>Range of values</td><td>16#0000 to 16#00F0</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Not possible.</td><td>Related instructions</td><td colspan="3">You can access this variable from the user program only with the following instruction.
• GetCJBError
You can use the following instruction to clear this variable.
• ResetCJBError</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_CJB_UnitErrSta</td></tr>
<tr><td>Meaning</td><td colspan="3">I/O Bus Unit Error Status</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Gives the error status of the I/O Bus Unit.
Note Do not use this variable in the user program. There may be a delay in updating it and concurrency problems may occur. Use this variable only to access status through communications from an external device. Refer to A-4-7 Meanings of Error Status Bits for the meanings of the error status bits.
Note You can use this system-defined variable only for NJ-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">ARRAY [0..3, 0..9] OF WORD</td><td>Range of values</td><td>16#0000 to 16#80F0</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Not possible.</td><td>Related instructions</td><td colspan="3">You can access this variable from the user program only with the following instruction.
• GetCJBError
You can clear this variable with the following instruction.
• ResetCJBError</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_CJB_InRespTm</td></tr>
<tr><td>Meaning</td><td colspan="3">Basic Input Unit Input Response Times</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Contains the response times of the Basic I/O Units.
Note You can use this system-defined variable only for NJ-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">ARRAY [0..3, 0..9] OF UINT</td><td>Range of values</td><td>0 to 320</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

Functional Classification: Auxiliary Area Bits for CJ-series Units

Variable name	_CJB_IOUnitInfo				
Meaning	Basic I/O Unit Information			Global/local	Global
Function	Shows the status of the Basic I/O Unit alarm output (load short-circuit protection). TRUE: Load short-circuit FALSE: No load short-circuit **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	ARRAY [0..3, 0..9, 0..7] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		
Auxiliary Area addresses: Words	A50 to A69				
Auxiliary Area addresses: Bits	A50.00 to A69.15				

Variable name	_CJB_CBU00InitSta to _CJB_CBU15InitSta				
Meaning	CPU Bus Unit Initializing Flags			Global/local	Global
Function	The corresponding variable is TRUE during initialization of the CPU Bus Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• ResetUnit		
Auxiliary Area addresses: Words	A302				
Auxiliary Area addresses: Bits	A302.00 to A302.15				

Variable name	_CJB_SIO00InitSta to _CJB_SIO95InitSta				
Meaning	Special I/O Unit Initializing Flags			Global/local	Global
Function	The corresponding variable is TRUE during initialization of the Special I/O Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• ResetUnit		
Auxiliary Area addresses: Words	A330 to A335				
Auxiliary Area addresses: Bits	A330.00 to A335.15				

Variable name		_CJB_CBU00Restart to _CJB_CBU15Restart				
Meaning		CPU Bus Unit Restart Bits			Global/local	Global
Function		The CPU Bus Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the CPU Bus Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Flag to TRUE with an instruction, the restart process begins from refresh processing in the next task period. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type		BOOL			Range of values	TRUE or FALSE
R/W access		RW	Retained	Not retained.	Network Publish	Published.
Usage in user program		Possible.	Related instructions	• ResetUnit		
Auxiliary Area addresses	Words	A501				
	Bits	A501.00 to A501.15				

Variable name		_CJB_SIO00Restart to _CJB_SIO95Restart				
Meaning		Special I/O Unit Restart Bits			Global/local	Global
Function		The Special I/O Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the CPU Bus Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Flag to TRUE with an instruction, the restart process begins from refresh processing in the next task period. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type		BOOL			Range of values	TRUE or FALSE
R/W access		RW	Retained	Not retained.	Network Publish	Published.
Usage in user program		Possible.	Related instructions	• ResetUnit		
Auxiliary Area addresses	Words	A502 to A507				
	Bits	A502.00 to A507.15				

<table>
<tr><td>Variable name</td><td colspan="5">_CJB_SCU00P1ChgSta
_CJB_SCU00P2ChgSta
to
_CJB_SCU15P1ChgSta
_CJB_SCU15P2ChgSta</td></tr>
<tr><td>Meaning</td><td colspan="3">Serial Communications Unit 0, Port 1/2 Settings Changing Flags
. . .
Serial Communications Units 1 to 15, Port 1/2 Settings Changing Flags</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">TRUE when the parameters of the specified port are being changed.
FALSE after the parameters are changed.
It is also possible for the user to indicate a change in serial port settings by turning ON the corresponding flag through the execution of an instruction or a user operation.
Note You can use this system-defined variable only for NJ-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>RW</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
<tr><td rowspan="2">Auxiliary Area addresses</td><td>Words</td><td colspan="4">Port on Serial Communications Unit with unit number 0: A620
Ports on Serial Communications Unit with unit numbers 1 to 15: A621 to A635</td></tr>
<tr><td>Bits</td><td colspan="4">Port on Serial Communications Unit with unit number 0: A620.01 to A620.02
Ports on Serial Communications Unit with unit numbers 1 to 15: A621.01 to A635.02</td></tr>
</table>

A-5-4 Motion Control Function Module, Category Name: _MC

● Functional Classification: Motion Control Functions

Variable name	_MC_ErrSta				
Meaning	Motion Control Function Module Error Status			Global/local	Global
Function	Shows the status of errors that are detected in the Motion Control Function Module. You can use this variable directly in the user program. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#40F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• GetMCError • ResetMCError • MC_Reset • MC_GroupReset		

Variable name	_MC_ComErrSta				
Meaning	Common Error Status			Global/local	Global
Function	Shows the status of errors that are detected in common processing for motion control. You can use this variable directly in the user program. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• GetMCError • ResetMCError		

Variable name	_MC_AX_ErrSta				
Meaning	Axis Error Status			Global/local	Global
Function	Shows the error status for each axis. The status of up to 256 axes*1 is shown. You can use this variable directly in the user program. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	ARRAY [0..255] OF WORD*1			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• GetMCError • ResetMCError • MC_Reset		

*1 For NJ-series CPU Units, the error status of up to 64 axes is shown and the data type is ARRAY [0..63] OF WORD.

Variable name	_MC_GRP_ErrSta				
Meaning	Axes Group Error Status			Global/local	Global
Function	Shows the error status for each axes group. The error status for up to 64 axes groups*1 is shown. You can use this variable directly in the user program. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	ARRAY [0..63] OF WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• GetMCError • ResetMCError • MC_GroupReset		

*1 For NJ-series CPU Units, the error status of up to 32 axes groups is shown and the data type is ARRAY [0..31] OF WORD.

Variable name	_MC_COM				
Meaning	Common Variable			Global/local	Global
Function	Shows the status that is common to the Motion Control Function Module. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members.				
Data type	_sCOMMON_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_MC_GRP				
Meaning	Axes Group Variables			Global/local	Global
Function	NX-series CPU Units: Used to specify axes groups and shows multiaxes coordinated control status, and multiaxes coordinated control settings for motion control instructions used for motion control 1. NJ-series CPU Units: Used to specify axes groups and shows multiaxes coordinated control status, and multiaxes coordinated control settings for motion control instructions. When you create an axes group on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members.				
Data type	ARRAY[0..63] OF _sGROUP_REF*1			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 For NJ-series CPU Units, the data type is ARRAY[0..31] OF _sGROUP_REF.

Variable name	_MC1_GRP				
Meaning	Axes Group Variables			Global/local	Global
Function	Used to specify axes groups and shows multiaxes coordinated control status, and multiaxes coordinated control settings for motion control instructions used for motion control 1. When you create an axes group on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members. **Note** You can use this system-defined variable only for NX-series CPU Units. You can access the same values of *_MC1_GRP* and *_MC_GRP* if the array element numbers of them are the same.				
Data type	ARRAY[0..63] OF _sGROUP_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_MC2_GRP				
Meaning	Axes Group Variables			Global/local	Global
Function	Used to specify axes groups and shows multiaxes coordinated control status, and multiaxes coordinated control settings for motion control instructions used for motion control 2. When you create an axes group on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY[0..63] OF _sGROUP_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_MC_AX				
Meaning	Axis Variables			Global/local	Global
Function	NX-series CPU Units: Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 1. NJ-series CPU Units: Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions. When you create an axis on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members.				
Data type	ARRAY[0..255] OF _sAXIS_REF*1			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 For NJ-series CPU Units, the data type is ARRAY[0..63] OF _sAXIS_REF.

Variable name	_MC1_AX				
Meaning	Axis Variables			Global/local	Global
Function	Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 1. When you create an axis on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members. **Note** You can use this system-defined variable only for NX-series CPU Units. You can access the same values of *_MC1_AX* and *_MC_AX* if the array element numbers of them are the same.				
Data type	ARRAY[0..255] OF _sAXIS_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_MC2_AX				
Meaning	Axis Variables			Global/local	Global
Function	Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 2. When you create an axis on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for details on structure members. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY[0..255] OF _sAXIS_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

A-5-5 EtherCAT Master Function Module, Category Name: _EC

● Functional Classification: EtherCAT Communications Errors

Variable name	_EC_ErrSta				
Meaning	Built-in EtherCAT Error			Global/local	Global
Function	This system-defined variable provides the collective status of errors in the EtherCAT Master Function Module. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Get EtherCAT Error Status • GetECError Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_PortErr				
Meaning	Communications Port Error			Global/local	Global
Function	This system-defined variable provides the collective status of errors in the communications ports for the EtherCAT master. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Get EtherCAT Error Status • GetECError Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_MstrErr				
Meaning	Master Error			Global/local	Global
Function	This system-defined variable provides the collective status of EtherCAT master errors and slave errors detected by the EtherCAT master. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Get EtherCAT Error Status • GetECError Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_SlavErr				
Meaning	Slave Error			Global/local	Global
Function	This system-defined variable provides the collective status of all the error status for EtherCAT slaves. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Get EtherCAT Error Status • GetECError Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_SlavErrTbl				
Meaning	Slave Error Table			Global/local	Global
Function	This system-defined variable gives the error status for each EtherCAT slave. The error status is given for each slave in the actual system configuration. This variable array indicates slaves in which there are errors. Status is provided for each EtherCAT slave node address (1 to 512)[*]. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	Array [1..512] OF WORD[*1]			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Get EtherCAT Error Status • GetECError Reset EtherCAT Controller Error • ResetECError		

*1 For NJ-series CPU Units, the node address is 1 to 192 and the data type is Array [1..192] OF WORD.

Variable name	_EC_MacAdrErr				
Meaning	MAC Address Error			Global/local	Global
Function	TRUE if there is an illegal MAC address.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_LanHwErr				
Meaning	Communications Controller Error			Global/local	Global
Function	TRUE if there is a communications controller hardware error.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_LinkOffErr				
Meaning	Link OFF Error			Global/local	Global
Function	TRUE if the communications controller link is not established.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_NetCfgErr				
Meaning	Network Configuration Information Error			Global/local	Global
Function	TRUE if there is illegal network configuration information.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_NetCfgCmpErr				
Meaning	Network Configuration Verification Error			Global/local	Global
Function	TRUE if the network configuration information does not match the actual network configuration.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_NetTopologyErr				
Meaning	Network Configuration Error			Global/local	Global
Function	TRUE if there is a network configuration error (too many devices connected or ring connection).				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_PDCommErr				
Meaning	Process Data Communications Error			Global/local	Global
Function	TRUE if there is an unexpected slave disconnection or connection or if a slave WDT error is detected during process data communications.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_PDTimeoutErr				
Meaning	Process Data Reception Timeout Error			Global/local	Global
Function	TRUE if a timeout occurs while receiving process data.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_PDSendErr				
Meaning	Process Data Transmission Error			Global/local	Global
Function	TRUE if there is a process data transmission error (cannot send within the process data communications period or transmission jitter is over the limit).				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_SlavAdrDupErr				
Meaning	Slave Node Address Duplicated Error			Global/local	Global
Function	TRUE if the same node address is set for more than one slave.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_SlavInitErr				
Meaning	Slave Initialization Error			Global/local	Global
Function	TRUE if there is an error in an initialization command addressed to a slave.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_SlavAppErr				
Meaning	Slave Application Error			Global/local	Global
Function	TRUE if there is an error in the slave's application status register.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_MsgErr				
Meaning	EtherCAT Message Error			Global/local	Global
Function	TRUE when a message is sent to a slave that does not support messages or when there is an error in the format of the response to a message that was sent to a slave.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	CoE messages (Read CoE SDO) • EC_CoESDORead CoE messages (Write CoE SDO) • EC_CoESDOWrite		

Variable name	_EC_SlavEmergErr				
Meaning	Emergency Message Detected			Global/local	Global
Function	TRUE if the master detects an emergency message that was sent by a slave.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

Variable name	_EC_CommErrTbl				
Meaning	Communications Error Slave Table			Global/local	Global
Function	Slaves are given in the table in the order of slave node addresses. The corresponding slave element is TRUE if the master detected an error for the slave.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Controller Error • ResetECError		

*1 For NJ-series CPU Units, the data type is Array [1..192] OF BOOL.

Note The values of all system-defined variables that are related to errors in EtherCAT communications do not change until the cause of the error is removed and then the error in the Controller is reset with the troubleshooting functions of the Sysmac Studio or the ResetECError instruction.

Variable name	_EC_CycleExceeded				
Meaning	EtherCAT Communications Cycle Exceeded			Global/local	Global
Function	TRUE if the CPU Unit cannot establish communications within the set communications period at startup. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: EtherCAT Communications Status

Variable name	_EC_RegSlavTbl				
Meaning	Registered Slave Table			Global/local	Global
Function	This table indicates the slaves that are registered in the network configuration information. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave is registered.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 For NJ-series CPU Units, the data type is Array [1..192] OF BOOL.

Variable name	_EC_EntrySlavTbl				
Meaning	Network Connected Slave Table			Global/local	Global
Function	This table indicates which slaves are connected to the network. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave has entered the network.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 For NJ-series CPU Units, the data type is Array [1..192] OF BOOL.

Variable name	_EC_MBXSlavTbl				
Meaning	Message Communications Enabled Slave Table			Global/local	Global
Function	This table indicates the slaves that can perform message communications. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if message communications are enabled for it (pre-operational, safe-operation, or operational state). **Note** Use this variable to confirm that message communications are possible for the relevant slave before you execute message communications with an EtherCAT slave.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Disconnect EtherCAT Slave • EC_DisconnectSlave Connect EtherCAT Slave • EC_ConnectSlave		

*1 For NJ-series CPU Units, the data type is Array [1..192] OF BOOL.

Variable name	_EC_PDSlavTbl				
Meaning	Process Data Communicating Slave Table			Global/local	Global
Function	This is a table that indicates the slaves that are performing process data communications. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if process data of the corresponding slave is enabled (operational) for both slave inputs and outputs. **Note** Use this variable to confirm that the data for the relevant slave is valid before controlling an EtherCAT slave.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Disconnect EtherCAT Slave • EC_DisconnectSlave Connect EtherCAT Slave • EC_ConnectSlave		

*1 For NJ-series CPU Units, the data type is Array [1..192] OF BOOL.

Variable name	_EC_DisconnSlavTbl				
Meaning	Disconnected Slave Table			Global/local	Global
Function	Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave was disconnected.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Disconnect EtherCAT Slave • EC_DisconnectSlave Connect EtherCAT Slave • EC_ConnectSlave		

*1 For NJ-series CPU Units, the data type is Array [1..192] OF BOOL.

Variable name	_EC_DisableSlavTbl				
Meaning	Disabled Slave Table			Global/local	Global
Function	Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave is disabled.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 For NJ-series CPU Units, the data type is Array [1..192] OF BOOL.

Variable name	_EC_PDActive				
Meaning	Process Data Communications Status			Global/local	Global
Function	TRUE when process data communications are performed with all slaves*.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Disconnect EtherCAT Slave • EC_DisconnectSlave Connect EtherCAT Slave • EC_ConnectSlave		

* Disabled slaves are not included.

Variable name	_EC_PktMonStop				
Meaning	Packet Monitoring Stopped			Global/local	Global
Function	TRUE when packet monitoring is stopped.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Stop Packet Monitor • EC_StopMon Start Packet Monitor • EC_StartMon		

Variable name	_EC_LinkStatus				
Meaning	Link Status			Global/local	Global
Function	TRUE if the communications controller link status is Link ON.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EC_PktSaving				
Meaning	Saving Packet Data File			Global/local	Global
Function	Shows whether a packet data file is being saved. TRUE: Packet data file being saved. FALSE: Packet data file not being saved.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Saving Packet Data File • EC_SaveMon		

Variable name	_EC_InDataInvalid				
Meaning	Input Data Invalid			Global/local	Global
Function	TRUE when process data communications performed in the primary periodic task are not normal and the input data is not valid.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Note All system-defined variables that are related to the status of EtherCAT communications give the current status.

Variable name	_EC_InData1Invalid				
Meaning	Input Data1 Invalid			Global/local	Global
Function	TRUE when process data communications performed in the primary periodic task are not normal and the input data is not valid. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Note All system-defined variables that are related to the status of EtherCAT communications give the current status.

Variable name	_EC_InData2Invalid				
Meaning	Input Data2 Invalid			Global/local	Global
Function	TRUE when process data communications performed in the priority-5 periodic task are not normal and the input data is not valid. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Note All system-defined variables that are related to the status of EtherCAT communications give the current status.

