OMRON

Industrial PC Platform

NY-series

IPC Machine Controller
Industrial Panel PC / Industrial Box PC

Software
User's Manual

NY532-1500
NY532-1400
NY532-1300
NY532-5400
NY512-1500
NY512-1400
NY512-1300

Industrial Panel PC
Industrial Box PC

SYSMAC
always in control

W558-E1-03

NOTE

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, Excel, and Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the United States 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.
- Intel and Intel Core are trademarks of Intel Corporation in the U.S. and / or other countries.

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 NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC.

This manual provides a collective term of Industrial Panel PC and Industrial Box PC which are applicable products as the NY-series Industrial PC. This manual also provides the range of devices that are directly controlled by the Controller functions embedded the Real-Time OS in the NY-series Industrial PC as the Controller.

This manual contains information that is necessary to use the NY-series Controller. Please read this manual and make sure you understand the functionality and performance of the NY-series Controller 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.

- NY-series IPC Machine Controller Industrial Panel PC
  - NY532-15□□
  - NY532-14□□
  - NY532-13□□
  - NY532-5400

- NY-series IPC Machine Controller Industrial Box PC
  - NY512-15□□
  - NY512-14□□
  - NY512-13□□

Part of the specifications and restrictions for the Industrial PC are given in other manuals. Refer to *Relevant Manuals* on page 2 and *Related Manuals* on page 34.

Relevant Manuals

The following table provides the relevant manuals for the NY-series Controller.
Read all of the manuals that are relevant to your system configuration and application before you use the NY-series Controller.
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										
	NY-series IPC Machine Controller Industrial Panel PC Hardware User's Manual	NY-series IPC Machine Controller Industrial Box PC Hardware User's Manual	NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Setup User's Manual	NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Software User's Manual	NY-series Instructions Reference Manual	NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Motion Control User's Manual	NY-series Motion Control Instructions Reference Manual	NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Built-in EtherCAT Port User's Manual	NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Built-in EtherNet/IP Port User's Manual	NJ/NY-series NC Integrated Controller User's Manual	NY-series Troubleshooting Manual
Introduction to NY-series Panel PCs	○										
Introduction to NY-series Box PCs		○									
Setting devices and hardware	○	○									
Using motion control						○					
Using EtherCAT								○			
Using EtherNet/IP									○		
Making setup*1			○								
Making initial settings											
Preparing to use Controllers											
Software settings				○							
Using motion control						○					
Using EtherCAT								○			
Using EtherNet/IP									○		
Using numerical control										○	
Writing the user program				○	○						
Using motion control						○	○				
Using EtherCAT								○			
Using EtherNet/IP									○		
Using numerical control										○	
Programming error processing											○
Testing operation and debugging				○							
Using motion control						○					
Using EtherCAT								○			
Using EtherNet/IP									○		
Using numerical control										○	
Learning about error management and corrections*2											○
Maintenance	○	○									
Using motion control						○					
Using EtherCAT								○			
Using EtherNet/IP									○		

*1 Refer to the *NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat. No. W568) for how to set up and how to use the utilities on Windows.

*2 Refer to the *NY-series Troubleshooting Manual* (Cat. No. W564) for the error management concepts and the error items.