A-5-6 EtherNet/IP Function Module, Category Name: _EIP

● Functional Classification: EtherNet/IP Communications Errors

Variable name	_EIP_ErrSta				
Meaning	Built-in EtherNet/IP Error			Global/local	Global
Function	This is the error status variable for the built-in EtherNet/IP port. NX-series CPU Units: Represents the collective status of the following error flags. • *_EIP1_PortErr* (Communications Port1 Error) • *_EIP2_PortErr* (Communications Port2 Error) • *_EIP1_CipErr* (CIP Communications1 Error) • *_EIP2_CipErr* (CIP Communications2 Error) • *_EIP_TcpAppErr* (TCP Application Communications Error) NJ-series CPU Units: Represents the collective status of the following error flags. • *_EIP_PortErr* (Communications Port Error) • *_EIP_CipErr* (CIP Communications Error) • *_EIP_TcpAppErr* (TCP Application Communications Error) **Note** Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP_PortErr				
Meaning	Communications Port Error			Global/local	Global
Function	This is the error status variable for the communications port. NX-series CPU Units: Represents the collective status of the following error flags. • *_EIP1_MacAdrErr* (Port1 MAC Address Error) • *_EIP1_LanHwErr* (Port1 Communications Controller Error) • *_EIP1_EtnCfgErr* (Port1 Basic Ethernet Setting Error) • *_EIP1_IPAdrCfgErr* (Port1 IP Address Setting Error) • *_EIP1_IPAdrDupErr* (Port1 IP Address Duplication Error) • *_EIP1_BootpErr* (Port1 BOOTP Server Error) • *_EIP_DNSCfgErr* (DNS Setting Error) • *_EIP_DNSSrvErr* (DNS Server Connection Error) • *_EIP_IPRTblErr* (IP Route Table Error) NJ-series CPU Units: Represents the collective status of the following error flags. • *_EIP_MacAdrErr* (MAC Address Error) • *_EIP_LanHwErr* (Communications Controller Error) • *_EIP_EtnCfgErr* (Basic Ethernet Setting Error) • *_EIP_IPAdrCfgErr* (IP Address Setting Error) • *_EIP_IPAdrDupErr* (IP Address Duplication Error) • *_EIP_BootpErr* (BOOTP Server Error) • *_EIP_DNSSrvErr* (DNS Server Connection Error) • *_EIP_IPRTblErr* (IP Route Table Error) **Note** If a Link OFF Detected or Built-in EtherNet/IP Processing Error occurs, it is recorded in the event log and then corresponding bit turns ON. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP1_PortErr				
Meaning	Communications Port1 Error			Global/local	Global
Function	This is the error status variable for the communications port 1. It represents the collective status of the following error flags. • *_EIP1_MacAdrErr* (Port1 MAC Address Error) • *_EIP1_LanHwErr* (Port1 Communications Controller Error) • *_EIP1_EtnCfgErr* (Port1 Basic Ethernet Setting Error) • *_EIP1_IPAdrCfgErr* (Port1 IP Address Setting Error) • *_EIP1_IPAdrDupErr* (Port1 IP Address Duplication Error) • *_EIP1_BootpErr* (Port1 BOOTP Server Error) • *_EIP_DNSCfgErr* (DNS Setting Error) • *_EIP_DNSSrvErr* (DNS Server Connection Error) • *_EIP_IPRTblErr* (IP Route Table Error) **Note** If a Link OFF Detected or Built-in EtherNet/IP Processing Error occurs, it is recorded in the event log and then corresponding bit turns ON. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP2_PortErr				
Meaning	Communications Port2 Error			Global/local	Global
Function	This is the error status variable for the communications port 2. It represents the collective status of the following error flags. • *_EIP2_MacAdrErr* (Port2 MAC Address Error) • *_EIP2_LanHwErr* (Port2 Communications Controller Error) • *_EIP2_EtnCfgErr* (Port2 Basic Ethernet Setting Error) • *_EIP2_IPAdrCfgErr* (Port2 IP Address Setting Error) • *_EIP2_IPAdrDupErr* (Port2 IP Address Duplication Error) • *_EIP2_BootpErr* (Port2 BOOTP Server Error) • *_EIP_DNSCfgErr* (DNS Setting Error) • *_EIP_DNSSrvErr* (DNS Server Connection Error) • *_EIP_IPRTblErr* (IP Route Table Error) **Note** If a Link OFF Detected or Built-in EtherNet/IP Processing Error occurs, it is recorded in the event log and then corresponding bit turns ON. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

<table>
<tr><td>Variable name</td><td colspan="5">_EIP_CipErr</td></tr>
<tr><td>Meaning</td><td colspan="3">CIP Communications Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">This is the error status variable for CIP communications.
NX-series CPU Units: Represents the collective status of the following error flags.
• *_EIP1_IdentityErr* (CIP Communications1 Identity Error)
• *_EIP1_TDLinkCfgErr* (CIP Communications1 Tag Data Link Setting Error)
• *_EIP1_TDLinkOpnErr* (CIP Communications1 Tag Data Link Connection Failed)
• *_EIP1_TDLinkErr* (CIP Communications1 Tag Data Link Communications Error)
• *_EIP1_TagAdrErr* (CIP Communications1 Tag Name Resolution Error)
• *_EIP1_MultiSwONErr* (CIP Communications1 Multiple Switches ON Error)
NJ-series CPU Units: Represents the collective status of the following error flags.
• *_EIP_IdentityErr* (Identity Error)
• *_EIP_TDLinkCfgErr* (Tag Data Link Setting Error)
• *_EIP_TDLinkOpnErr* (Tag Data Link Connection Failed)
• *_EIP_TDLinkErr* (Tag Data Link Communications Error)
• *_EIP_TagAdrErr* (Tag Name Resolution Error)
• *_EIP_MultiSwOnErr* (Multiple Switches ON Error)
Note If a Tag Name Resolution Error occurs, it is recorded in the event log and this variable changes to TRUE. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.</td></tr>
<tr><td>Data type</td><td colspan="3">WORD</td><td>Range of values</td><td>16#0000 to 16#00F0</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">You can access this variable from the user program with the following instruction.
• GetEIPError</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_EIP1_CipErr</td></tr>
<tr><td>Meaning</td><td colspan="3">CIP Communications1 Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">This is the error status variable for CIP communications 1.
It represents the collective status of the following error flags.
• *_EIP1_IdentityErr* (CIP Communications1 Identity Error)
• *_EIP1_TDLinkCfgErr* (CIP Communications1 Tag Data Link Setting Error)
• *_EIP1_TDLinkOpnErr* (CIP Communications1 Tag Data Link Connection Failed)
• *_EIP1_TDLinkErr* (CIP Communications1 Tag Data Link Communications Error)
• *_EIP1_TagAdrErr* (CIP Communications1 Tag Name Resolution Error)
• *_EIP1_MultiSwONErr* (CIP Communications1 Multiple Switches ON Error)
Note If a Tag Name Resolution Error occurs, it is recorded in the event log and this variable changes to TRUE. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.
Note You can use this system-defined variable only for NX-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">WORD</td><td>Range of values</td><td>16#0000 to 16#00F0</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">You can access this variable from the user program with the following instruction.
• GetEIPError</td></tr>
</table>

Variable name	_EIP2_CipErr				
Meaning	CIP Communications2 Error			Global/local	Global
Function	This is the error status variable for CIP communications 2. It represents the collective status of the following error flags. • *_EIP2_IdentityErr* (CIP Communications2 Identity Error) • *_EIP2_TDLinkCfgErr* (CIP Communications2 Tag Data Link Setting Error) • *_EIP2_TDLinkOpnErr* (CIP Communications2 Tag Data Link Connection Failed) • *_EIP2_TDLinkErr* (CIP Communications2 Tag Data Link Communications Error) • *_EIP2_TagAdrErr* (CIP Communications2 Tag Name Resolution Error) • *_EIP2_MultiSwONErr* (CIP Communications2 Multiple Switches ON Error) **Note** If a Tag Name Resolution Error occurs, it is recorded in the event log and this variable changes to TRUE. Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP_TcpAppErr				
Meaning	TCP Application Communications Error			Global/local	Global
Function	This is the error status variable for TCP application communications. It represents the collective status of the following error flags. • *_EIP_TcpAppCfgErr* (TCP/IP Setting Error) • *_EIP_NTPSrvErr* (NTP Server Connection Error) **Note** Refer to *A-4-7 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP_MacAdrErr				
Meaning	MAC Address Error			Global/local	Global
Function	NX-series CPU Units: Indicates that an error occurred when the MAC address was read on the communications port 1 at startup. TRUE: Error FALSE: Normal NJ-series CPU Units: Indicates that an error occurred when the MAC address was read at startup. TRUE: Error FALSE: Normal				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

<table>
<tr><td>Variable name</td><td colspan="5">_EIP1_MacAdrErr</td></tr>
<tr><td>Meaning</td><td colspan="3">Port1 MAC Address Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Indicates that an error occurred when the MAC address was read on the communications port 1 at startup.
TRUE: Error
FALSE: Normal
Note You can use this system-defined variable only for NX-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_EIP2_MacAdrErr</td></tr>
<tr><td>Meaning</td><td colspan="3">Port2 MAC Address Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Indicates that an error occurred when the MAC address was read on the communications port 2 at startup.
TRUE: Error
FALSE: Normal
Note You can use this system-defined variable only for NX-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_EIP_LanHwErr</td></tr>
<tr><td>Meaning</td><td colspan="3">Communications Controller Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">NX-series CPU Units: Indicates that a communications controller failure occurred on the communications port 1.
TRUE: Failure
FALSE: Normal

NJ-series CPU Units: Indicates that a communications controller failure occurred.
TRUE: Failure
FALSE: Normal</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_EIP1_LanHwErr</td></tr>
<tr><td>Meaning</td><td colspan="3">Port1 Communications Controller Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Indicates that a communications controller failure occurred on the communications port 1.
TRUE: Failure
FALSE: Normal
Note You can use this system-defined variable only for NX-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

Variable name	_EIP2_LanHwErr				
Meaning	Port2 Communications Controller Error			Global/local	Global
Function	Indicates that a communications controller failure occurred on the communications port 2. TRUE: Failure FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_EtnCfgErr				
Meaning	Basic Ethernet Setting Error			Global/local	Global
Function	NX-series CPU Units: Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 1 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal NJ-series CPU Units: Indicates that the Ethernet communications speed setting (Speed/Duplex) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_EtnCfgErr				
Meaning	Port1 Basic Ethernet Setting Error			Global/local	Global
Function	Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 1 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_EtnCfgErr				
Meaning	Port2 Basic Ethernet Setting Error			Global/local	Global
Function	Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 2 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_IPAdrCfgErr				
Meaning	IP Address Setting Error			Global/local	Global
Function	NX-series CPU Units: Indicates the IP address setting errors for the communications port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The DNS settings are not correct. FALSE: Normal NJ-series CPU Units: Indicates the IP address setting errors. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The DNS settings are not correct. FALSE: Normal				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_IPAdrCfgErr				
Meaning	Port1 IP Address Setting Error			Global/local	Global
Function	Indicates the IP address setting errors for the communications port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The DNS settings are not correct. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_IPAdrCfgErr				
Meaning	Port2 IP Address Setting Error			Global/local	Global
Function	Indicates the IP address setting errors for the communications port 2. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The DNS settings are not correct. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_IPAdrDupErr				
Meaning	IP Address Duplication Error			Global/local	Global
Function	NX-series CPU Units: Indicates that the same IP address is assigned to more than one node for the communications port 1. TRUE: Duplication occurred. FALSE: Other than the above. NJ-series CPU Units: Indicates that the same IP address is assigned to more than one node. TRUE: Duplication occurred. FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_IPAdrDupErr				
Meaning	Port1 IP Address Duplication Error			Global/local	Global
Function	Indicates that the same IP address is assigned to more than one node for the communications port 1. TRUE: Duplication occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_IPAdrDupErr				
Meaning	Port2 IP Address Duplication Error			Global/local	Global
Function	Indicates that the same IP address is assigned to more than one node for the communications port 2. TRUE: Duplication occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_DNSCfgErr				
Meaning	DNS Setting Error			Global/local	Global
Function	Indicates that the DNS or hosts settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_BootpErr				
Meaning	BOOTP Server Error			Global/local	Global
Function	NX-series CPU Units: Indicates that a BOOTP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. NJ-series CPU Units: Indicates that a BOOTP server connection failure occurred. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_BootpErr				
Meaning	Port1 BOOTP Server Error			Global/local	Global
Function	Indicates that a BOOTP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_BootpErr				
Meaning	Port2 BOOTP Server Error			Global/local	Global
Function	Indicates that a BOOTP server connection failure occurred on the communications port 2. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

<table>
<tr><td>Variable name</td><td colspan="5">_EIP_IPRTblErr</td></tr>
<tr><td>Meaning</td><td colspan="3">IP Route Table Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">NX-series CPU Units: Indicates that the default gateway settings or IP router table settings are incorrect. Or, a read operation failed.
TRUE: Setting incorrect or read failed
FALSE: Normal
Note For NX-series CPU Units, the host setting errors will be indicated to _EIP_DNSCfgErr.

NJ-series CPU Units: Indicates that the IP router table or hosts settings are incorrect. Or, a read operation failed.
TRUE: Setting incorrect or read failed
FALSE: Normal
Note For NJ-series CPU Units, the default gateway setting errors will be indicated to _EIP_IPAdrCfgErr.</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_EIP_IdentityErr</td></tr>
<tr><td>Meaning</td><td colspan="3">Identity Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">NX-series CPU Units: Indicates that the identity information for CIP communications 1 (which you cannot overwrite) is incorrect. Or, a read operation failed.
TRUE: Setting incorrect or read failed
FALSE: Normal

NJ-series CPU Units: Indicates that the identity information (which you cannot overwrite) is incorrect. Or, a read operation failed.
TRUE: Setting incorrect or read failed
FALSE: Normal</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_EIP1_IdentityErr</td></tr>
<tr><td>Meaning</td><td colspan="3">CIP Communications1 Identity Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Indicates that the identity information for CIP communications 1 (which you cannot overwrite) is incorrect. Or, a read operation failed.
TRUE: Setting incorrect or read failed
FALSE: Normal
Note You can use this system-defined variable only for NX-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

Variable name	_EIP2_IdentityErr				
Meaning	CIP Communications2 Identity Error			Global/local	Global
Function	Indicates that the identity information for CIP communications 2 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkCfgErr				
Meaning	Tag Data Link Setting Error			Global/local	Global
Function	NX-series CPU Units: Indicates that the tag data link settings for CIP communications 1 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal NJ-series CPU Units: Indicates that the tag data link settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkCfgErr				
Meaning	CIP Communications1 Tag Data Link Setting Error			Global/local	Global
Function	Indicates that the tag data link settings for CIP communications 1 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkCfgErr				
Meaning	CIP Communications2 Tag Data Link Setting Error			Global/local	Global
Function	Indicates that the tag data link settings for CIP communications 2 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkOpnErr				
Meaning	Tag Data Link Connection Failed			Global/local	Global
Function	NX-series CPU Units: Indicates that establishing a tag data link connection for CIP communications 1 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. •The information registered for a target node in the tag data link parameters is different from the actual node information. •There was no response from the remote node. FALSE: Other than the above. NJ-series CPU Units: Indicates that establishing a tag data link connection failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. •The information registered for a target node in the tag data link parameters is different from the actual node information. •There was no response from the remote node. FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkOpnErr				
Meaning	CIP Communications1 Tag Data Link Connection Failed			Global/local	Global
Function	Indicates that establishing a tag data link connection for CIP communications 1 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. •The information registered for a target node in the tag data link parameters is different from the actual node information. •There was no response from the remote node. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkOpnErr				
Meaning	CIP Communications2 Tag Data Link Connection Failed			Global/local	Global
Function	Indicates that establishing a tag data link connection for CIP communications 2 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. •The information registered for a target node in the tag data link parameters is different from the actual node information. •There was no response from the remote node. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkErr				
Meaning	Tag Data Link Communications Error			Global/local	Global
Function	NX-series CPU Units: Indicates that a timeout occurred in a tag data link connection for CIP communications 1. TRUE: A timeout occurred. FALSE: Other than the above. NJ-series CPU Units: Indicates that a timeout occurred in a tag data link connection. TRUE: A timeout occurred. FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkErr				
Meaning	CIP Communications1 Tag Data Link Communications Error			Global/local	Global
Function	Indicates that a timeout occurred in a tag data link connection for CIP communications 1. TRUE: A timeout occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkErr				
Meaning	CIP Communications2 Tag Data Link Communications Error			Global/local	Global
Function	Indicates that a timeout occurred in a tag data link connection for CIP communications 2. TRUE: A timeout occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

<table>
<tr><td>Variable name</td><td colspan="5">_EIP_TagAdrErr</td></tr>
<tr><td>Meaning</td><td colspan="3">Tag Name Resolution Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">NX-series CPU Units: Indicates that tag resolution for CIP communications 1 failed (i.e., the address could not be identified from the tag name).
TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible.
•The size of the network variable is different from the tag settings.
•The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit.
•There is no network variable in the CPU Unit that corresponds to the tag setting.
FALSE: Other than the above.

NJ-series CPU Units: Indicates that tag resolution failed (i.e., the address could not be identified from the tag name).
TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible.
•The size of the network variable is different from the tag settings.
•The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit.
•There is no network variable in the CPU Unit that corresponds to the tag setting.
FALSE: Other than the above.</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_EIP1_TagAdrErr</td></tr>
<tr><td>Meaning</td><td colspan="3">CIP Communications1 Tag Name Resolution Error</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Indicates that tag resolution for CIP communications 1 failed (i.e., the address could not be identified from the tag name).
TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible.
•The size of the network variable is different from the tag settings.
•The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit.
•There is no network variable in the CPU Unit that corresponds to the tag setting.
FALSE: Other than the above.
Note You can use this system-defined variable only for NX-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

Variable name	_EIP2_TagAdrErr				
Meaning	CIP Communications2 Tag Name Resolution Error			Global/local	Global
Function	Indicates that tag resolution for CIP communications 2 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. •The size of the network variable is different from the tag settings. •The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. •There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_MultiSwONErr				
Meaning	Multiple Switches ON Error			Global/local	Global
Function	NX-series CPU Units: Indicates that more than one switch turned ON at the same time in CIP communications 1. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. NJ-series CPU Units: Indicates that more than one switch turned ON at the same time. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_MultiSwONErr				
Meaning	CIP Communications1 Multiple Switches ON Error			Global/local	Global
Function	Indicates that more than one switch turned ON at the same time in CIP communications 1. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_MultiSwONErr				
Meaning	CIP Communications2 Multiple Switches ON Error			Global/local	Global
Function	Indicates that more than one switch turned ON at the same time in CIP communications 2. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TcpAppCfgErr				
Meaning	TCP/IP Setting Error			Global/local	Global
Function	TRUE: At least one of the set values for a TCP/IP application (FTP, NTP, SNMP) is incorrect. Or, a read operation failed. FALSE: Normal.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_NTPSrvErr				
Meaning	NTP Server Connection Error			Global/local	Global
Function	TRUE: The NTP client failed to connect to the server (timeout). FALSE: NTP is not set or the connection was successful.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_DNSSrvErr				
Meaning	DNS Server Connection Error			Global/local	Global
Function	TRUE: The DNS client failed to connect to the server (timeout). FALSE: DNS is not enabled. Or, DNS is enabled and the connection was successful.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: EtherNet/IP Communications Status