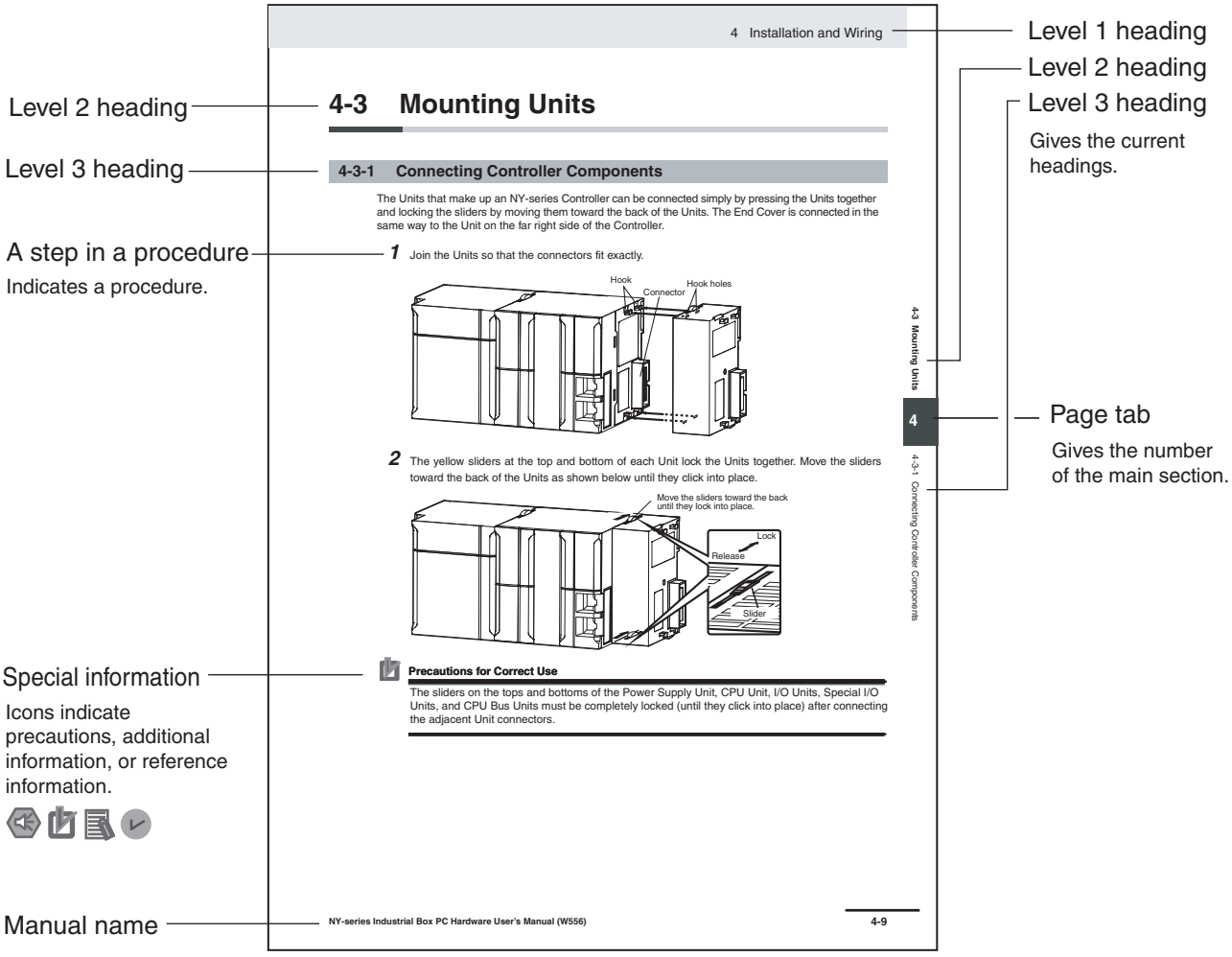
Manual Structure

Some of the descriptions of functions in this manual are common to NJ/NX-series. Therefore, note the following conditions.

- The same function names are used for the common functions of the NJ/NX/NY-series. If the term “CPU Unit” is included in the function names, such as the CPU Unit names, CPU Unit write protection and other functions, it indicates the “Controller” in the NY-series.
- The “CPU Unit” that is described in a list of function specifications in this manual also indicates the “Controller” in the NY-series.

Page Structure

The following page structure is used in this manual.



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 Controller 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 7 Checking Operation and Actual Operation

Section 8 Controller Functions

Section 9 Backup Functions

Section 10 Communications Setup

Section 11 Example of Actual Application Procedures

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 NY-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

Disassembly and Dropping

Do not attempt to disassemble, repair, or modify the product in any way. Doing so may result in malfunction or fire.

Installation

Always connect to a ground of 100 Ω or less when installing the products.

Ensure that installation and post-installation checks of the product are performed by personnel in charge who possess a thorough understanding of the machinery to be installed.

Fail-safe Measures

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

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

The CPU Unit will turn OFF all outputs from 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 a CPU watchdog timer error or CPU reset occurs
- If a major fault level Controller error occurs
- While the product 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 behavior may occur when an error occurs in internal memory of the product. 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. Adjust all of the settings in the slaves and Units.

The use of an uninterruptible power supply (UPS) allows normal operation to continue even if a momentary power failure occurs, possibly resulting in the reception of an erroneous signal from an external device affected by the momentary power failure. Take external fail-safe measures. Where necessary, monitor the power supply voltage on the system for external devices and use it as an interlock condition.

Do not use the input functions of the touchscreen in applications that involve human life, in applications that may result in serious injury, or for emergency stop switches.

Downloading

Always confirm safety at the destination before you transfer a user program, configuration data, setup data, or device variables from the Sysmac studio. The devices or machines may perform unexpected operation regardless of the operating mode of the product.

Actual Operation

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

Security setting adjustments should only be performed by the engineer in charge that possesses a thorough understanding of the security settings. Selecting non-recommended security settings can put your system at risk.

Changing BIOS information is only allowed for the engineer in charge who possesses a thorough understanding of the BIOS settings because it can change the behavior of the product.

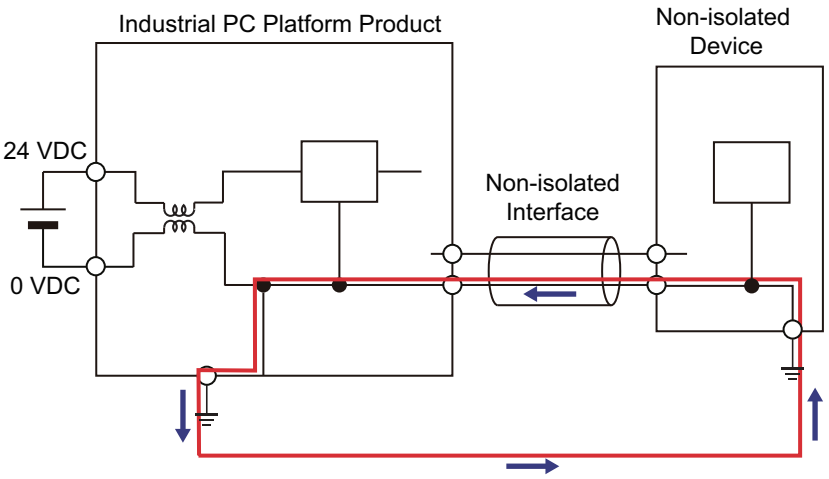
Water or other liquid present on the touchscreen surface may create false touch behavior and unexpected operation. Wipe away the liquid on the touchscreen before operation.

Cautions

⚠ Caution

Wiring

The product has an internal non-isolated DC power supply. Circuit ground (0VDC) and frame ground are connected together. When connecting a non-isolated device or a non-isolated interface to the product, take appropriate actions to avoid communication failures or damage to the mentioned ports.



Never ground the 24 VDC side of the power supply. This may cause a short circuit.

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.

Precautions for Safe Use

Disassembly, Dropping, Mounting, Installation and Storage

- Do not drop the product or subject it to abnormal vibration or shock. Doing so may result in product malfunction or burning.
- When unpacking, check carefully for any external scratches or other damages. Also, shake the product gently and check for any abnormal sound.
- Always use the devices specified in the relevant manual.
- Always install equipment that is included in the product specifications. Not doing so may result in failure or malfunction.
- Install the product in the correct orientation and temperature according to the specifications in the manual to prevent overheating. Not doing so may result in malfunction.
- When connecting peripheral devices to the product, ensure sufficient countermeasures against noise and static electricity during installation of the peripheral devices.
- The product must be installed in a control panel.
- The mounting panel must be between 1.6 and 6.0 mm thick. Tighten the Mounting Brackets evenly to a torque of 0.6 Nm to maintain water and dust resistance. If the tightening torque exceeds the specified value, or the tightening is not even, deformation of the front panel may occur. Additionally, make sure the panel is not dirty or warped and that it is strong enough to hold the product.
- Do not let metal particles enter the product when preparing the panel. Do not allow wire clippings, shavings, or other foreign material to enter any product. Otherwise, the product burning, failure, or malfunction may occur. Cover the product or take other suitable countermeasures, especially during wiring work.

Wiring

- Follow the instructions in the manual to correctly perform connector wiring and insertion. Double-check all wiring and connector insertion before turning ON the power supply.
- Always ensure connectors, cables, PCIe Cards and Storage devices are completely locked in place to prevent accidental disconnection.
- Before you connect a computer to the product, 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 on the product. A difference in electrical potential between the computer and the product may cause failure or malfunction.
- Do not bend or pull the cables beyond normal limit.
  Do not place heavy objects on top of the cables or other wiring lines. Doing so may break the cables.
- Always use power supply wires with sufficient wire diameters to prevent voltage drop and burning. 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.
- Be sure that all mounting bracket screws and cable connector screws are tightened to the torque specified in the relevant manuals. The loose screws may result in fire or malfunction.
- Use crimp terminals for wiring.
- Use a power cable with a conductor cross-section of 0.2mm^2 to 2.5mm^2. Remove 7 mm of sheath before connecting the wires.