Variable name	_EIP_EtnOnlineSta				
Meaning	Online			Global/local	Global
Function	NX-series CPU Units: Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 1 (that is, the link is ON, IP address is defined, and there are no errors). TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. NJ-series CPU Units: Indicates that the built-in EtherNet/IP port's communications can be used via the communications port (that is, the link is ON, IP address is defined, and there are no errors). TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_EtnOnlineSta				
Meaning	Port1 Online			Global/local	Global
Function	Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 1 (that is, the link is ON, IP address is defined, and there are no errors). TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_EtnOnlineSta				
Meaning	Port2 Online			Global/local	Global
Function	Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 2 (that is, the link is ON, IP address is defined, and there are no errors). TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkRunSta				
Meaning	Tag Data Link Communications Status			Global/local	Global
Function	NX-series CPU Units: Indicates that at least one connection is in normal operation in CIP communications 1. TRUE: Normal operation FALSE: Other than the above. NJ-series CPU Units: Indicates that at least one connection is in normal operation. TRUE: Normal operation FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkRunSta				
Meaning	CIP Communications1 Tag Data Link Communications Status			Global/local	Global
Function	Indicates that at least one connection is in normal operation in CIP communications 1. TRUE: Normal operation FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkRunSta				
Meaning	CIP Communications2 Tag Data Link Communications Status			Global/local	Global
Function	Indicates that at least one connection is in normal operation in CIP communications 2. TRUE: Normal operation FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkAllRunSta				
Meaning	All Tag Data Link Communications Status			Global/local	Global
Function	NX-series CPU Units: Indicates that all tag data links are communicating in CIP communications 1. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. NJ-series CPU Units: Indicates that all tag data links are communicating. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkAllRunSta				
Meaning	CIP Communications1 All Tag Data Link Communications Status			Global/local	Global
Function	Indicates that all tag data links are communicating in CIP communications 1. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkAllRunSta				
Meaning	CIP Communications2 All Tag Data Link Communications Status			Global/local	Global
Function	Indicates that all tag data links are communicating in CIP communications 2. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_RegTargetSta [255]				
Meaning	Registered Target Node Information			Global/local	Global
Function	NX-series CPU Units: Gives a list of nodes for which built-in EtherNet/IP connections are registered for CIP communications 1. This variable is valid only when the built-in EtherNet/IP port is the originator. *Array[x]* is TRUE: The connection to the node with a target node ID of x is registered. *Array[x]* is FALSE: The connection to the node with a target node ID of x is not registered. NJ-series CPU Units: Gives a list of nodes for which built-in EtherNet/IP connections are registered. This variable is valid only when the built-in EtherNet/IP port is the originator. *Array[x]* is TRUE: The connection to the node with a target node ID of x is registered. *Array[x]* is FALSE: The connection to the node with a target node ID of x is not registered.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_RegTargetSta [255]				
Meaning	CIP Communications1 Registered Target Node Information			Global/local	Global
Function	Gives a list of nodes for which built-in EtherNet/IP connections are registered for CIP communications 1. This variable is valid only when the built-in EtherNet/IP port is the originator. *Array[x]* is TRUE: The connection to the node with a target node ID of x is registered. *Array[x]* is FALSE: The connection to the node with a target node ID of x is not registered. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_RegTargetSta [255]				
Meaning	CIP Communications2 Registered Target Node Information			Global/local	Global
Function	Gives a list of nodes for which built-in EtherNet/IP connections are registered for CIP communications 2. This variable is valid only when the built-in EtherNet/IP port is the originator. *Array[x]* is TRUE: The connection to the node with a target node ID of x is registered. *Array[x]* is FALSE: The connection to the node with a target node ID of x is not registered. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_EstbTargetSta [255]				
Meaning	Normal Target Node Information			Global/local	Global
Function	NX-series CPU Units: Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 1. *Array[x]* is TRUE: The connection to the node with a target node ID of x was established normally. *Array[x]* is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. NJ-series CPU Units: Gives a list of nodes that have normally established EtherNet/IP connections. *Array[x]* is TRUE: The connection to the node with a target node ID of x was established normally. *Array[x]* is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_EstbTargetSta [255]				
Meaning	CIP Communications1 Normal Target Node Information			Global/local	Global
Function	Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 1. *Array[x]* is TRUE: The connection to the node with a target node ID of x was established normally. *Array[x]* is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_EstbTargetSta [255]				
Meaning	CIP Communications2 Normal Target Node Information			Global/local	Global
Function	Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 2. *Array[x]* is TRUE: The connection to the node with a target node ID of x was established normally. *Array[x]* is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TargetPLCModeSta [255]				
Meaning	Target PLC Operating Mode			Global/local	Global
Function	NX-series CPU Units: Shows the operating status of the target node Controllers that are connected for CIP communications 1, with the built-in EtherNet/IP port as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. If the corresponding Normal Target Node Information is FALSE, the Target Node Controller Operating Information indicates the previous operating status. *Array[x]* is TRUE: This is the operating state of the target Controller with a node address of x. *Array[x]* is FALSE: Other than the above. NJ-series CPU Units: Shows the operating status of the target node Controllers that are connected with the built-in EtherNet/IP port as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. If the corresponding Normal Target Node Information is FALSE, the Target Node Controller Operating Information indicates the previous operating status. *Array[x]* is TRUE: This is the operating state of the target Controller with a node address of x. *Array[x]* is FALSE: Other than the above.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TargetPLCModeSta [255]				
Meaning	CIP Communications1 Target PLC Operating Mode			Global/local	Global
Function	Shows the operating status of the target node Controllers that are connected for CIP communications 1, with the built-in EtherNet/IP port as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. If the corresponding Normal Target Node Information is FALSE, the Target Node Controller Operating Information indicates the previous operating status. *Array[x]* is TRUE: This is the operating state of the target Controller with a node address of x. *Array[x]* is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TargetPLCModeSta [255]				
Meaning	CIP Communications2 Target PLC Operating Mode			Global/local	Global
Function	Shows the operating status of the target node Controllers that are connected for CIP communications 2, with the built-in EtherNet/IP port as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. If the corresponding Normal Target Node Information is FALSE, the Target Node Controller Operating Information indicates the previous operating status. *Array[x]* is TRUE: This is the operating state of the target Controller with a node address of x. *Array[x]* is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TargetPLCErr [255]				
Meaning	Target PLC Error Information			Global/local	Global
Function	NX-series CPU Units: Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 1, with the built-in EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. *Array[x]* is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. *Array[x]* is FALSE: Other than the above. NJ-series CPU Units: Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected with the built-in EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. *Array[x]* is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. *Array[x]* is FALSE: Other than the above.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TargetPLCErr [255]				
Meaning	CIP Communications1 Target PLC Error Information			Global/local	Global
Function	Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 1, with the built-in EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. *Array[x]* is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. *Array[x]* is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TargetPLCErr [255]				
Meaning	CIP Communications2 Target PLC Error Information			Global/local	Global
Function	Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 2, with the built-in EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. *Array[x]* is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. *Array[x]* is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

<table>
<tr><td>Variable name</td><td colspan="5">_EIP_TargetNodeErr [255]</td></tr>
<tr><td>Meaning</td><td colspan="3">Target Node Error Information</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">NX-series CPU Units: Indicates that the connection for the Registered Target Node Information for CIP communications 1 was not established or that an error occurred in the target Controller.
The array elements are valid only when the Registered Target Node Information is TRUE.
Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller.
Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE).

NJ-series CPU Units: Indicates that the connection for the Registered Target Node Information was not established or that an error occurred in the target Controller.
The array elements are valid only when the Registered Target Node Information is TRUE.
Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller.
Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE).</td></tr>
<tr><td>Data type</td><td colspan="3">ARRAY [0..255] OF BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_EIP1_TargetNodeErr [255]</td></tr>
<tr><td>Meaning</td><td colspan="3">CIP Communications1 Target Node Error Information</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Indicates that the connection for the Registered Target Node Information for CIP communications 1 was not established or that an error occurred in the target Controller.
The array elements are valid only when the Registered Target Node Information is TRUE.
Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller.
Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE).

Note You can use this system-defined variable only for NX-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">ARRAY [0..255] OF BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="5">_EIP2_TargetNodeErr [255]</td></tr>
<tr><td>Meaning</td><td colspan="3">CIP Communications2 Target Node Error Information</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Indicates that the connection for the Registered Target Node Information for CIP communications 2 was not established or that an error occurred in the target Controller.
The array elements are valid only when the Registered Target Node Information is TRUE.
Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller.
Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE).
Note You can use this system-defined variable only for NX-series CPU Units.</td></tr>
<tr><td>Data type</td><td colspan="3">ARRAY [0..255] OF BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Possible.</td><td>Related instructions</td><td colspan="3">---</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="3">_EIP_NTPResult</td><td>Member name</td><td>.ExecTime</td></tr>
<tr><td>Meaning</td><td colspan="3">NTP Last Operation Time</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">Gives the last time that NTP processing ended normally.
The time that was obtained from the NTP server is stored when the time is obtained normally.
The time is not stored if it is not obtained from the NTP server normally.
Note Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device.</td></tr>
<tr><td>Data type</td><td colspan="3">Structure: _sNTP_RESULT
Members: DATE_AND_TIME</td><td>Range of values</td><td>Depends on data type.</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Not possible.</td><td>Related instructions</td><td colspan="3">You can read the contents of this variable with the GetNTPStatus instruction.</td></tr>
</table>

<table>
<tr><td>Variable name</td><td colspan="3">_EIP_NTPResult</td><td>Member name</td><td>.ExecNormal</td></tr>
<tr><td>Meaning</td><td colspan="3">NTP Operation Result</td><td>Global/local</td><td>Global</td></tr>
<tr><td>Function</td><td colspan="5">This variable shows if the NTP operation ended normally.
TRUE: Indicates an NTP normal end.
FALSE: Indicates that NTP operation ended in an error or has not been executed even once.
Note Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device.</td></tr>
<tr><td>Data type</td><td colspan="3">BOOL</td><td>Range of values</td><td>TRUE or FALSE</td></tr>
<tr><td>R/W access</td><td>R</td><td>Retained</td><td>Not retained.</td><td>Network Publish</td><td>Published.</td></tr>
<tr><td>Usage in user program</td><td>Not possible.</td><td>Related instructions</td><td colspan="3">You can read the contents of this variable with the GetNTPStatus instruction.</td></tr>
</table>

● Functional Classification: EtherNet/IP Communications Switches

Variable name	_EIP_TDLinkStartCmd				
Meaning	Tag Data Link Communications Start Switch			Global/local	Global
Function	NX-series CPU Units: Change this variable to TRUE to start tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation starts. NJ-series CPU Units: Change this variable to TRUE to start tag data links. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkStartCmd				
Meaning	CIP Communications1 Tag Data Link Communications Start Switch			Global/local	Global
Function	Change this variable to TRUE to start tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkStartCmd				
Meaning	CIP Communications2 Tag Data Link Communications Start Switch			Global/local	Global
Function	Change this variable to TRUE to start tag data links for CIP communications 2. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkStopCmd				
Meaning	Tag Data Link Communications Stop Switch			Global/local	Global
Function	NX-series CPU Units: Change this variable to TRUE to stop tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation stops. NJ-series CPU Units: Change this variable to TRUE to stop tag data links. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkStopCmd				
Meaning	CIP Communications1 Tag Data Link Communications Stop Switch			Global/local	Global
Function	Change this variable to TRUE to stop tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkStopCmd				
Meaning	CIP Communications2 Tag Data Link Communications Stop Switch			Global/local	Global
Function	Change this variable to TRUE to stop tag data links for CIP communications 2. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

A-6 Attributes of CPU Unit Data

The following table shows the attributes of the CPU Unit data including the Retain/Non-retain attribute in the following cases: power interruption, power on, operating mode change, and major fault level Controller error.

CPU Unit data			Data retention at power interruptions	When power is turned ON	Status changes		Writing when write protection is enabled	Transferring data with the Sysmac Studio	Operating modes permitting writing	Overwriting in RUN mode
					Change between PROGRAM mode and RUN mode	When a Major Fault Level Controller Error occurs		Synchronized data		
User program	**POUs and user program execution ID in user program**		Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM/RUN mode (online editing)	Supported. Online editing
Task Setup	**Task Settings**		Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM mode	Not supported.
Variables	**Variable tables (but not variable values)**	**Device variable**	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM mode	Not supported.
		User-defined variables	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM/RUN mode (online editing)	Supported. Online editing
Data type		**User-defined data types**	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM/RUN mode (online editing)	Supported. Online editing
Controller name		**CPU Unit name**	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM/RUN mode	Supported.
		Built-in EtherNet/IP port name	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Supported	Not retained.	PROGRAM/RUN mode	Supported.

CPU Unit data			Data retention at power interruptions	When power is turned ON	Status changes		Writing when write protection is enabled	Transferring data with the Sysmac Studio	Operating modes permitting writing	Overwriting in RUN mode
					Change between PROGRAM mode and RUN mode	When a Major Fault Level Controller Error occurs		Synchronized data		
Controller Setup	Operation Settings	Operation Settings Error settings	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	CPU Unit name: RUN/PROGRAM mode, Other settings: PROGRAM mode	Not supported.
Controller Setup	Security Settings	Protection Settings at Startup	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	Write Protection and other settings: PROGRAM mode	Supported.
Controller Setup	Built-in EtherNet/IP Port Settings	TCP/IP Settings, Built-in EtherNet/IP Port Link Settings, Service Settings. SNMP Settings, SNMP Trap Settings, NTP Settings, FTP Settings, and IP Router Tables	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM mode	Not supported.
Controller Setup	Built-in EtherNet/IP Port Settings	Tag data link settings for built-in EtherNet/IP port	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Supported	Not retained.	PROGRAM/RUN	Not supported.
Controller Setup	FINS Settings	Node Address Settings, FINS/UDP Settings, FINS/TCP Settings, FINS Routing Tables	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM mode	Not supported.
Motion Control Setup	Axis assignments, axis parameter settings, axes group parameter settings, MC common parameter settings		Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM mode	Not supported.
Cam Data			Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM mode	Not supported.
Event Setting Table	Event Setting Table	User-defined error messages	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	RUN/PROGRAM mode	Not supported.
Bus configuration	CJ-series bus configuration	I/O table	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM mode	Not supported.

CPU Unit data			Data retention at power interruptions	When power is turned ON	Status changes		Writing when write protection is enabled	Transferring data with the Sysmac Studio	Operating modes permitting writing	Overwriting in RUN mode
					Change between PROGRAM mode and RUN mode	When a Major Fault Level Controller Error occurs		Synchronized data		
Special I/O Unit Settings/CPU Bus Unit Settings	**CJ-series Unit Settings**	**Data in CJ-series Units, such as protocol macros**	Retained (in CJ-series Units).	---	Retained.	Retained.	Supported.	Not retained.	Depends on the Unit.	
		Words allocated to CPU Bus Units, Example: Controller Link Data Link Tables.	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Supported.	Not retained.	RUN/PROGRAM mode	Supported.
		Words allocated in DM Area	Retained (with Battery).	Same as before power interruption.	Retained.	Retained.	Supported	Retained.	RUN/PROGRAM mode	Supported.
EtherCAT Configuration	**EtherCAT Network Configuration**	**Network configuration information.**	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM mode	Not supported.
EtherCAT Settings	**EtherCAT Settings**	**Master**	Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Retained.	PROGRAM mode	Not supported.
		Settings in Slaves	Retained (by slaves).	---	Retained.	Retained.	Supported.	Retained.	RUN/PROGRAM mode	Supported.
Operation Authority Verification			Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Not retained.	PROGRAM mode	Not supported.
User program execution ID in CPU Unit			Retained (with non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Not supported.	Not retained.	PROGRAM mode	Not supported.
Present values of variables	**Values of non-retained variables**	**User-defined variables and device variables**	Not retained.	Initial values	Initial values	Initial values	Supported.	Not retained.	RUN/PROGRAM mode	Supported.
	Values of retained variables	**User-defined variables and device variables**	Retained (with Battery).	Same as before power interruption.	Retained.	Retained.	Supported.	Not retained.	RUN/PROGRAM mode	Supported.
Contents of memory used for CJ-series Units	**CIO/WR**		Not retained.	16#0000	16#0000	16#0000	Supported.	Not retained.	RUN/PROGRAM mode	Supported.
	HR/DM/EM		Retained (with Battery).	Same as before power interruption.	Retained.	Retained.	Supported.	Not retained.	RUN/PROGRAM mode	Supported.

CPU Unit data			Data retention at power interruptions	When power is turned ON	Status changes: Change between PROGRAM mode and RUN mode	Status changes: When a Major Fault Level Controller Error occurs	Writing when write protection is enabled	Transferring data with the Sysmac Studio: Synchronized data	Operating modes permitting writing	Overwriting in RUN mode
Event logs	**Logs**	**System log** **User event log**	Retained (with Battery).	Same as before power interruption.	Retained.	Retained.	Supported.	Not retained.		Supported.
Internal clock	**Depends on the specifications of each system-defined variable.**		Retained (with Battery).	With Battery: Retained (continued), Without Battery: Not predictable (may stop).	Retained (continued).	Retained (continued).	Supported.	Not retained.	RUN/PROGRAM mode	Supported.
Absolute encoder home offset			Retained (with Battery).	Same as before power interruption.	Retained (continued).	Retained (continued).	Supported.	Not retained.		Not supported.

A-7 Contents of Memory Used for CJ-series Units

You can specify addresses in the memory used for CJ-series Units for AT specifications for variables. Details on each area are provided below.

Note You cannot use memory for CJ-series Units with NX-series CPU Units.

A-7-1 CIO Area

I/O Bits

Description

The bits in this area are allocated to input and output terminals on CJ-series Basic I/O Units. The number of words (16 bits each) that is required for each CJ-series Basic I/O Unit are allocated in order based on the position where the Units are connected (from left to right starting from the Unit that is closest to the CPU Unit). Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

Addresses

Addresses	Word addresses	Bit addresses
Range	CIO 0 to CIO 159	0.00 to 159.15

Additional Information

You can access this area on NJ-series CPU Units through device variables allocated to I/O ports. We therefore recommend that you do not use AT specifications to access this area. You should use AT specifications for the CIO Area only when you specify addresses for some of the Special Units.

CPU Bus Unit Area

Description

The bits in this area are allocated to control and status information for CJ-series CPU Bus Units. Each Unit is allocated 25 words based on its unit number. Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

Addresses

Addresses	Word addresses	Bit addresses	Words per Unit
Range	CIO 1500 to CIO 1899	CIO 1500.00 to CIO 1899.15	25 words

The words that are allocated are listed in the following table.

Word addresses	Unit Number
CIO 1500 to CIO 1524	0

Word addresses	Unit Number
CIO 1525 to CIO 1549	1
to	to
CIO 1875 to CIO 1899	F

For details on how to use the allocated words, refer to the operation manual for the CJ-series CPU Bus Unit.

Precautions for Correct Use

You can access the CPU Bus Unit Area in NJ-series CPU Units through the device variables that are allocated to I/O ports. We therefore recommend that you do not use AT specifications to access this area. You should use AT specifications for the CIO Area only when you specify addresses for some of the Special Units.

Special I/O Unit Area

● Description

The bits in this area are allocated to control and status information for CJ-series Special I/O Units. Each Unit is allocated 10 words based on the unit number for up to a total of 96 Units (unit numbers 0 to 95). Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses	Words per Unit
Range	CIO 2000 to CIO 2959 (10 words × 96 unit numbers)	CIO 2000.00 to CIO 2959.15	10 words

The words that are allocated are listed in the following table.

Word addresses	Unit Number
CIO 2000 to CIO 2009	0
CIO 2010 to CIO 2019	1
to	to
CIO 2950 to CIO 2959	95

For details on how to use the allocated words, refer to the operation manual for the CJ-series Special I/O Unit.

Additional Information

You can access the Special I/O Unit Area in NJ-series CPU Units through the device variables that are allocated to I/O ports. We therefore recommend that you do not use AT specifications to access this area.

DeviceNet Area

● Description

The bits in this area are allocated to the slaves when the remote I/O master function of a DeviceNet Unit is used (fixed allocations only). Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses
Range	CIO 3200 to CIO 3799	CIO 3200.00 to CIO 3799.15

Words in this area are allocated to slaves for fixed allocations according to fixed allocation setting 1, 2, or 3 in the software switches in the CIO Area. Select one of these fixed areas.

Addresses	Master to slave output area	Slave to master input area
Fixed allocation area 1	CIO 3200 to CIO 3263	CIO 3300 to CIO 3363
Fixed allocation area 2	CIO 3400 to CIO 3463	CIO 3500 to CIO 3563
Fixed allocation area 3	CIO 3600 to CIO 3663	CIO 3700 to CIO 3763

You can allocate memory in the DeviceNet Area even if you use fixed allocations to use the remote I/O slave function of a DeviceNet Unit.

Addresses	Master to slave output area	Slave to master input area
Fixed allocation area 1	CIO 3370	CIO 3270
Fixed allocation area 2	CIO 3570	CIO 3470
Fixed allocation area 3	CIO 3770	CIO 3670

Refer to the *CJ-series DeviceNet Units Operation Manual for NJ-series CPU Unit* (Cat. No. W497) for details.

CIO Area Work Areas

● Description

You use the bits in these areas only in programming. You cannot use them to input or output data through external I/O terminals. If you need work bits, you should normally use bits in this area. Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses
Range	CIO 1300 to CIO 1499 and CIO 3800 to CIO 6143	CIO 1300.00 to CIO 1499.15 and CIO 3800.00 to CIO 6143.15

A-7-2 Internal I/O Area

● Description

You use the bits in these areas only in programming. You cannot use them to input or output data through external I/O terminals. If you need work bits, you should normally use bits in this area. Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

- **Addresses**

Addresses	Word addresses	Bit addresses
Range	W000 to W511	W000.00 to W511.15

A-7-3 Holding Area

- **Description**

You use the words and bits in this area only in programming. The status of the words and bits in this area are retained during power interruptions or when the operating mode is changed between PROGRAM and RUN mode.

- **Addresses**

Addresses	Word addresses	Bit addresses
Range	H0 to H511	H0.00 to H511.15

A-7-4 DM Area

- **Description**

This is a general-purpose data area used to read and write 16-bit words. You can also add a bit number to address specify bits. Data in this area is retained during power interruption or when the operating mode is changed between PROGRAM and RUN mode.

- **Addresses**

Addresses	Word addresses	Bit addresses
Range	D0 to D32767	D0.00 to D32767.15

DM Area Words for Special Units

- **Description**

The following words in the DM Area are allocated to initial settings for Special Units.