- Observe the following precautions to prevent broken wires.
  - When you remove the sheath, be careful not to damage the conductor.
  - Connect the conductor without twisting the wires.
  - Do not weld the conductors. Doing so may cause the wires to break with vibration.

Power Supply Design and Turning ON/OFF the Power Supply

- Always use a power supply that provides power within the rated range.
- Use a DC power supply with a slight voltage fluctuation that will provide a stable output even if the input is momentarily interrupted for 10 ms. Use a DC power supply with reinforced insulation or double insulation. The rated power supply voltage is 24 VDC with an allowable range of 19.2 to 28.8 VDC.
- Do not perform a dielectric strength test.
- Always use the recommended uninterruptable power supply (UPS) to prevent data loss and other system file integrity issues caused by unexpected power interruption. Back up the system files in the planned way to prevent data loss and other system file integrity issues caused by incorrect operation.
- Use an Omron S8BA UPS with the correct revision number to prevent improper system shutdown.
- 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 Output Unit specifications. Implement fail-safe circuits so that external devices do not operate incorrectly.
- Power ON after the DVI cable is connected between the product and an external monitor.
- Always check the power supply and power connections before applying power. Incorrect power connections can damage the product or cause burning.
- Do not turn ON the power supply to the product when a part of a human body or a conductive object is touching the surface of the touchscreen. Doing so will cause the touchscreen functionality to be disabled. Remove the conductive object and cycle the power supply to restore the touchscreen functionality.
- Always turn OFF the power supply to system before you attempt any of the following.
  - Inserting or removing PCIe cards
  - Connecting cables
  - Wiring the system
  - Connecting or disconnecting the connectors
  - Replacing or removing the HDD/SSD.
  - Replace the Battery
  - Replace the Fan Unit

Actual Operation

- Choose a OS password that is not obvious to prevent unauthorized access.
- Remember the OS user name and password. The product is inaccessible without it.
- Before operating the system, please make sure the appropriate software is installed and configured. Doing so may prevent unexpected operation.
- Install all updates and ensure the browser stays up-to-date.
- Install all updates and ensure the firewall stays up-to-date.
- Make sure that your OS environment is protected against malicious software and viruses.
- Install all updates and ensure virus definitions stay up-to-date.
- Do not remove the fan cover while the power is ON. Contact with the rotating fan may result in injury.
- Virtual memory settings can affect the performance of the system. Disable the paging file after installation of applications or updates.
- Correctly perform wiring and setting, and ensure that the shutdown by the UPS can be executed.

- Always create a Rescue Disk using the Rescue Disk Utility and restore to recover the HDD/SSD configuration if necessary.

Operation

- Confirm that no adverse effect will occur in the system before you attempt any of following.
  - Changing the operating mode of the product (including changing the setting of the Startup Mode)
  - Changing the user program or settings
  - Changing set values or present values
  - Forced refreshing
- Do not carry out the following operations when accessing a USB device or an SD Memory Card.
  - Turn OFF the power supply of the product.
  - Press the Power Button of the product.
  - Remove a USB device or SD memory card.
- If two different function modules are used together, such as when you see PCIe connected board 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 Output Unit specifications if a partial fault level error occurs in one of the function modules.
- Do not attempt to remove or touch the fan unit while the product is powered ON or immediately after the power supply is turned OFF. If you attempt to replace the fan unit then, there is a risk of personal injury due to hot or rotating parts.
- Press the power button for several seconds to force the product shutdown. Always back up files in the planned way to prevent data loss or system file corruption.
- Do not touch any product housing when power is being supplied or immediately after the power supply is turned OFF. Doing so may result in burn injury.
- Confirm the safety of the system before using the touch panel.
- Signals from the touchscreen may not be entered if the touchscreen is pressed consecutively at high speed. Only move on to the next operation after confirming that the product has detected the previous input of the touchscreen.
- Do not accidentally press the touchscreen when the backlight is not lit or when the display does not appear. Confirm the safety of the system before pressing the touchscreen.
- Do not use hard or pointed objects to operate or scrub the touchscreen, otherwise the surface of the touchscreen may be damaged.
- Always confirm safety at the connected equipment before you perform the following operations when the device output hold configuration is set to enable. The equipment may operate unexpectedly because the last status for outputs is retained.
  - Changing the operating mode of the product
  - When downloaded

General Communications

- Unexpected operation may result if inappropriate data link tables are set. Even if appropriate data link tables have been set, confirm that the 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.
- Separate the machine network segment from the office network to avoid communication failures.

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.

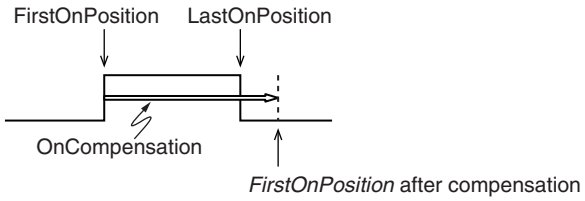
EtherCAT Communications

- 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.
- If the Fail-soft Operation Setting 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. Make sure that the Fail-soft Operation Setting results in safe operation when a device error occurs.
- 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. Refer to the manual for the slave. 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 *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherCAT Port User's Manual* (Cat. No. W562) for details.
- 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.
- 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.
- 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.
- 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.

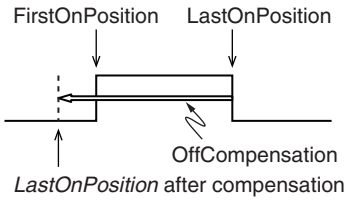
Motion Control

- The motor is stopped if communications are interrupted between the Sysmac Studio and the product during an MC Test Run. Connect the communications cable between the computer and product securely and confirm that the system will not be adversely affected before you perform an MC Test Run.

- 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.
- 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.
- Confirm the axis number carefully before you perform an MC Test Run.
- 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.
- 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.



Restoring Data

- The absolute encoder home offsets are backed up with a non-volatile memory in the product 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.
- You can partially or cannot at all back up, restore, or compare data of the settings depending on 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.

Battery Replacement

- 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.
- The Battery may leak, rupture, heat, or ignite. Never short-circuit, charge, disassemble, heat, or incinerate the Battery or subject it to strong shock.

Fan Unit Replacement

- In the case of an extended storage period, check the performance of the fan unit before production starts.

Product Replacement

- Make sure that the required data, including the user program, configurations, settings and variables is transferred to a product that was replaced and to externally connected devices before restarting operation.
  Be sure to include the tag data link settings and routing tables, which are stored in the product.

Cleaning, Maintenance and Disposal

- Do not use corrosive substances to clean the product. Doing so may result in the failure or malfunction.
- Periodically check the installation conditions in applications where the product is subject to contact with oil or water.
- As the rubber gasket will deteriorate, shrink, or harden depending on the operating environment, periodical inspection is necessary.
- Dispose of the product and Batteries according to local ordinances as they apply.

- 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 product contains a lithium battery with a perchlorate content of 6ppb or higher. When exporting an end product containing the product to or shipping through California, USA, label all packing and shipping containers appropriately.

Precautions for Correct Use

Storage, Installation and Mounting

- Do not operate or store the product 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
  - Locations outdoors subject to direct wind and rain
  - Locations subject to strong ultraviolet light
- Always install the product with sufficient surrounding space to allow for adequate heat dissipation and cooling effect.
- Take appropriate and sufficient countermeasures when installing the product 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
- Always touch a grounded piece of metal to discharge static electricity from your body before starting an installation or maintenance procedure.
- Insert USB devices and PCIe devices correctly to avoid the burning, failure or malfunction.
- Execute a backup of the product before PCIe addition or replacement. Be sure that the PCIe device works correctly before you use them for actual operation. PCIe devices and their related software may cause an OS boot failure or crash.
- The backlight has a finite life and if that is exceeded, the product may fail or malfunction. Check the brightness periodically and if necessary, replace the product.