- **Addresses**

Addresses	Type of CJ-series Special Unit	Word addresses	Words per Unit
Range	CJ-series Special I/O Units	D20000 to D29599 (100 words × 96 unit numbers)	100 words
	CJ-series CPU Bus Units	D30000 to D31599 (100 words × 16 unit numbers)	100 words

The words that are allocated are listed in the following table.
CJ-series Special I/O Units

Word addresses	Unit Number
D20000 to D20099	0
D20100 to D20199	1
to	to
D29500 to D29599	95

CJ-series CPU Bus Units

Word addresses	Unit Number
D30000 to D30099	0
D30100 to D30199	1
to	to
D31500 to D31599	F

For details on how to use the allocated words, refer to the operation manual for the Special Unit.

Additional Information

You can access the DM Area words that are allocated to Special Units in NJ-series CPU Units through the device variables that are allocated to I/O ports. We therefore recommend that you do not use AT specifications to access this area.

A-7-5 EM Area

● Description

This is a general-purpose data area used to read and write 16-bit words. You can also add a bit number to address specify bits. Data in this area is retained during power interruption or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses
Range	NJ501-□□□□: E0_0 to E18_32767 NJ301-□□□□: E0_0 to E3_32767 NJ101-□□□□: E0_0 to E3_32767	NJ501-□□□□: E0_0.00 to E18_3276.15 NJ301-□□□□: E0_0.00 to E3_32767.15 NJ101-□□□□: E0_0.00 to E3_32767.15

Note The number of banks is given in hexadecimal.

A-8 Variable Memory Allocation Methods

You must be aware of the way in which memory is allocated to variables to align the memory locations of the members of structure or union variables with variables in other devices. Adjustments are necessary mainly when structure variables are used in the following type of communications with other devices.

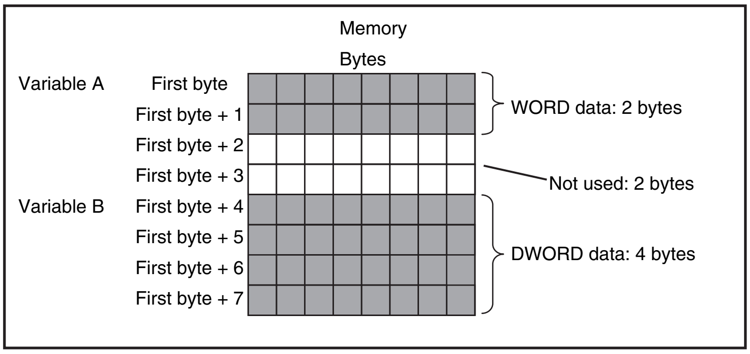
- When using EtherNet/IP tag data links or CIP messages to access variables between NJ/NX-series CPU Units and other CPU Units
- When using structure variables to exchange data with devices other than CPU Units, such as ID Tags

A-8-1 Variable Memory Allocation Rules

The amount of memory and the memory locations that are allocated for a variable depend on the data type of the variable. The amount of memory and the memory locations that are allocated for array elements, structure members, and union members depend on the data types, but also on the declarations that are made for the arrays, structures, and unions.

Data Type Alignment and Memory Allocation Amounts

The data size is determined for each data type. The data size is the minimum amount of memory that is required to store the value or values of that data type. On the other hand, memory for variables is automatically structured by the Controller for the most efficient access. Therefore, the total amount of memory that is required for variables is not necessarily the total of the data sizes of the variables. For example, if WORD and DWORD variables are declared, the total of the data sizes is six bytes, but eight bytes are allocated in memory, as shown in the following figure.



Variable Table

Name	Data type
A	WORD
B	DWORD

This information for determining the location of a variable in memory is called the alignment. The alignment is determined for each data type. The amount of memory and the memory locations for the variables are given below.

Item	Specification
Amount of memory that is allocated	An integral multiple of the alignment. However, the minimum amount of memory is the data size.
Locations in memory	At an integral multiple of the alignment starting from the start of the variable in memory.

The alignments and the amounts of memory that are allocated for the basic data types and enumerations are given below.

Data type	Alignment [bytes]	Amount of memory that is allocated [bytes]
BOOL	2	2
BYTE, USINT, or SINT	1	1
WORD, UINT, or INT	2	2
DWORD, UDINT, or DINT	4	4
LWORD, ULINT, or LINT	8	8
REAL	4	4
LREAL	8	8
TIME, DATE, TIME_OF_DAY, or DATE_AND_TIME	8	8
STRING[N+1][*1]	1	N+1
Enumerations	4	4

*1 N is the maximum number of characters handled. For example, if a maximum of 10 single-byte characters are handled, the NULL character is added, so memory for 11 characters must be reserved.

The elements of arrays and the members of structures and unions are located in memory for the most efficient access. The alignments and the amounts of memory that are allocated for arrays, structures, and unions are determined by the variable declarations, as described below.

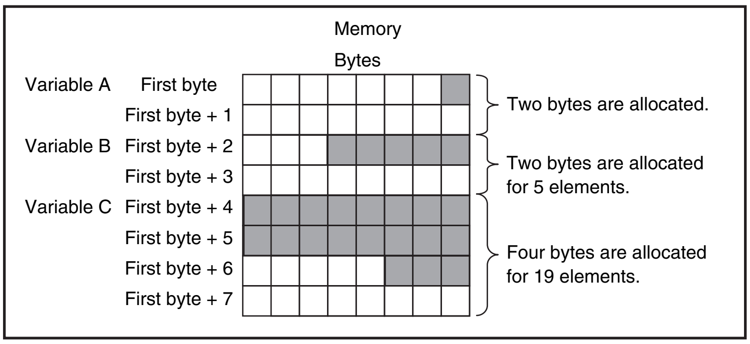
Data type	Alignment	Amount of memory that is allocated
Array	Same as alignment of the data type of the elements	(Amount of memory that is allocated for the data type of the elements) × Number of elements[*]
Structure	The largest alignment of all of the members	The integral multiple of the alignment that is larger than the total amount of memory that is allocated when the members are arranged in order at integral multiples of the alignment of the data types of the members
Union	The largest alignment of all of the members	The largest amount of memory that is allocated for any of the members

* BOOL arrays are an exception. Refer to *Precautions for Correct Use*, below, for the amount of memory that is allocated for BOOL arrays.

Precautions for Correct Use

Amount of Memory That Is Allocated for BOOL Arrays

Two bytes are allocated in memory for individual BOOL variables, BOOL structure members, and BOOL union variables. However, for a BOOL array, two bytes of memory are not allocated for each element. One bit is allocated in order for each element. For the entire array, a multiple of two bytes of memory is allocated (including unused bits).



Variable Table

Name	Data type
A	BOOL
B	ARRAY[1..5]OF BOOL
C	ARRAY[0..18]OF BOOL

Therefore, the following formula gives the amount of memory that is allocated for a BOOL array. For 1 to 16 elements, 2 bytes are allocated. For 17 to 32 elements, 4 bytes are allocated.

$$\text{Amount of memory} = 2\left[\frac{\text{Number of elements} - 1}{16}\right] + 2$$

Truncate the decimal portion of the result of the calculation in brackets.

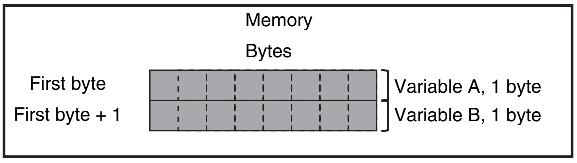
Specific examples of the rules for memory allocation for variables of each data type are given below.

Basic Data Types

● Variables with One-Byte Alignments (e.g., BYTE)

One byte of memory is allocated for the one-byte alignment.

Example: Two consecutive BYTE variables



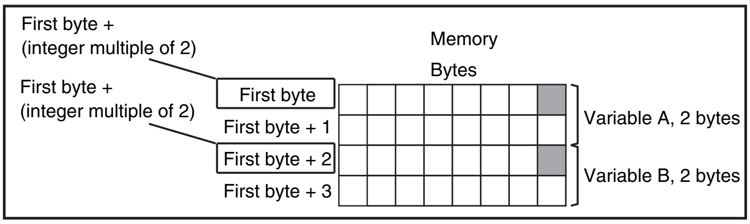
Variable Table

Name	Data type
A	BYTE
B	BYTE

● Variables with Two-byte Alignments (e.g., BOOL and WORD)

Two bytes of memory are allocated for the two-byte alignment.

Example: Two consecutive BOOL variables



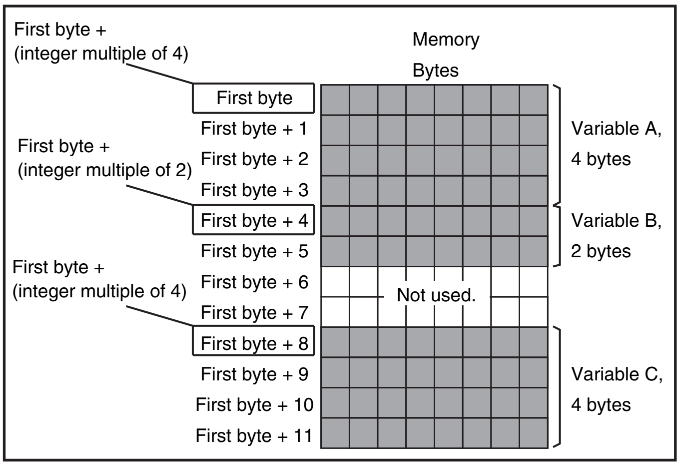
Variable Table

Name	Data type
A	BOOL
B	BOOL

● Variables with Four-byte Alignments (e.g., DWORD)

Four bytes of memory are allocated for the four-byte alignment.
The location of the first byte of data in memory is an integer multiple of four bytes. Therefore, if a variable with a two-byte alignment, such as WORD data, is inserted, two bytes of unused memory will remain.

Example: Consecutive variables in the following order: DWORD, WORD, and DWORD



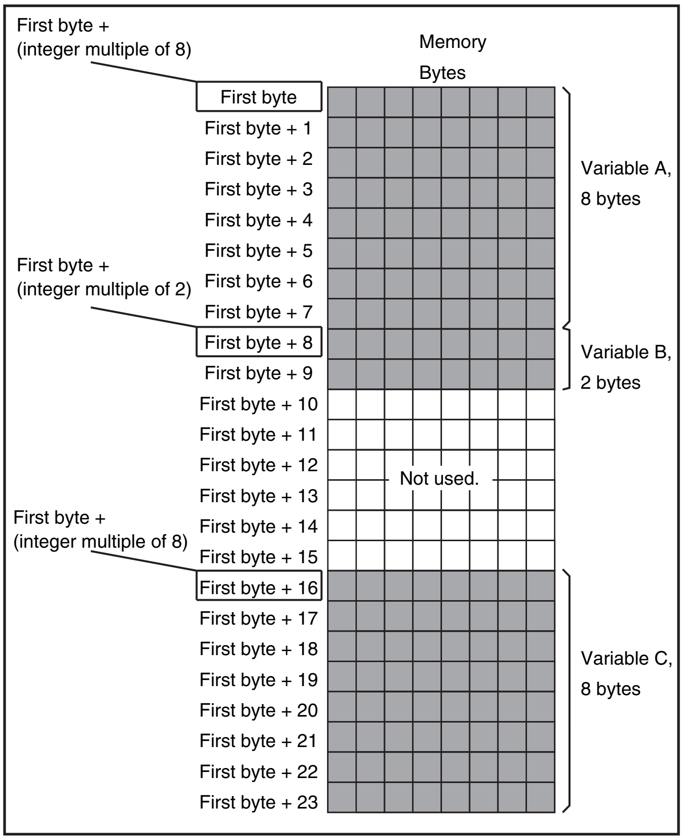
Variable Table

Name	Data type
A	DWORD
B	WORD
C	DWORD

● Variables with Eight-byte Alignments (e.g., LWORD)

Eight bytes of memory are allocated for the eight-byte alignment.
The location of the first byte of data in memory is an integer multiple of eight bytes. Therefore, if a variable with a two-byte alignment, such as WORD data, is inserted, six bytes of unused memory will remain. If a variable with a four-byte alignment, such as DWORD data, is inserted, four bytes of unused memory will remain.

Example: Consecutive variables in the following order: LWORD, WORD, and LWORD



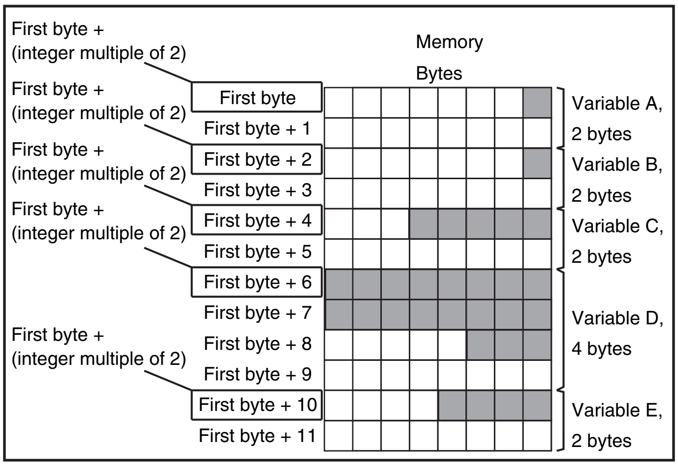
Variable Table

Name	Data type
A	LWORD
B	WORD
C	LWORD

Arrays

A continuous section of memory is allocated for the elements of the array based on the data size of the data type of the array variable. The alignment of an array is the same as alignment of the data type of the elements.

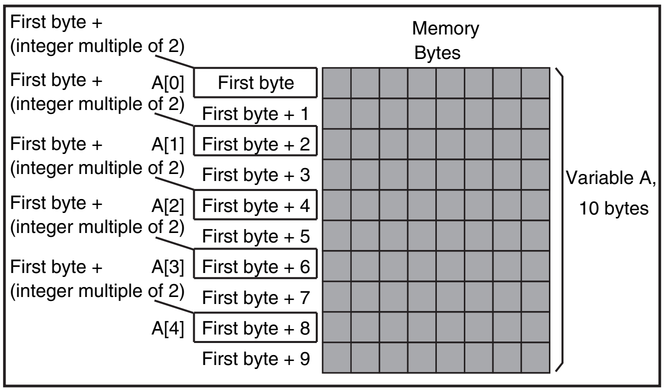
Example: Continuous variables in the following order: two BOOL variable, one BOOL array with five elements, one BOOL array with 19 elements, and one BOOL array with four elements



Variable Table

Name	Data type
A	BOOL
B	BOOL
C	ARRAY[1..5]OF BOOL
D	ARRAY[0..18]OF BOOL
E	ARRAY[5..8]OF BOOL

Example: INT array with five elements



Variable Table

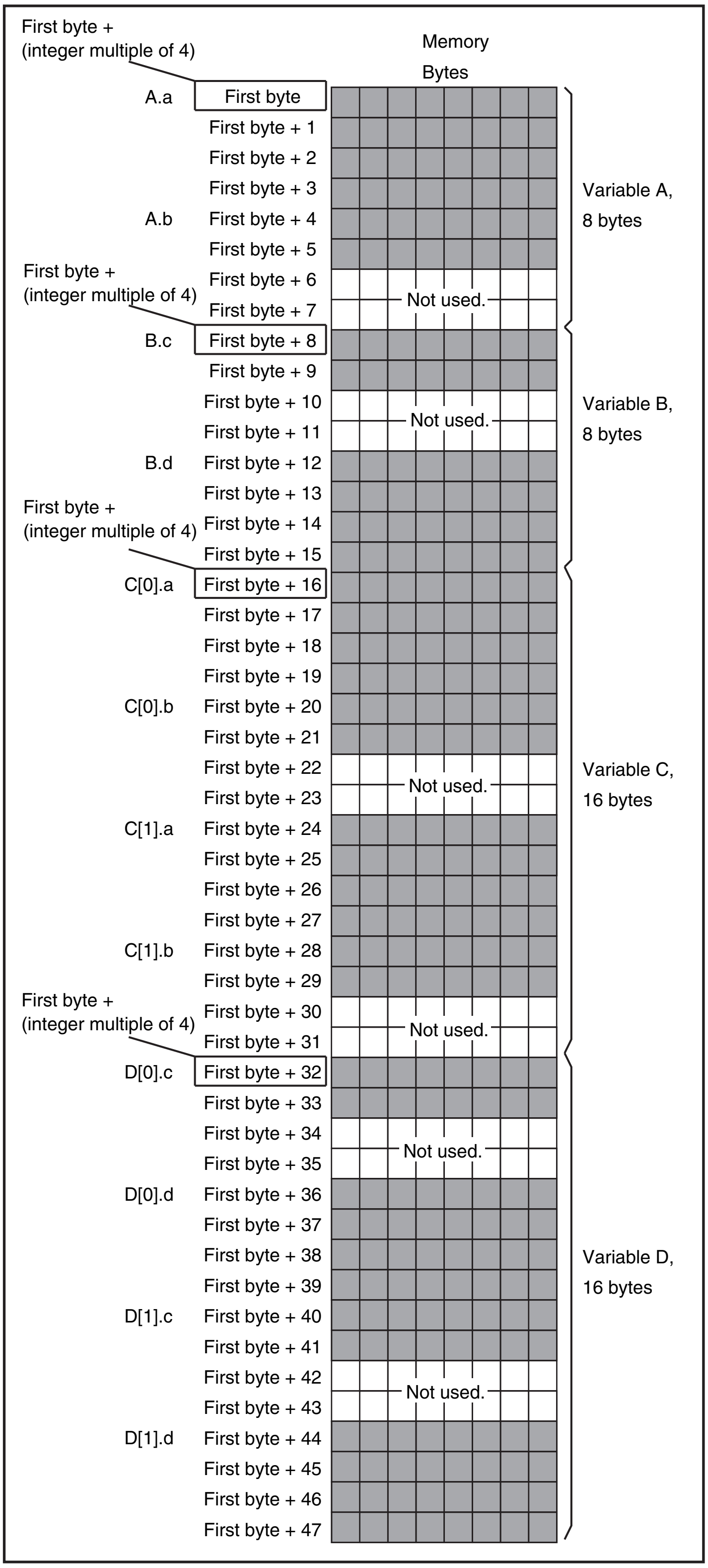
Name	Data type
Variable A	ARRAY[0..4] OF INT

Structures

For a structure variable, the members are located in memory in the order that they are declared. Each member is located at an integer multiple of the alignment of the data type of the member. Therefore, there can be unused memory between members or at the end of members. The alignment of a structure is the largest alignment of all of the members. The amount of memory that is allocated is the integral multiple of the alignment that is larger than the total amount of memory that is allocated when the members are arranged in order at integral multiples of the alignment of the data types of the members.

Example: The alignments and the amounts of memory that are allocated for the four variable declarations given in the following figure are given in the following table.

Variable	Alignment [bytes]	Amount of memory that is allocated [bytes]
A	4	8
B	4	8
C	4	16
D	4	16



Data Type Definitions

Name	Data type
Structure *STR_A*	STRUCT
a	DINT
b	INT

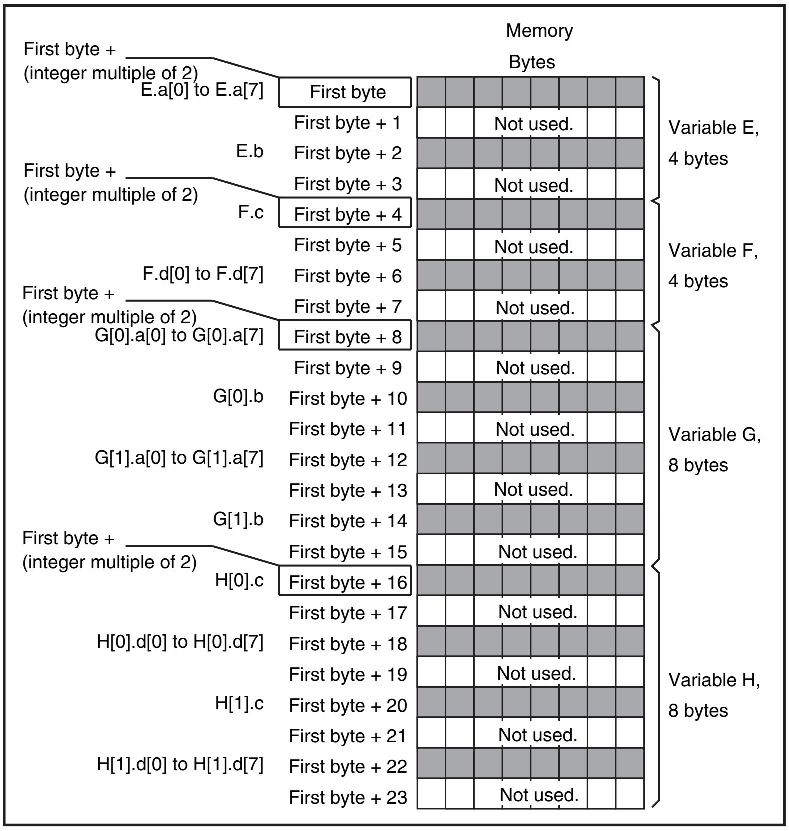
Name	Data type
Structure *STR_B*	STRUCT
c	INT
d	DINT

Variable Table

Name	Data type
Variable A	Structure *STR_A*
Variable B	Structure *STR_B*
Variable C	ARRAY[0..1] OF STR_A
Variable D	ARRAY[0..1] OF STR_B

Example: The alignments and the amounts of memory that are allocated for the four variable declarations given in the following figure are given in the following table.

Variable	Alignment [bytes]	Amount of memory that is allocated [bytes]
E	2	4
F	2	4
G	2	8
H	2	8



Data Type Definitions

Name	Data type
Structure *STR_C*	STRUCT
a	ARRAY[0..7] OF BOOL
b	BYTE

Name	Data type
Structure *STR_D*	STRUCT
c	BYTE
d	ARRAY[0..7] OF BOOL

Variable Table

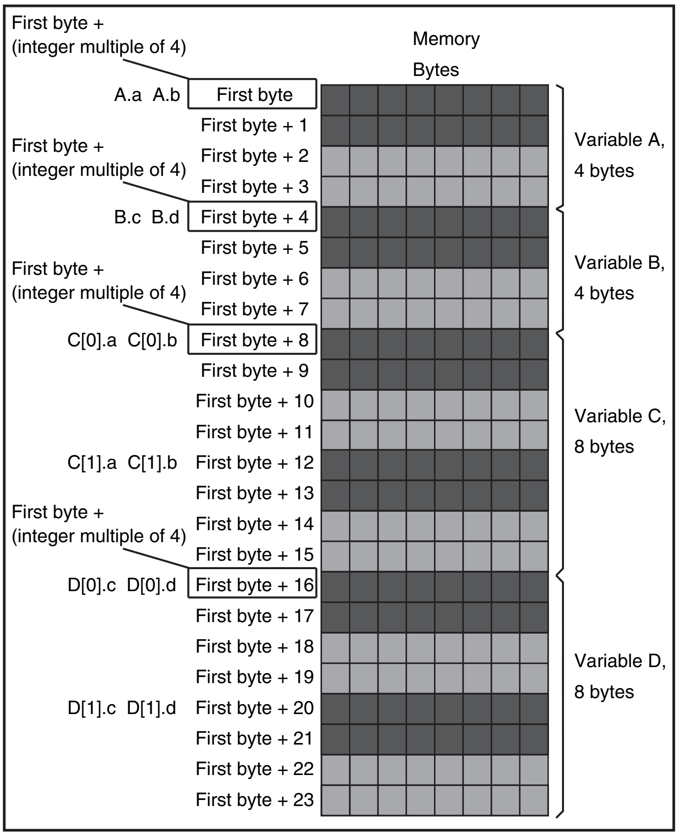
Name	Data type
Variable E	Structure *STR_C*
Variable F	Structure *STR_D*
Variable G	ARRAY[0..1] OF STR_C
Variable H	ARRAY[0..1] OF STR_D

Unions

For a union variable, the members overlap in the same memory locations. The alignment of a union is largest alignment of all of the members. The amount of memory that is allocated is the largest amount of memory that is allocated for any of the members.

Example: The alignments and the amounts of memory that are allocated for the four variable declarations given in the following figure are given in the following table.

Variable	Alignment [bytes]	Amount of memory that is allocated [bytes]
A	4	4
B	4	4
C	4	8
D	4	8



Data Type Definitions

Name	Data type
Union *UNI_A*	UNION
a	DWORD
b	WORD

Name	Data type
Union *UNI_B*	UNION
c	WORD
d	DWORD

Variable Table

Name	Data type
Variable A	Union *UNI_A*
Variable B	Union *UNI_B*
Variable C	ARRAY[0..1] OF UNI_A
Variable D	ARRAY[0..1] OF UNI_B

A-8-2 Important Case Examples

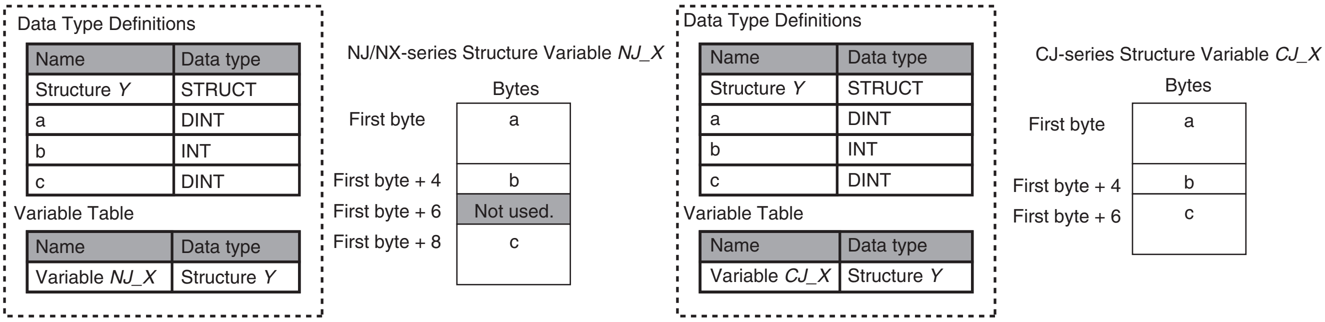
When you exchange structure variable data between an NJ/NX-series CPU Unit and a remote device, you must align the memory configuration of the structure variable members with those of the remote device. This section describes what to do in either the NJ/NX-series CPU Unit or in the remote device.

Additional Information

This is not necessary when you exchange data between NJ/NX-series CPU Units.

Aligning the Memory Configuration with a Remote Device

There are two methods that you can use to align the memory configuration with a remote device. For example, the differences in the memory configuration for structure variables between an NJ/NX-series CPU Unit and a CJ-series CPU Unit are shown below.
This section describes how to align the memory configuration for these Units.



● Method 1: Changing the Memory Configuration of the Structure Variable in the NJ/NX-series CPU Unit

With an NJ/NX-series CPU Unit, you can specify member offsets to change the memory configuration of the members of a structure variable. You can change the memory configuration of the members of a structure variable in the NJ/NX-series CPU Unit so that it is the same as the memory configuration in a remote device that the CPU Unit will communicate with. Specify the member offsets for a structure variable when you register the structure data type.

To communicate with a CJ-series CPU Unit, you can set the offset type to *CJ* to automatically use the CJ-series memory structure. You can set the offset type to *User* to freely set your own offsets.

Version Information

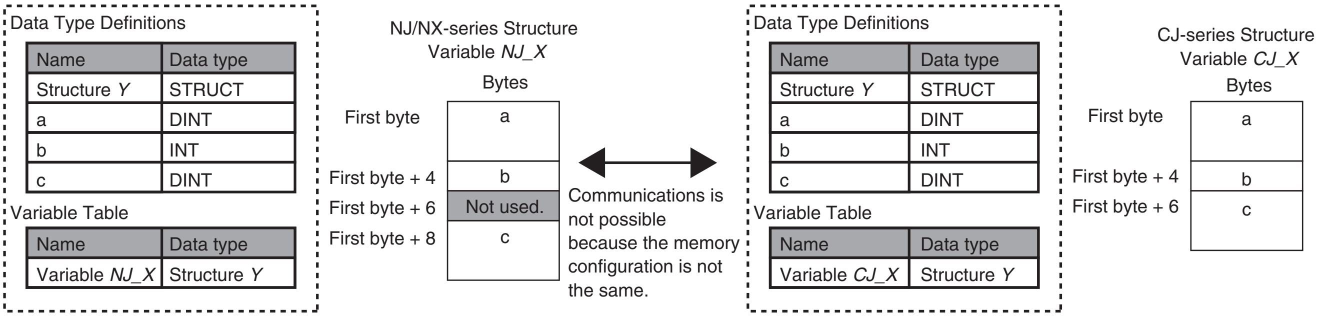
The following table gives the unit version of the CPU Units and the Sysmac Studio version that are required to specify member offsets.

Unit version of CPU Unit	Sysmac Studio version		
	1.01 or lower	1.02	1.03 or higher
1.01 or later	Not possible.	Possible.*	Possible.
1.00	Not possible.	Not possible.	Not possible.

* You cannot select the memory offset type. You can set member offsets.

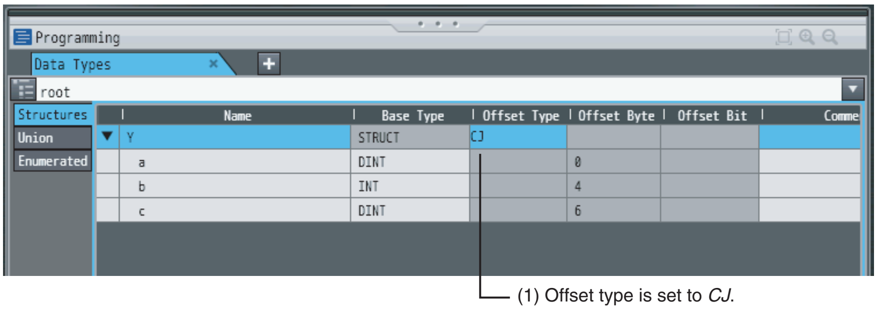
If you change the memory configuration of a structure variable by setting offsets, you must make the same changes for the same structure variable in other NJ/NX-series CPU Units on the network.
Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No W504-E1-03 or later) for the procedure to change the memory configuration of a structure variable.

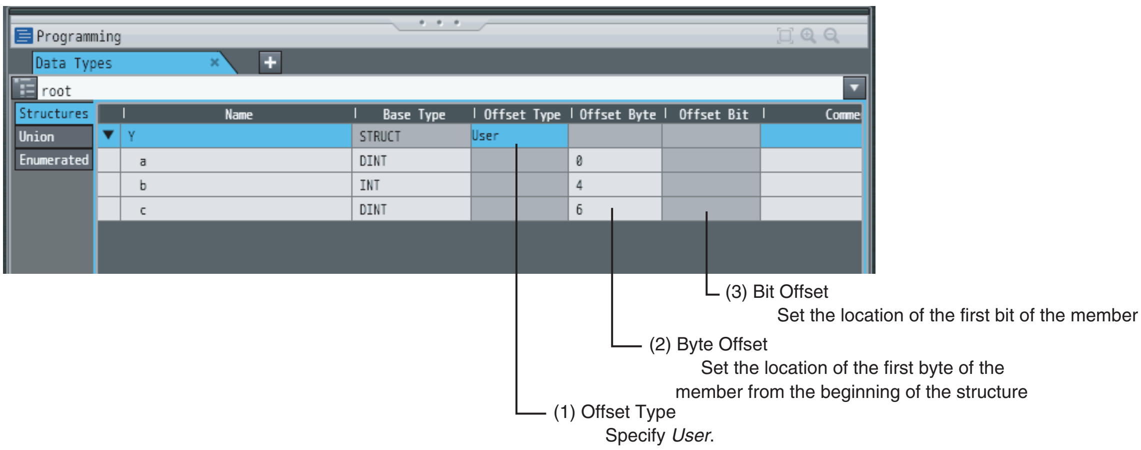
Example: The following example shows how the memory configuration of the structure variable in the CJ-series CPU Unit is changed to match the memory configuration of the structure variable in the NJ/NX-series CPU Unit.

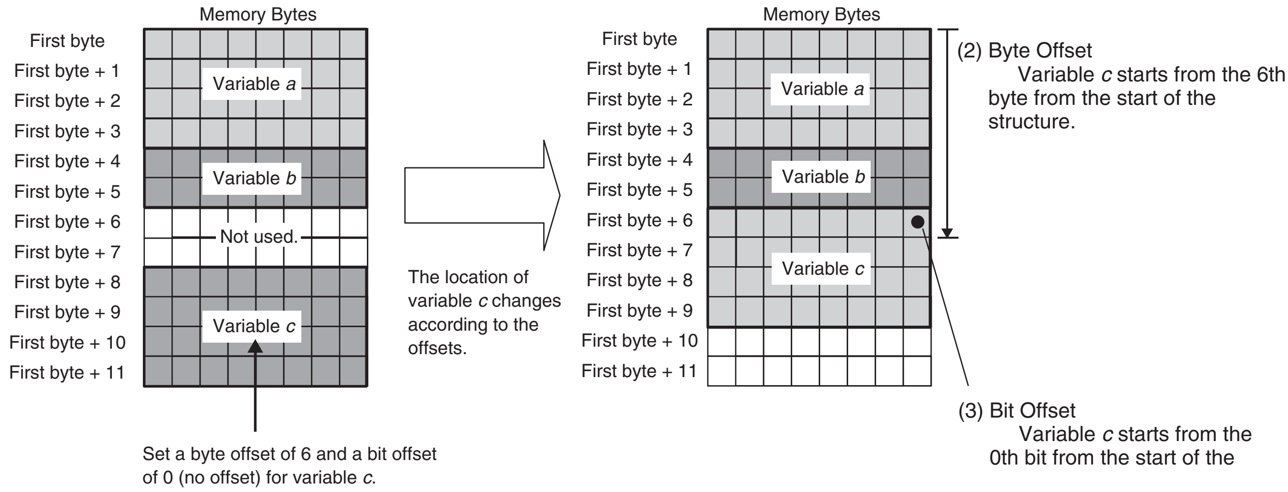


To align the memory configurations in the NJ-series and CJ-series CPU Units, offsets are set in the Sysmac Studio.

Here, the following offsets are set for member c of data type Y of the structure variable NJ_X.



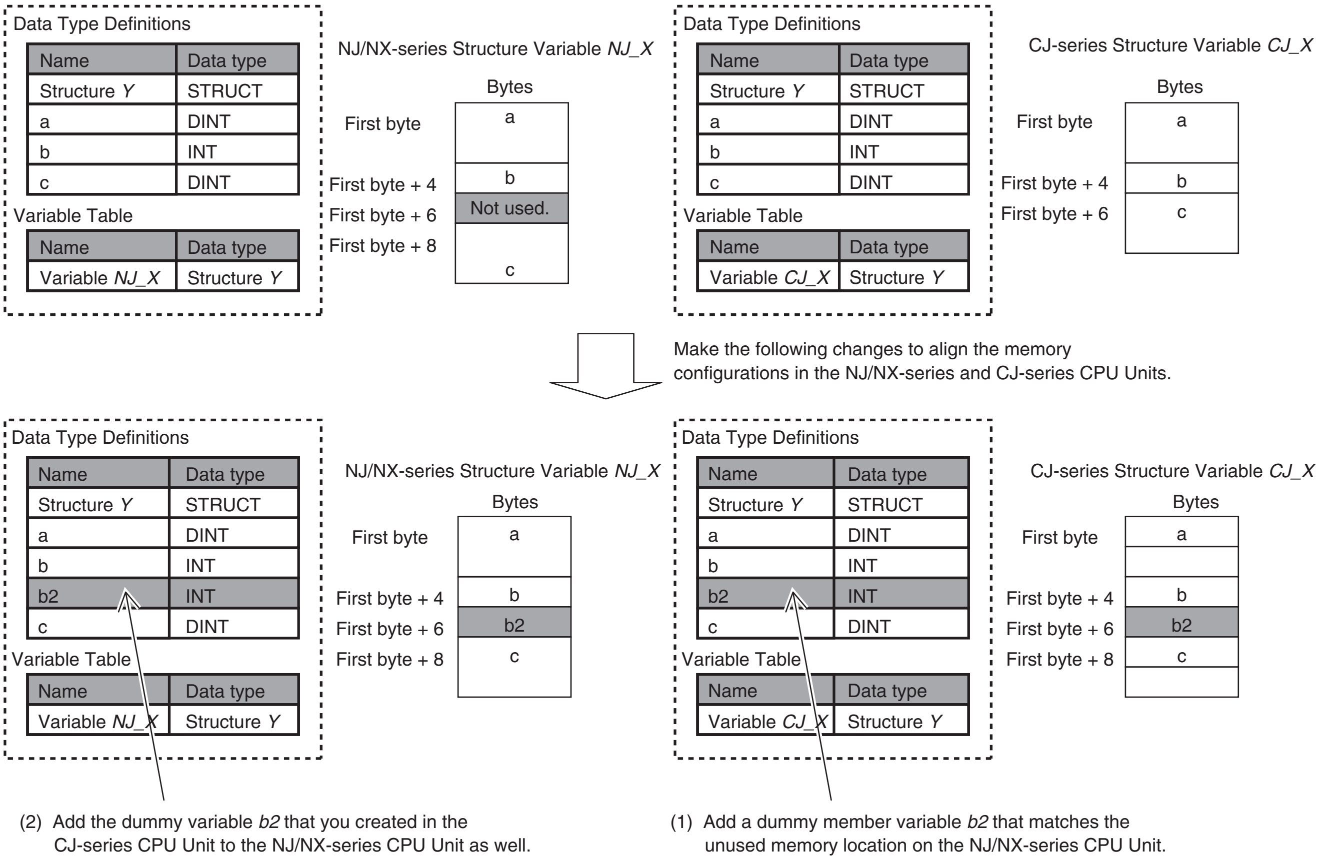




● Method 2: Changing the Memory Configuration of the Structure Variable in the Remote Device

You can insert a member into the structure variable of the remote device to change it to match the memory configuration of the structure variable in the NJ/NX-series CPU Unit. Both the memory configuration and the data types must be the same between the two structure variables. You therefore need to create the same members in both the remote device and the NJ/NX-series CPU Unit.

Example: The following example shows how the memory configuration of the structure variable in the CJ-series CPU Unit is changed to match the memory configuration of the structure variable in the NJ/NX-series CPU Unit.



A-9 Registering a Symbol Table on the CX-Designer

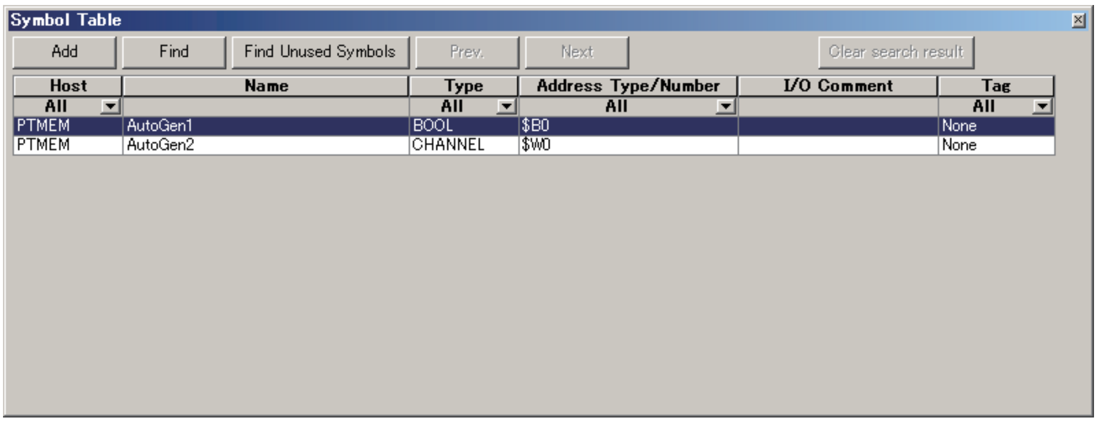
When you connect the NJ/NX-series Controller to an NS-series PT, you can use variables on the CX-Designer to set addresses for the functional objects. The variables are managed in a symbol table. This section shows how to copy a table of variables from a Microsoft Excel spreadsheet to register them all at the same time in a symbol table. Refer to the *CX-Designer User's Manual* (Cat. No. V099) for detailed information on the CX-Designer.