Wiring

- Always ensure the rated supply voltage is connected to the product.
- Do not allow wire clippings, shavings, or other foreign material to enter the product. Otherwise, burning, failure, or malfunction may occur. Cover the products or take other suitable countermeasures, especially during wiring work.
- Ensure that available software checks are performed by personnel in charge who possess a thorough understanding of the software.
- Do not use cables exceeding the maximum specified length. Doing so may cause malfunction.
- Do not connect an AC power supply to the DC power connector.

Power Supply Design and Turning OFF the Power Supply

- If the product experiences a sudden loss of power or disconnecting the cable while saving a setting or transfer of data is underway, the changes may not be stored and unexpected behavior may occur.

Actual Operation and Operation

- After an OS update or a peripheral device driver update for the product is executed, the product behavior might be different. Confirm that operation is correct before you start actual operation.
- Ensure the fan is operational to provide adequate cooling while the power is turned ON.
- HDD and SSD storage devices, SD Memory Cards, Power button, Fan unit and Battery have finite lives and if those are exceeded, the product may fail or malfunction.
- Always monitor the fan status. If a fan is used beyond its service life, the *Low Revolution Speed* warning message is displayed and the product overheating may occur.
- Monitor the battery's condition. When a battery has low voltage, the system time will be lost.
- You can operate the touchscreen even when you wear some gloves. Confirm that you can correctly operate the touchscreen while wearing gloves prior to actual operation.
- Do not reset or power OFF the product while the password is being changed. If you fail to save the password there is a possibility that the project will not work.
- Always confirm safety at the connected equipment before you reset Controller errors with an event level of partial fault or higher from 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 output 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.
- The functions that are supported depend on the unit version of the product. The version of Sysmac Studio that supports the functions that were added for an upgrade is also required to use those functions. Refer to Version Information for NY-series Controllers for the relationship between the unit versions of the controller and the Sysmac Studio versions, and for the functions that are supported by each unit version.
- The touchscreen supports 5 simultaneous touches. When the number of touches is exceeded, not all touch points will be detected.
- The capacitive touchscreen reacts to contact on its surface. Accidental touching the surface of the touchscreen may cause unintended behavior.
- Confirm the device output hold configuration before you change the operating mode of the product or execute the download.

Error Processing

- In applications that use the results of instructions that read the error status, consider the effect 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

- 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.

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

- 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

- Set the Servo Drives to stop operation if an error occurs in EtherCAT communications between the Controller and a Servo Drive.
- 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.
- 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.
- Make sure that the communications distance, number of devices 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.
- After you transfer the user program, the product 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.

Battery Replacement

- Turn ON the power after replacing the Battery for a product that has been unused for a long time. Leaving the product unused again without turning ON the power even once after the Battery is replaced may result in a shorter Battery life.
- Make sure to use a battery of the correct type, install the battery properly.
- Apply power for at least five minutes before changing the battery. Mount a new battery within five minutes after turning OFF the power supply. If power is not supplied for at least five minutes, the clock data may be lost. Check the clock data after changing the battery.

SD Memory Cards

- Insert an SD memory card completely and ensure it is in place.

Cleaning and Maintenance

- Do not use corrosive substances to clean the product.
- Turn OFF the product or disable the touchscreen for cleaning with water.

Regulations and Standards

Conformance to EU Directives

Applicable Directives

- EMC Directives

Concepts

● EMC Directive

OMRON devices that comply with EU 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 EU 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
EMI (Electromagnetic Interference): EN 61131-2 (Radiated emission: 10-m regulations)

● Conformance to EU Directives

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

- The NY-series Controller must be installed within a control panel.
- You must use the power supply in SELV specifications for the DC power supplies connected to DC Power Supply Units and I/O Units.

NY-series Controllers that comply with EU 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.

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

Hardware revisions and unit versions are used to manage the hardware and software in NY-series Controllers and EtherCAT slaves. The hardware revision or 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 hardware revisions or unit versions.

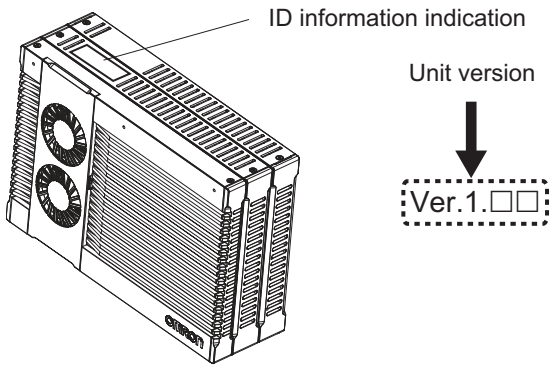
Checking Versions

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 back side of the product.