1 **Use the following format to create a table of variables in a Microsoft Excel spreadsheet.**

You must use the same number and arrangement of columns as in the following format. Do not omit any columns even if they are empty, like the *Address type/address* and *I/O comment* columns that are shown below.

Host	Name	Type	Address type/address	I/O comment	Tag
HOST3	_Card1.BkupCmd.ExecBkup	BOOL			TRUE
HOST3	_Card1.BkupCmd.CancelBkup	BOOL			TRUE
HOST3	_Card1.BkupCmd.ExecVefy	BOOL			TRUE
HOST3	_Card1.BkupCmd.CancelVefy	BOOL			TRUE
HOST3	_Card1.BkupCmd.DirName	STRING(64)			TRUE
HOST3	_Card1.BkupSta.Done	BOOL			TRUE
HOST3	_Card1.BkupSta.Active	BOOL			TRUE
HOST3	_Card1.BkupSta.Err	BOOL			TRUE
HOST3	_Card1.VefySta.Done	BOOL			TRUE
HOST3	_Card1.VefySta.Active	BOOL			TRUE
HOST3	_Card1.VefySta.VefyRslt	BOOL			TRUE
HOST3	_Card1.VefySta.Err	BOOL			TRUE
HOST3	_BackupBusy	BOOL			TRUE

2 **Start the CX-Designer and open the Symbol Table Dialog Box.**

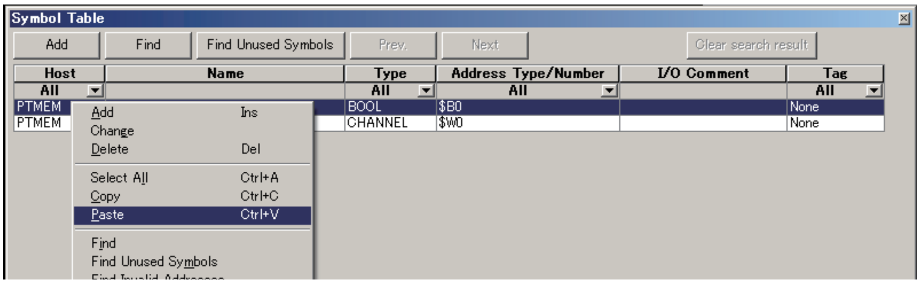


3 **Copy the shaded portion of the Microsoft Excel spreadsheet.**

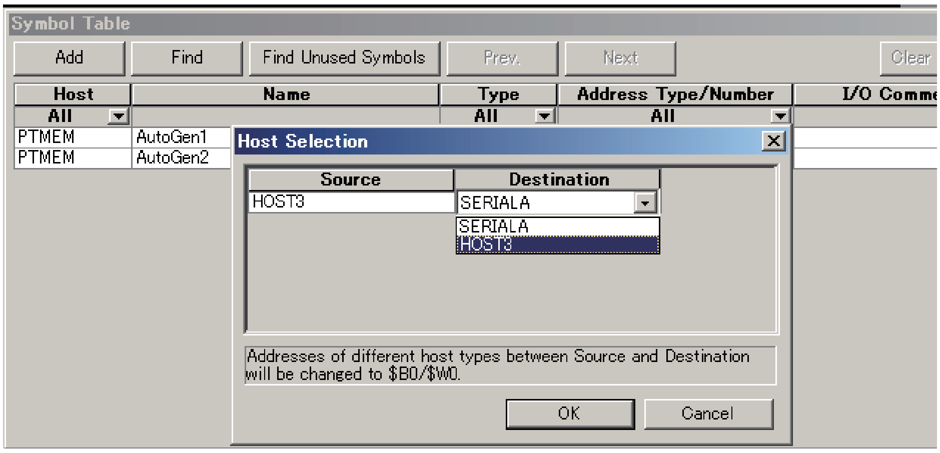
Always copy all of the columns that are shown below.

Host	Name	Type	Address type/address	I/O com-ment	Tag
HOST3	_Card1.BkupCmd.ExecBkup	BOOL			TRUE
HOST3	_Card1.BkupCmd.CancelBkup	BOOL			TRUE
HOST3	_Card1.BkupCmd.ExecVefy	BOOL			TRUE
HOST3	_Card1.BkupCmd.CancelVefy	BOOL			TRUE
HOST3	_Card1.BkupCmd.DirName	STRING(64)			TRUE
HOST3	_Card1.BkupSta.Done	BOOL			TRUE
HOST3	_Card1.BkupSta.Active	BOOL			TRUE
HOST3	_Card1.BkupSta.Err	BOOL			TRUE
HOST3	_Card1.VefySta.Done	BOOL			TRUE
HOST3	_Card1.VefySta.Active	BOOL			TRUE
HOST3	_Card1.VefySta.VefyRslt	BOOL			TRUE
HOST3	_Card1.VefySta.Err	BOOL			TRUE
HOST3	_BackupBusy	BOOL			TRUE

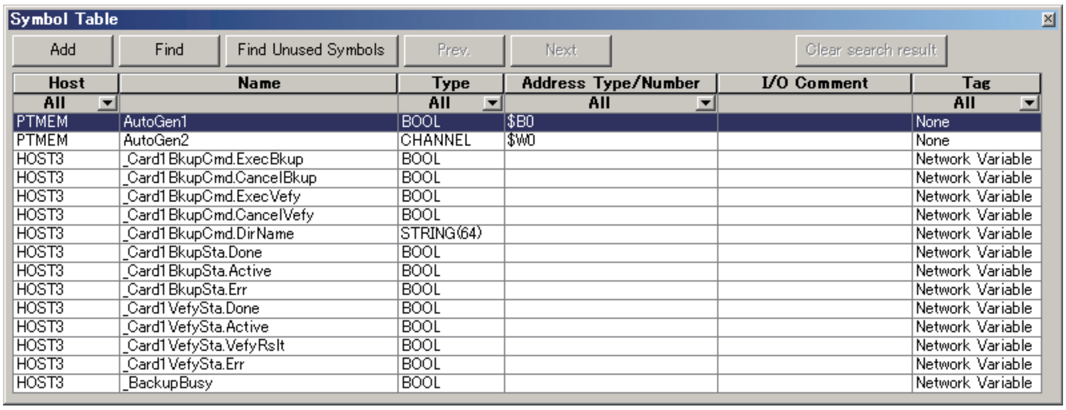
4 **Right-click in the Symbol Table Dialog Box in the CX-Designer and select *Paste* from the menu.**



5 **In the Host Selection Dialog Box on the CX-Designer, select the NJ/NX-series Controller host and then click the OK Button.**



The variables are registered in the Symbol Table Dialog Box of the CX-Designer.



Symbol Table

Add | Find | Find Unused Symbols | Prev. | Next | Clear search result

Host	Name	Type	Address Type/Number	I/O Comment	Tag
All		All	All		All
PTMEM	AutoGen1	BOOL	$B0		None
PTMEM	AutoGen2	CHANNEL	$W0		None
HOST3	_Card1BkupCmd.ExecBkup	BOOL			Network Variable
HOST3	_Card1BkupCmd.CancelBkup	BOOL			Network Variable
HOST3	_Card1BkupCmd.ExecVefy	BOOL			Network Variable
HOST3	_Card1BkupCmd.CancelVefy	BOOL			Network Variable
HOST3	_Card1BkupCmd.DirName	STRING(64)			Network Variable
HOST3	_Card1BkupSta.Done	BOOL			Network Variable
HOST3	_Card1BkupSta.Active	BOOL			Network Variable
HOST3	_Card1BkupSta.Err	BOOL			Network Variable
HOST3	_Card1VefySta.Done	BOOL			Network Variable
HOST3	_Card1VefySta.Active	BOOL			Network Variable
HOST3	_Card1VefySta.VefyRslt	BOOL			Network Variable
HOST3	_Card1VefySta.Err	BOOL			Network Variable
HOST3	_BackupBusy	BOOL			Network Variable

A-10 Enable/Disable EtherCAT Slaves and Axes

You can enable and disable EtherCAT slaves and axes using programming instructions. You can use this for the following types of applications.

- Managing more than one machine with different EtherCAT slave configurations and axis compositions with one project on the Sysmac Studio.
- Leaving one production line running while you change the EtherCAT slave configuration or axis composition of another line.

This section describes the instructions and system-defined variables that are used and provides some application examples.

Version Information

A CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher are required to use the instructions to enable and disable EtherCAT slaves and axes.

A-10-1 Project Settings When Using EtherCAT Slaves and Axes

When you turn ON the power supply or download the project, disable in advance any EtherCAT slaves that may not be installed in the EtherCAT network. Also, set any axes for those EtherCAT slaves to unused axes. If any EtherCAT slaves that are not installed on the EtherCAT network are enabled or if any of their axes are set to used axes, an error will occur when operation is started.

Additional Information

- You can also enable and disable EtherCAT slaves in the following Sysmac Studio settings: **Configurations and Setup** − **EtherCAT** − **Network Configuration** − **Enable/Disable Settings.** If you use the Sysmac Studio settings, however, you would have to use the Sysmac Studio to change the settings every time or you would have to change the project file depending on the machine to handle the application that is described later in *Application 1: Centralized Management of Machines with Different EtherCAT Slave Configuration and Axis Composition* on page A-173.
- You can disable an EtherCAT slave to enable removing it or installing it on the EtherCAT network.

A-10-2 Using Instructions to Enable/Disable EtherCAT Slaves and Axes

You can use instructions in the user program to enable and disable EtherCAT slaves and axes. Separate instructions are used to enable and disable EtherCAT slaves and to enable and disable axes. Both instructions are given in the following table.

Item changed	Instruction
EtherCAT slaves	EC_ChangeEnableSetting (Enable/Disable EtherCAT Slave) instruction
Axes	MC_ChangeAxisUse (Change Axis Use) instruction

Version Information

A CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher are required to use the EC_ChangeEnableSetting and MC_ChangeAxisUse instructions.

EC_ChangeEnableSetting Instruction

The EC_ChangeEnableSetting (Enable/Disable EtherCAT Slave) instruction is used to enable and disable EtherCAT slaves. You can use the EC_ChangeEnableSetting instruction to enable or disable the EtherCAT slave with the specified node address. If you cycle the power supply to the Controller after this instruction is executed, the settings will return to the settings from before instruction execution. Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for the detailed specifications of the EC_ChangeEnableSetting instruction.

MC_ChangeAxisUse Instruction

The MC_ChangeAxisUse (Change Axis Use) instruction is used to enable and disable axes. The MC_ChangeAxisUse instruction changes the setting of the Axis Use axis parameter of the specified axis between *Used Axis* and *Unused Axis.* If you cycle the power supply to the Controller after this instruction is executed, the settings will return to the settings from before instruction execution. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508) for the detailed specifications of the MC_ChangeAxisUse instruction.

A-10-3 System-defined Variables That Indicate EtherCAT Slave or Axis Status

You can check the values of system-defined variables to get the current status of EtherCAT slaves and axes. The system-defined variables for these are given below.

Accessed status	System-defined variable name
EtherCAT slaves	*_EC_DisableSlavTbl[]* (Disabled Slave Table)
Axes	*_MC_AX[].Cfg.AxEnable* (Axis Use)*1

*1 You can also use *_MC1_AX[].Cfg.AxEnable* and *_MC2_AX[].Cfg.AxEnable* with the NX-series CPU Unit.

_EC_DisableSlavTbl[] (Disabled Slave Table)

The *_EC_DisableSlavTbl[]* (Disabled Slave Table) system-defined variable tells whether each EtherCAT slave is currently disabled. The node address is specified for the array subscript. The meanings of the values in *_EC_DisableSlavTbl[]* (Disabled Slave Table) are given below.

Value	Meaning
TRUE	The EtherCAT slave with the specified node address is disabled.
FALSE	The EtherCAT slave with the specified node address is enabled.

_MC_AX[].Cfg.AxEnable (Axis Use)

The *_MC_AX[].Cfg.AxEnable* (Axis Use) system-defined variable tells whether each axis is defined and whether each axis is used. The axis number is specified for the array subscript. The meanings of the values in *_MC_AX[].Cfg.AxEnable* (Axis Use) are given below.

Value	Meaning
0: _mcNoneAxis	The specified axis is an undefined axis.
1: _mcUnusedAxis	The specified axis is an unused axis.
2: _mcUsedAxis	The specified axis is a used axis.

A-10-4 Enabling/Disabling Execution of Program

There are certain programs associated with the EtherCAT slaves and axes, which are enabled or disabled. These associated programs must be enabled or disabled as the EtherCAT slaves and axes are enabled or disabled. To enable or disable the program, use the following instructions in the user program.

Function	Instruction
Enable program	PrgStart instruction
Disable program	PrgStop instruction

Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502-E1-10 or later) for the detailed specifications of the PrgStart instruction and PrgStop instruction.

Precautions for Correct Use

When you want to disable the program, first disable the EtherCAT slave and axis which the program is associated with, and then disable the program.

Version Information

A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the PrgStart and PrgStop instructions.

A-10-5 Checking Enabled/Disabled Program

You can use the PrgStatus instruction to check the program is enabled or disabled that is associated with the EtherCAT slave and axis that are enabled and disabled.
Refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502-E1-10 or later) for the detailed specifications of the PrgStatus instruction.

Version Information

A CPU Unit with unit version 1.08 or later and the Sysmac Studio version 1.09 or higher are required to use the PrgStatus instruction.

A-10-6 Settings with the Sysmac Studio

You can also enable/disable the EtherCAT slaves and axes and set to enable/disable the program at the start of operation using the Sysmac Studio. Some applications require that EtherCAT slave status, axis status and program status at the start of operation are set in advance with the Sysmac Studio.

Enabling/Disabling EtherCAT Slaves with Sysmac Studio

Use the following procedure to enable an EtherCAT slave on the Sysmac Studio.

1. Right-click **EtherCAT** under **Configurations and Setup** and select ***Edit*** from the menu.

   The EtherCAT Tab Page is displayed.

2. In the Toolbox, right-click the EtherCAT slave you want to connect and select ***Insert*** from the menu.

   The selected EtherCAT slave is displayed under the EtherCAT master on the EtherCAT Tab Page. Also, the Parameter Settings Area for the EtherCAT slave is displayed on the right side of the EtherCAT Tab Page.

3 Set the value of **Enable/Disable Settings** to ***Enabled*** on the Parameter Settings Area for the EtherCAT slave.

Enabling/Disabling Axis with Sysmac Studio

Use the following procedure to enable an axis on the Sysmac Studio.

1 Right-click **Axis Settings** under **Configurations and Setup** - **Motion Control Setup** and select ***Add*** - ***Axis Settings*** from the menu.

The axis *MC_Axis000(0)* is added under **Axis Settings**.

2 Right-click *MC_Axis000(0)* and select ***Edit*** from the menu.

The Axis Basic Settings Display appears.

3 Set **Axis Use** to ***Used Axis***.

Running/Stopping Program at the Start of Operation with Sysmac Studio

Use the following procedure to execute a program at the start of operation on the Sysmac Studio.

1 Right-click **Task Settings** under **Configurations and Setup** and select ***Edit*** from the menu.

The Task Settings Tab Page is displayed.

2 Click the **Program Assignment Settings** Button.

The Program Assignment Settings Display appears.

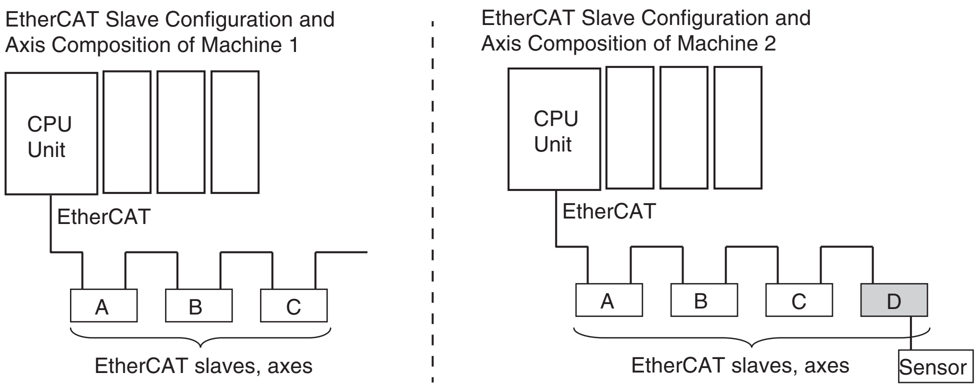
3 Set **Initial Status** of the program to ***Run*** on the Program Assignment Settings Display.

A-10-7 Examples of Applications of Enabling/Disabling EtherCAT Slaves and Axes

This section provides concrete examples of applications in which EtherCAT slaves and axes are enabled and disabled.

Application 1: Centralized Management of Machines with Different EtherCAT Slave Configuration and Axis Composition

Assume that the EtherCAT slaves and axis compositions for the NJ-series Controllers are different for machines 1 and 2 as shown below. These two machines are centrally managed using one Sysmac Studio project.



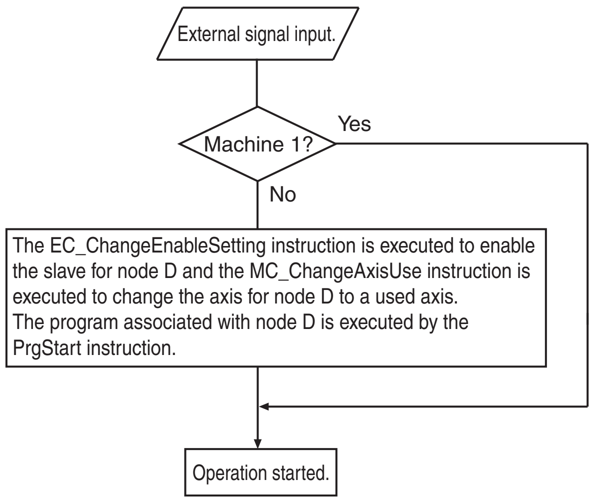
In the Sysmac Studio project, an EtherCAT Slave Configuration is created for all four EtherCAT slaves and axes in A, B, C, and D in the figure. Then, on the Sysmac Studio, you set the EtherCAT slave enable/disable settings, Axis Use parameter settings, and the associated program run/stop status at the start of operation according to machine 1, as shown in the following table.

EtherCAT slave	Installed in EtherCAT network	Enable/disable setting	Axis Use parameter setting	Associated programs
A, B, and C	Installed.	Enabled.	Used Axis	Run at the start of operation.
D	Not installed.	Disabled.	Unused Axis	Stop at the start of operation.

To make changes for machine 2, you use instructions to change the EtherCAT slave enable/disable settings, Axis Use parameter settings, and the associated program enable/disable settings as shown in the following table.

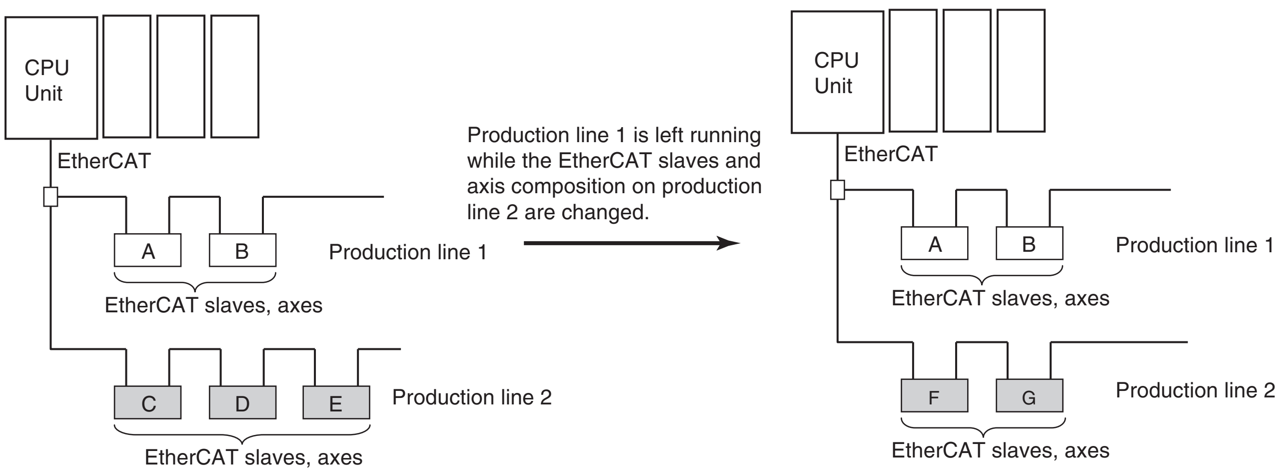
EtherCAT slave	Installed in EtherCAT network	Enable/disable setting	Axis Use parameter setting	Associated programs
A, B, and C	Installed.	Enabled.	Used Axis	Executed.
D	Installed.	Enabled.	Used Axis	Executed.

The user program algorithm is shown in the following figure. A signal is input to the Controller from an external device to specify whether machine 1 or machine 2 is operated.



Application 2: Changing the EtherCAT Slave Configuration and Axis Composition during Operation

In the following figure, production line 1 is left running while the EtherCAT slaves and axis composition on production line 2 are changed.



In the Sysmac Studio project, an EtherCAT slave configuration is created for all seven EtherCAT slaves and axes in A to G in the figure.

On the Sysmac Studio, set the EtherCAT slave enable/disable settings, Axis Use parameter settings, and the associated program run/stop status at the start of operation for nodes A to G as shown in the following table. These are the settings for the configuration before change.

EtherCAT slave	Installed in EtherCAT network	Enable/disable setting	Axis Use parameter setting	Associated programs
A and B	Installed.	Enabled.	Used Axis	Run at the start of operation.
C, E, and D	Installed.	Enabled.	Used Axis	Run at the start of operation.
F and G	Not installed.	Disabled.	Unused Axis	Stop at the start of operation.

The following procedure is used to change the EtherCAT slaves and axes that are used from C, D, E to F and G.

1. Stop production line 4.
2. Use the MC_ChangeAxisUse instruction to set the Axis Use parameters for C, D, and E to *Unused Axis.*
3. Use the EC_ChangeEnableSetting instruction to disable the settings for EtherCAT slaves C, D, and E.
4. Use the PrgStop instruction to disable the programs associated with C, D and E.
5. Remove EtherCAT slaves C, D, and E from production line 4.
6. Install EtherCAT slaves F and G on production line 4.
7. Use the EC_ChangeEnableSetting instruction to enable the settings for EtherCAT slaves F and G.
8. Use the MC_ChangeAxisUse instruction to set the Axis Use parameters for F and G to *Used Axis.*
9. Use the PrgStart instruction to enable the programs associated with F and G.
10. Start production line 4 again.

As the result of the above steps, the EtherCAT slave enable/disable settings, Axis Use parameter settings, and the associated programs enable/disable settings are changed as shown below.

EtherCAT slave	Installed in EtherCAT network	Enable/disable setting	Axis Use parameter setting	Associated programs
A and B	Installed.	Enabled.	Used Axis	Enabled.
C, E, and D	Not installed.	Disabled.	Unused Axis	Disabled.
F and G	Installed.	Enabled.	Used Axis	Enabled.

Precautions for Correct Use

When you want to disable the program, first disable the EtherCAT slave and axis which the program is associated with, and then disable the program.

A-11 Size Restrictions for the User Program

There are size restrictions for the user program due to the limitations of the memory capacity in the CPU Unit and other factors. If you exceed these restrictions, errors will occur during operation. This section describes each of the size restrictions of a user program that is created in a CPU Unit.

You can check the approximate sizes of the user program and variables with the memory display functions of the Sysmac Studio.

Be careful not to exceed these restrictions when you create the user program. The restrictions that are given in this section, however, are only reference values for use as guidelines. We recommend that you ensure ample leeway for the restrictions to allow for the possibility of future user program expansion as well as for other reasons.

Precautions for Correct Use

Errors can occur during online editing even if the user program size restrictions are not exceeded. This is because even if you change the user program with online editing, other data that is allocated in the memory of the CPU Unit may remain. If errors occur, change the Controller to PROGRAM mode and transfer the user program to the Controller again to reset the errors.

A-11-1 User Program Object Restrictions

This section describes the restrictions to user program objects. There are restrictions for the following objects.

- POU
- Variables
- Data type definitions
- Constants (literals)

POU Restrictions

There are restrictions both on POU definitions and POU instances.

● POU Definition Restrictions

POU definitions are subject to the following restrictions.

Restriction	CPU Unit model			
	NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum number of programs	1,000	500	500	500
Maximum value of the following: Number of function block definitions + Number of function definitions + Number of ladder diagram sections	6,000	3,000	750	450
Maximum total number of input, output, and in-out variables in function block and function definitions	64	64	64	64

● POU Instance Restrictions

POU instances are subject to the following restrictions. The maximum number of POU instances depends on the model and unit version of the CPU Unit and the version of the Sysmac Studio.

Maximum Number of POU Instances for the NX701-□□□□

Restriction	CPU Unit model
	NX701-□□□□
Maximum number of POU instances	48,000

Maximum Number of POU Instances for the NJ501-□□□□

Restriction	Version	CPU Unit model
	Sysmac Studio	NJ501-□□□□
Maximum number of POU instances	1.05 or lower	6,000
	1.06 or higher	9,000

Maximum Number of POU Instances for the NJ301-□□□□

Restriction	Unit version/version		CPU Unit model
	CPU Unit	Sysmac Studio	NJ301-□□□□
Maximum number of POU instances	---	1.04 or lower	1,500
	1.03 or earlier	1.05 or higher	2,400
	1.04 or later		3,000

Maximum Number of POU Instances for the NJ101-□□□□

Restriction	CPU Unit model
	NJ101-□□□□
Maximum number of POU instances	1,800

Refer to *Number of POU Instances* on page A-179 for information on counting POU instances.

Restrictions to Variables

There are restrictions to both variable usage and variable definitions.

● Restrictions to Variable Usage

The usage of variables is subject to the following restrictions.

Restriction	CPU Unit model			
	NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum total size in Mbytes of variables without a Retain attribute*1	256	4	2	2
Maximum total size*1 in Mbytes of variables with a Retain attribute	4	2	0.5	0.5
Maximum number of variables*2 without a Retain attribute	360,000	90,000	22,500	22,500

Restriction	CPU Unit model			
	NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum number of variables*3 with a Retain attribute	40,000	10,000	*4	5,000
Maximum number of network variables	40,000	40,000	27,500	27,500

*1 The data size of each variable depends on its data type. Refer to *6-3-5 Data Types* for the sizes of the data types.

*2 Refer to *Number of Variables without a Retain Attribute* on page A-179 for information on counting variables.

*3 Refer to *Number of Variables with a Retain Attribute* on page A-180 for information on counting variables.

*4 The restriction depends on the Sysmac Studio version and the unit version of the CPU Unit as follows: Sysmac Studio version 1.05 or higher and CPU Unit with unit version 1.04 or later: 5,000
Other combinations: 2,500

● Restrictions to Variable Definitions

Variable definitions are subject to the following restrictions.

Restriction	CPU Unit model			
	NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum number of elements per array	65,535	65,535	65,535	65,535
Maximum number of dimensions in an array	3	3	3	3
Maximum size in MB of an array	8	4	2	2
Maximum value of a subscript (element number) for an array	65,535	65,535	65,535	65,535
Maximum size in bytes*1 of a string variable	1,986	1,986	1,986	1,986

*1 The NULL character at the end must be counted. Therefore, there are 1,985 single-byte characters in a string that has a size of 1,986 bytes.

Restrictions to Data Type Definitions

Data type definitions are subject to the following restrictions.

Restriction	CPU Unit model			
	NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum number*1 of data type definitions	8,000	2,000	1,000	1,000
Maximum number of levels in a structure definition	8	8	8	8
Maximum number of members in a structure definition	2,048	2,048	2,048	2,048
Maximum size in MB of a structure variable	8	4	2	2
Maximum number of members in a union definition	4	4	4	4
Maximum number of enumerators in an enumeration definition	2,048	2,048	2,048	2,048

*1 Refer to *Number of Data Type Definitions* on page A-180 for information on counting data types.

Restrictions to Constants (Literals)

The constants (literals) are subject to the following restrictions.

Restriction	CPU Unit model			
	NX701-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum size in bytes of a constant (literal)	1,985	1,985	1,985	1,985

A-11-2 Counting User Program Objects

This section describes how to count POU instances, variables with a Retain attribute, variables without a Retain attribute, and data type definitions. The information in this section is provided only as guidelines. The methods for counting objects sometimes varies with the unit version of the CPU Unit. Always use the Sysmac Studio to confirm that user program object sizes are suitable.

Number of POU Instances

POU instances are counted as described below.

● Objects Counted as POU Instances

The following objects are counted as POU instances.

- Programs
- Function block instances (both user-created instances and instructions are included)
- Functions (both user-created instances and instructions are included)

● Precautions in Counting POU Instances

Observe the following precautions when you count POU instances.

- If n instances of a function block are used for the same function block definition, count them as n instances.
- If the same function is used more than once in the same task, count them as one instance regardless of the actual number of functions.
- If the same function is used in different tasks, count them as one instance for each task.

Number of Variables without a Retain Attribute

Variables without a Retain attribute are counted as described below.

● Objects Counted as Variables without a Retain Attribute

The following objects are counted as variables without a Retain attribute.

- Global variables without a Retain attribute
- Local variables without a Retain attribute in programs and function block instances (both user-created instances and instructions are included)

● Precautions in Counting Variables without a Retain Attribute

Observe the following precautions when you count variables without a Retain attribute.

- Count arrays as one variable each regardless of the number of elements.
- Count function block instances as one variable. Both user-created instances and instructions are included for function block instances.

- Count arrays of function block instances as one variable each regardless of the number of elements. However, count one variable for each element of the array for the number of variables without a Retain attribute that are used in the function block.

Number of Variables with a Retain Attribute

Variables with a Retain attribute are counted as described below.

● Objects Counted as Variables with a Retain Attribute

The following objects are counted as variables with a Retain attribute.

- Global variables with a Retain attribute
- Local variables with a Retain attribute in programs and function block instances (both user-created instances and instructions are included)

● Precautions in Counting Variables with a Retain Attribute

Observe the following precautions when you count variables with a Retain attribute.

- Count arrays as one variable each regardless of the number of elements.
- Do not count arrays of function block instances. However, count one variable for each element of the array for the number of variables with a Retain attribute that are used in the function blocks.

Number of Data Type Definitions

Data type definitions are counted as described below.

● Objects Counted as Data Type Definitions

The following objects are counted as data type definitions.

- User-created structure definitions
- User-created union definitions
- User-created enumeration definitions

A-12 Replacing CPU Units with Unit Version 1.02 or Earlier

An NJ-series CPU Unit with a unit version of 1.02 or earlier does not support the SD Memory Card backup functions and Sysmac Studio Controller backup functions. Therefore, the following work is required to replace a CPU Unit when it fails or to change to a newer version.

Work	Description
Upload the data from the CPU Unit.	Upload the following three types of data from the old CPU Unit. Each of these must be uploaded separately. • Project • Present values of variables and memory • Tag data link tables
Connect the new CPU Unit.	Remove the old CPU Unit from the Controller and connect the new CPU Unit.
Download the data to the CPU Unit.	Download the three types of data that you stored in the computer to the new CPU Unit.

Details on the above work is provided in the following sections.

A-12-1 Uploading the Data from the CPU Unit

Upload the following three types of data from the old CPU Unit. Each of these must be uploaded separately. Use the Sysmac Studio and the Network Configurator.

Data to upload	Contents of data	Support Software
Project	• Unit Configuration and Unit Setup • I/O Map • Controller Setup (Operation Settings and Built-in EtherNet/IP Port Settings) • Motion Control Setup • Cam Data Settings • Event Setup • Task Settings • POUs • Data type definitions and global variable definitions	Sysmac Studio
Present values of variables and memory	• Values of variables with a Retain attribute • Values of the Holding, DM, and EM Areas in the memory for CJ-series Units • Absolute encoder home offsets	Sysmac Studio
Tag data link tables*1	• Tag data link settings for EtherNet/IP	Sysmac Studio Network Configurator*2

*1 You need to upload tag data link tables only when tag data links are set.
*2 Use the Network Configurator with the Sysmac Studio version 1.09 or lower.

Precautions for Correct Use

The following data in the CPU Unit is not included in the project, present values of variables and memory, or tag data link tables. Therefore, you must set them again after you replace the CPU Unit.

- Data Trace Settings
- Controller name
- Operation authority verification
- Time zone setting for the built-in clock

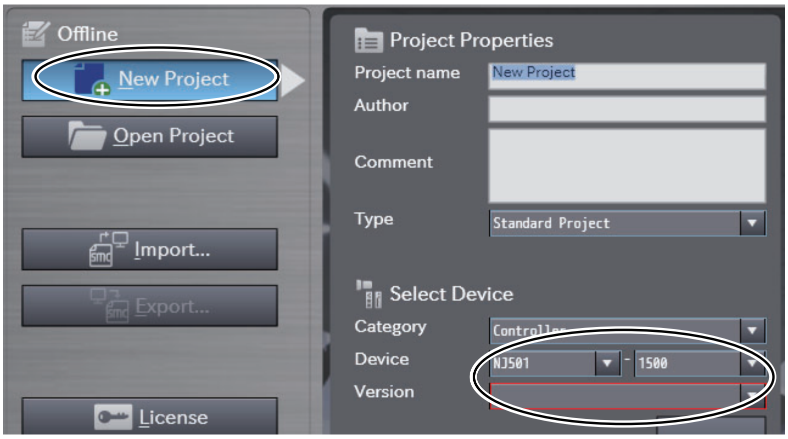
Uploading the Project

Use the following procedure to upload the project.

1 **Start the Sysmac Studio on the computer that is connected to the NJ-series Controller.**

2 **Click the New Project Button and create a new project.**

3 **Select the device and version of the CPU Unit to replace.**



4 **Click the Online Button in the toolbar.**

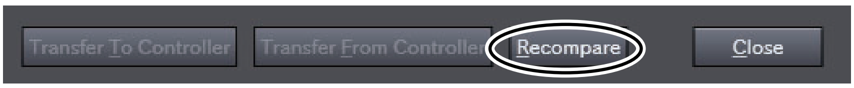
5 **Click the Synchronize Button in the toolbar.**

6 **Clear the Do not transfer the EtherNet/IP connection settings (built-in port and Unit) Check Box and click the Transfer From Controller Button.**

The project in the Controller is uploaded to the computer.

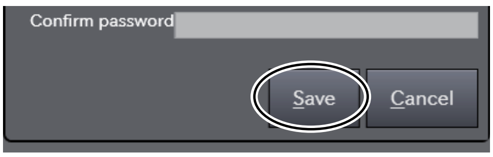
7 **Click the Recompare Button.**

The uploaded project is compared to the project in the Controller.



8 **Click the Save Button.**

The project is saved in the computer.



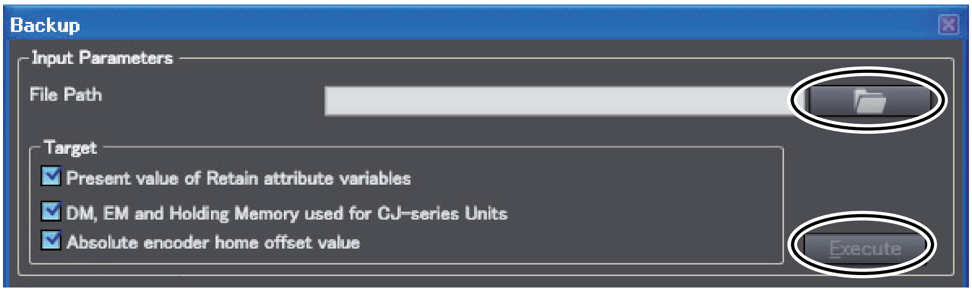
Uploading the Present Values of Variables and Memory

Use the Sysmac Studio's variable and memory backup function to upload the present values of variables and memory. Refer to *9-7 Sysmac Studio Variable and Memory Backup Functions* for details on the Sysmac Studio variable and memory backup functions.

Use the following procedure.

1 **Select *Backup* – *Backup Variables and Memory* from the Tools Menu on the Sysmac Studio.**

2 **Select the *Present value of Retain attribute variable, DM, EM and Holding Memory used for CJ -series Units*, and *Absolute encoder home offset value* Check Boxes and click the Execute Button.**
The variable and memory data is uploaded to the computer.



Uploading Tag Data Link Tables

The Network Configurator is used with the Sysmac Studio version 1.09 or lower.

Use the following procedure to upload the tag data link tables.

1 **Start the Network Configurator on the computer that is connected to the NJ-series Controller.**

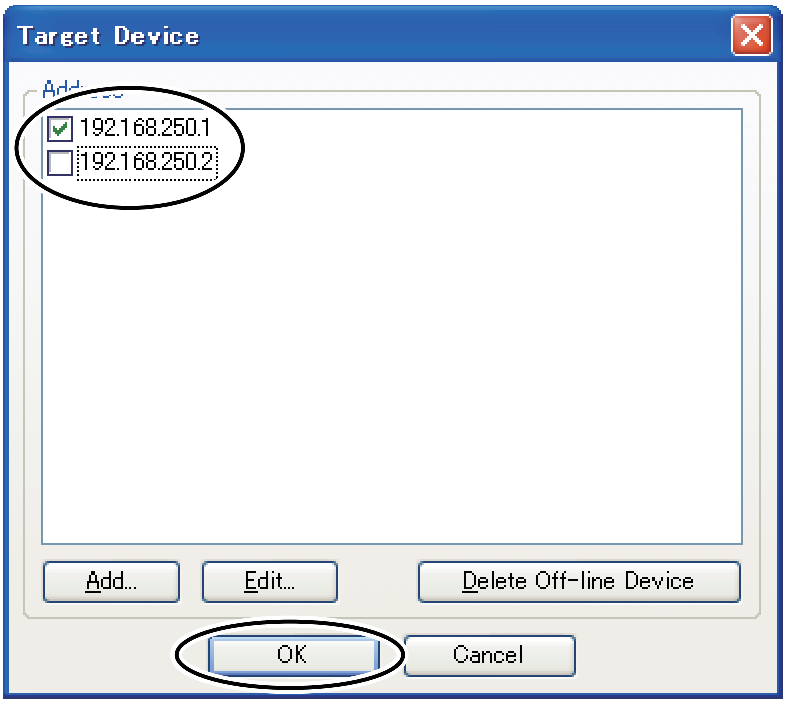
2 **Select *Network* – *Connect* from the toolbar.**

3 **Select *Network* – *Upload* from the toolbar.**

The following message is displayed: "Uploading all devices parameters form network will start based on the current document, OK?"

4 **Click the Yes Button.**

5 **Select only the IP address of the connected CPU Unit as the device and click the OK Button.**



6 **Select *File* – *Save As*.**

The tag data link tables are uploaded to the computer.

A-12-2 Connecting the New CPU Unit

Remove the old CPU Unit and connect the new CPU Unit. Refer to the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) for details on the connection methods.

Perform the following actions for the new CPU Unit.

- Insert the SD Memory Card that was in the old CPU Unit into the new CPU Unit.
- Set the DIP switch to the same settings as the old CPU Unit.

A-12-3 Downloading the Data to the CPU Unit

Download the project, present values of variables and memory, and tag data link tables to the new CPU Unit.

Precautions for Safe Use

Check the operation of the downloaded project for proper execution before you use it for actual operation.

Downloading the Project

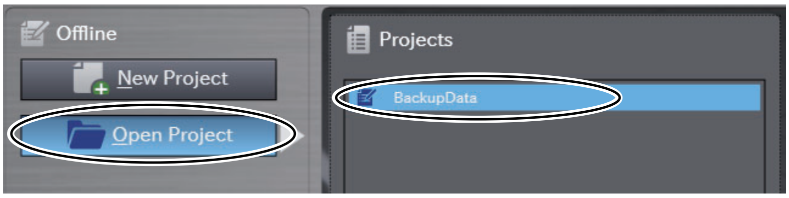
Use the following procedure to download the project.

1 **Start the Sysmac Studio on the computer that is connected to the NJ-series Controller.**

2 **Click the Open Project Button.**

3 **From the project list, select the project that you uploaded from the old CPU Unit.**

In the following example, the name of the project that you uploaded from the previous CPU Unit is *BackupData.*



4 **Click the Online Button in the toolbar.**

5 **Click the Synchronize Button in the toolbar.**

6 **Clear the Do not transfer the EtherNet/IP connection settings (built-in port and Unit) Check Box and click the Transfer To Controller Button.**

The project in the computer is downloaded to the Controller.

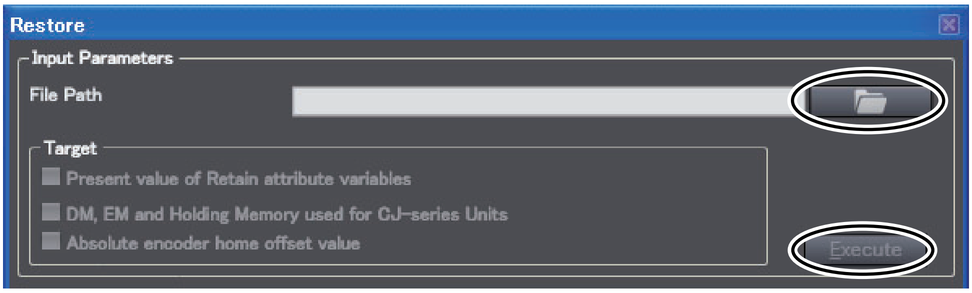
Downloading the Present Values of Variables and Memory

Use the Sysmac Studio's variable and memory backup function to download the present values of variables and memory. Refer to *9-7 Sysmac Studio Variable and Memory Backup Functions* for details on the Sysmac Studio variable and memory backup functions.

Use the following procedure.

1. **Select *Backup* – *Restore Variables and Memory* from the Tools Menu on the Sysmac Studio.**
2. **Select the data file that you uploaded from the old CPU Unit and click the Execute Button.**

   The variable and memory data is downloaded to the Controller.



Downloading Tag Data Link Tables

The Network Configurator is used with the Sysmac Studio version 1.09 or lower.

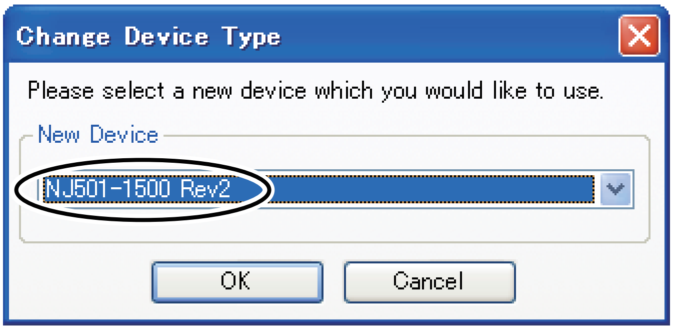
Use the following procedure to download the tag data link tables.

1 **Start the Network Configurator on the computer that is connected to the NJ-series Controller.**

2 **Select *Network* – *Connect* from the toolbar.**

3 **Select *File* – *Open* and open the tag data link file that you uploaded from the old CPU Unit.**

4 **To download tag data link tables to a CPU Unit with unit version 1.03 or later when the tag data link tables were uploaded from a CPU Unit with unit version 1.02 or earlier, select *Device* – *Change Device* and select *NJ□□□-1□□□ Rev2* as the new device.**



5 **Select *Network* – *Download* from the toolbar.**

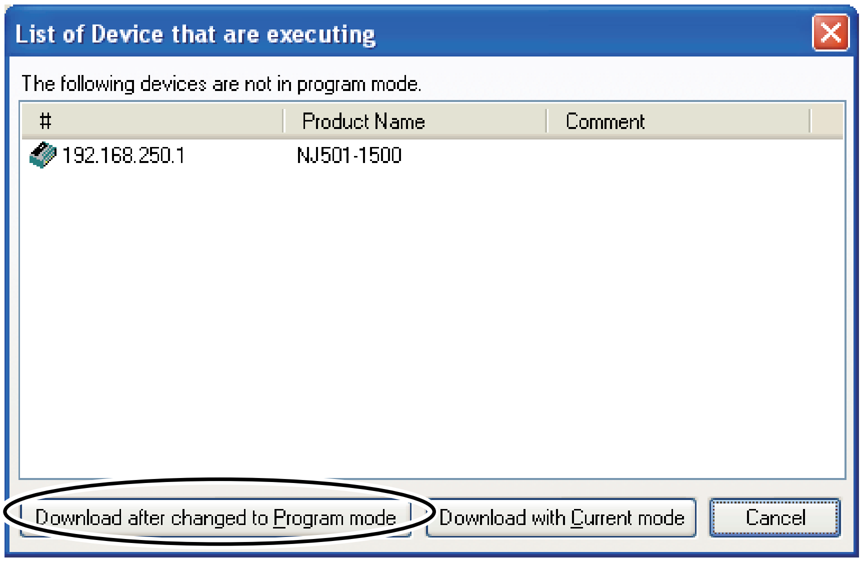
The following message is displayed: “In order to enable new configuration, downloading parameters to all devices will start, OK?”

6 **Click the Yes Button.**

A list of the currently active devices is displayed.

7 **Click the Download after changed to Program mode Button.**

The tag data link tables are downloaded to the Controller.



Precautions for Correct Use

If *NJ□□□-1□□□ Rev2* is not displayed as a new device selection in step 4, above, use the newest version of the Network Configurator.

A-13 Version Information for NX-series Controllers

This section describes the relationship between the unit versions of NX-series CPU Units and the Sysmac Studio versions, and the functions that are supported for each unit version.

This section also describes how the unit versions of NX-series CPU Units correspond to Sysmac Studio versions. Normally use the corresponding versions.

Unit Versions and Corresponding Sysmac Studio Versions

This following table gives the relationship between the unit versions of NX-series CPU Units and the corresponding Sysmac Studio versions.

Unit version of CPU Unit	Corresponding version of Sysmac Studio
Ver. 1.10	Ver. 1.13

Specifications When Not Using the Sysmac Studio Version That Corresponds to the Unit Version of the CPU Unit

The specifications when you do not use the Sysmac Studio version that corresponds to the unit version of the NX-series CPU Unit are given in this section.

● Using Sysmac Studio Version 1.12 or Lower

You cannot use the NX-series CPU Unit with Sysmac Studio version 1.12 or lower.

A-14 Version Information for NJ-series Controllers

This section describes the relationship between the unit versions of NJ-series CPU Units and the Sysmac Studio versions, and the functions that are supported for each unit version.

A-14-1 Relationship between Unit Versions of CPU Units and Sysmac Studio Versions

This section also describes how the unit versions of NJ-series CPU Units correspond to Sysmac Studio versions. Normally use the corresponding versions.

Unit Versions and Corresponding Sysmac Studio Versions

The following table gives the relationship between unit versions of NJ-series CPU Units and the corresponding Sysmac Studio versions.

Unit version of CPU Unit	Corresponding version of Sysmac Studio
Ver.1.10*1	Ver.1.13*2
	Ver.1.12
Ver.1.09	Ver.1.10
Ver.1.08	Ver.1.09
Ver.1.07	Ver.1.08
Ver.1.06	Ver.1.07
Ver.1.05	Ver.1.06
Ver.1.04	Ver.1.05
Ver.1.03	Ver.1.04
Ver.1.02	Ver.1.03
Ver.1.01	Ver.1.02
Ver.1.00*3	Ver.1.01
	Ver.1.00

*1 There is no NJ101-□□□□ CPU Unit with unit version 1.09 or earlier.

*2 Use an NJ101-□□□□ CPU Unit with Sysmac Studio version 1.13 or higher. You cannot use an NJ101-□□□□ CPU Unit with Sysmac Studio version 1.12 or lower.

*3 There is no NJ301-□□□□ CPU Unit with unit version 1.00. Therefore, you cannot use an NJ301-□□□□ CPU Unit with Sysmac Studio version 1.01 or lower.

Specifications When Not Using the Sysmac Studio Version That Corresponds to the Unit Version of the CPU Unit

The specifications when you do not use the Sysmac Studio version that corresponds to the unit version of the NJ-series CPU Unit are given in this section.

● Using a Lower Sysmac Studio Version

If you use a lower version of the Sysmac Studio, you can use only the functions of the unit version of the CPU Unit that corresponds to the Sysmac Studio version.

Example: Unit version of CPU Unit: 1.04
Sysmac Studio version: 1.04
Unit version 1.03 of the CPU Unit corresponds to Sysmac Studio version 1.04. Therefore, you can use only the functions that are supported by unit version 1.03 of the CPU Unit. You cannot use functionality that was added for unit version 1.04 or later of the CPU Unit.

● Using a CPU Unit with an Earlier Unit Version

If you use an NJ-series CPU Unit with an earlier version, select the unit version of the connected CPU Unit or an earlier unit version in the Select Device Area of the Project Properties Dialog Box on the Sysmac Studio. You can use only the functions that are supported by the unit version of the connected CPU Unit.

Example: Unit version of CPU Unit: 1.03
Sysmac Studio version: 1.05
Unit version 1.04 of the CPU Unit corresponds to Sysmac Studio version 1.05. However, the connected CPU Unit is unit version 1.03, so select version 1.03 or earlier as the version in the Select Device Area of the Project Properties Dialog Box. If you select version 1.03 as the version in the Select Device Area of the Project Properties Dialog Box, you can use only the functions that are supported by unit version 1.03 of the CPU Unit. You cannot use functionality that was added for unit version 1.04 or later of the CPU Unit.

Precautions for Correct Use

An error will occur if you perform the following type of operation.

- Create a project on Sysmac Studio version 1.02 or higher with unit version 1.01 or later selected as the version in the Select Device Area of the Project Properties Dialog Box.
- Upload the project to Sysmac Studio version 1.01.

Additional Information

Unit Version Settings for Project Devices

- With Sysmac Studio version 1.02 or higher, you can select the unit version in the Select Device Area of the relevant dialog boxes.
- You can select any unit version that is the same as or earlier than the unit version of the CPU Unit. For example, if the unit version of the CPU Unit is 1.01, select either 1.00 or 1.01.
- The Sysmac Studio will treat the CPU Unit as a CPU Unit with the unit version that is selected for the project device. For example, if you set unit version 1.00 for project device, you can use only the functionality for unit version 1.00 on the Sysmac Studio.
- You can transfer a project to the Sysmac Studio if the unit version that is set for the project device is the same as or earlier than the unit version of the destination CPU Unit.
- Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504-E1-03 or later) for details on the Select Device Area of the relevant dialog boxes.

A-14-2 Functions That Were Added or Changed for Each Unit Version

This section gives the functions that were added or changed for each unit version of NJ-series CPU Unit.

● Additions and Changes to Functional Specifications

The following table gives the unit version of the CPU Units and the Sysmac Studio version for each addition or change to the functional specifications.

Function				Addition/ change	Unit version	Sysmac Studio version	Reference
Tasks	Function	Conditionally executed tasks		Addition	1.03	1.04	*5-2 Overview of Tasks*
Programming	Namespaces			Addition	1.01	1.02	*6-7 Namespaces*
	Data types	Structure data types	Specifying member offsets	Addition	1.01	1.02	*Specifying Structure Member Offsets* on page 6-42
				Change		1.03	
	Libraries			Addition	1.01	1.02	*6-8 Libraries*
Motion control	Single axes	Single-axis position control	Cyclic synchronous absolute positioning	Addition	1.03	1.04	*NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507)
		Auxiliary function for single-axis control	Homing with specified parameters	Addition	1.03	1.04	
			Enabling digital cam switches	Addition	1.06	1.07	
			Command position compensation	Addition	1.10	1.12	
			Start velocity	Addition	1.05	1.06	
	Axes groups	Multi-axes coordinated control	Axes group cyclic synchronous absolute positioning	Addition	1.01	1.02	
		Auxiliary functions for multi-axes coordinated control	Reading axes group positions	Addition	1.01	1.02	
			Changing the axes in a group	Addition	1.01	1.02	
	Common items	Cams	Generating cam tables	Addition	1.08	1.09	
		Parameters	Changing axis parameters	Addition	1.08	1.09	
	Auxiliary functions	Input signal logic inversion		Addition	1.05	1.06	
Unit (I/O) management	NX Units			Addition	1.05	1.06	*NX-series EtherCAT Coupler Unit User's Manual* (Cat. No. W519)

Function				Addition/ change	Unit version	Sysmac Studio version	Reference
Communications	EtherNet/ IP port	TCP/IP applications	FTP client	Addition	1.08	1.09	*NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506)
	EtherCAT port	Packet monitoring[*1] (NJ301-□□□□)		Addition	1.10	1.12	*NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual* (Cat. No. W505)
	Communications instructions			Change	1.08	1.09	*NJ/NX-series Instructions Reference Manual* (Cat. No. W502)
Debugging function	Differential monitoring			Addition	1.03	1.04	*8-5-5 Differential Monitoring*
Reliability functions	Self diagnosis	Controller errors	Changing levels	Addition	1.03	1.04	*8-7 Changing Event Levels*
Security	Asset protection and preventing incorrect operation	Protection	Data protection	Addition	1.01	1.02	*8-4-4 Data Protection*
		Operation authority verification	Number of groups	Change	1.01	1.02	*8-4-5 Operation Authority Verification*
SD Memory Cards	Application	Automatic transfer from SD Memory Card		Addition	1.03	1.04	*9-4 Automatic Transfers from SD Memory Cards*
Backing up data	SD Memory Card backups	Operating methods	CPU Unit front-panel DIP switch	Addition	1.03	1.04	*9-2 SD Memory Card Backups*
			Specification with system-defined variables	Addition	1.03	1.04	
			SD Memory Card Window in Sysmac Studio	Addition	1.03	1.04	
			Special instruction	Addition	1.08	1.09	
		Protection	Disabling backups to SD Memory Cards	Addition	1.03	1.04	*9-3 Disabling Backups to SD Memory Cards*
	Sysmac Studio Controller backups			Addition	1.03	1.04	*9-5 Sysmac Studio Controller Backups*

*1 This addition applies only to an NJ301-□□□□ CPU Unit. The NJ501-□□□□ and NJ101-□□□□ CPU Units support packet monitoring with all versions.

Note Refer to the manuals for the function modules for additions and changes to function module functions for each unit version of the CPU Units.

● Addition of Mountable CJ-series Units

The CJ-series Units that can be mounted have increased for the new unit version of the CPU Unit. For details, refer to *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500).

● Additions and Changes to Basic Instructions and Motion Control Instructions

The basic instructions and motion control instructions that you can use have increased or changed for the new unit version of the CPU Unit. For details, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) and *NJ/NX-series Motion Control Instructions Reference Manual* (Cat. No. W508).

Additions and Changes to Controller Events

The events that can occur have increased or changed for the new unit version of the CPU Unit. There are also changes in the recovery methods to use when some errors occur.
For details, refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503).

Additions and Changes to System-defined Variables

The system-defined variables that you can use have increased or changed for the new unit version of the CPU Unit. Refer to *A-4 System-defined Variables* for details.

A-14-3 Performance Improvements for Unit Version Upgrades

This section introduces the functions for which performance was improved for each unit version of NJ-series CPU Unit and for each Sysmac Studio version.

Function				Performance value	Unit version	Sysmac Studio version
Programming	Program capacity	Quantities	Number of POU instances (NJ501-□□□□)	9,000	---	1.06 or higher
				6,000	---	1.05 or lower
			Number of POU instances (NJ301-□□□□)	3,000	1.04 or later	1.05 or higher
				1,500		1.04 or lower
				2,400	1.03 or earlier	1.05 or higher
				1,500		1.04 or lower
	Memory capacity for variables	Variables with a Retain attribute	Number of variables*1 (NJ301-□□□□)	5,000	1.04 or later	1.05 or higher
				2,500		1.04 or lower
				2,500	1.03 or earlier	---
Motion Control	Number of controlled axes	Maximum number of controlled axes*2*3*4 (NJ301-□□□□)		15 axes	1.06 or later	1.07 or higher
				8 axes (NJ301-1200) 4 axes (NJ301-1100)	Other than the above combination	
		Maximum number of axes for single-axis control*4*5 (NJ301-□□□□)		15 axes	1.06 or later	1.07 or higher
				8 axes (NJ301-1200) 4 axes (NJ301-1100)	Other than the above combination	
Built-in EtherNet/IP port	CIP service: Tag data links (cyclic communications)	Packet interval		Can be set for each connection. 1 to 10,000 ms in 1-ms increments	1.03 or higher	---
				Can be set for each connection. 10 to 10,000 ms in 1-ms increments	1.02 or lower	
		Permissible communications band		3,000 pps*6 (including heartbeat)	1.03 or higher	---
				1,000 pps (including heartbeat)	1.02 or lower	
	Number of TCP sockets			30	1.03 or higher	---
				16	1.02 or lower	
Built-in EtherCAT port	Communications cycle*7 (NJ301-□□□□)			500, 1,000, 2,000, or 4,000 μs	1.03 or higher	---
				1,000, 2,000, or 4,000 μs	1.02 or lower	

*1 The performance improvement applies only to an NJ301-□□□□ CPU Unit. The maximum number of variables with a Retain attributes for the NJ501-□□□□ is 10,000.

*2 This is the total for all axis types.

*3 The performance improvement applies only to an NJ301-□□□□ CPU Unit. The maximum numbers of controlled axes for the NJ501-□□□□ are as follows:
NJ501-1500: 64 axes, NJ501-1400: 32 axes, and NJ501-1300: 16 axes

*4 There is no change in the maximum number of used real axes.

*5 The performance improvement applies only to an NJ301-□□□□ CPU Unit. The maximum numbers of axes for single-axis control for the NJ501-□□□□ are as follows:
NJ501-1500: 64 axes, NJ501-1400: 32 axes, and NJ501-1300: 16 axes

*6 Here, pps means “packets per second” and indicates the number of packets that can be processed in one second.

*7 The performance improvement applies only to an NJ301-□□□□ CPU Unit. You can use 500, 1,000, 2,000 or 4,000 µs communications cycle with an NJ501-□□□□ CPU Unit, and 1,000, 2,000 or 4,000 µs communications cycle with an NJ101-□□□□ CPU Unit.

I

Index

I

Index

D

E

I

I

T

U

V

W

I

OMRON Corporation **Industrial Automation Company**
Tokyo, JAPAN

Contact: www.ia.omron.com

Regional Headquarters

OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

© OMRON Corporation 2011 All Rights Reserved.
In the interest of product improvement,
specifications are subject to change without notice.

Cat. No. W501-E1-12 0415