The ID information on an NY-series NY5□2-□□□□ Controller 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 NY-series Controller

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 Controller.

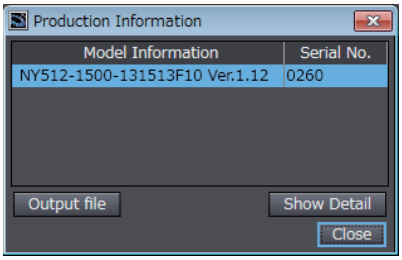
1 Right-click **CPU Rack** under **Configurations and Setup** - **CPU/Expansion Racks** in the Multiview Explorer and select **Production Information**.

The Production Information Dialog Box is displayed.

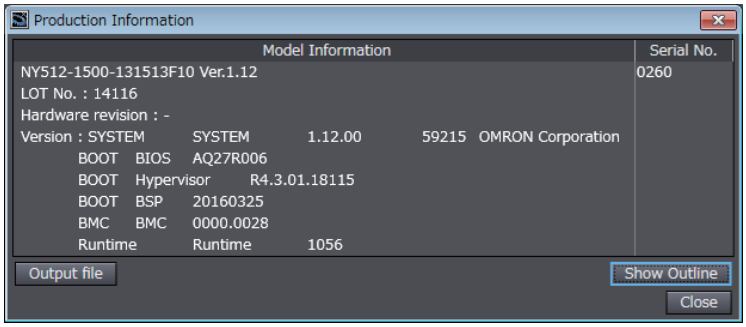
Changing Information Displayed in Production Information Dialog Box

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 revision, and other 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.

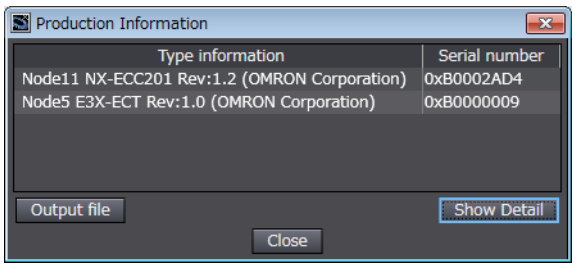
2 Right-click the master on the EtherCAT Tab Page and select **Display Production Information**.

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

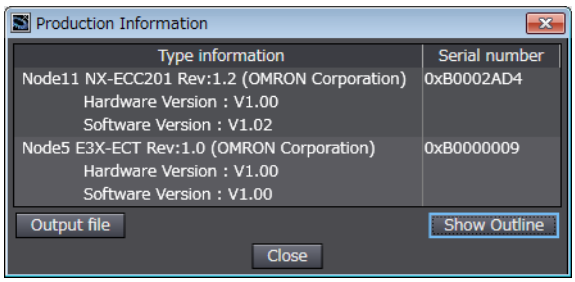
Changing Information Displayed in Production Information Dialog Box

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

Related Manuals

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

Manual name	Cat. No.	Model numbers	Application	Description
NY-series IPC Machine Controller Industrial Panel PC Hardware User's Manual	W557	NY532-□□□□	Learning the basic specifications of the NY-series Industrial Panel PCs, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NY-series system is provided along with the following information on the Industrial Panel PC. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NY-series IPC Machine Controller Industrial Panel PC Hardware User's Manual	W556	NY512-□□□□	Learning the basic specifications of the NY-series Industrial Box PCs, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NY-series system is provided along with the following information on the Industrial Box PC. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Setup User's Manual	W568	NY532-□□□□ NY512-□□□□	Learning the initial settings of the NY-series Industrial PCs and preparations to use Controllers.	The following information is provided on an introduction to the entire NY-series system. • Two OS systems • Initial settings • Industrial PC Support Utility • NYCompolet • Industrial PC API • Backup and recovery
NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Software User's Manual	W558	NY532-□□□□ NY512-□□□□	Learning how to program and set up the Controller functions of an NY-series Industrial PC.	The following information is provided on the NY-series Controller functions. • Controller operation • Controller features • Controller settings • Programming based on IEC 61131-3 language specifications
NY-series Instructions Reference Manual	W560	NY532-□□□□ NY512-□□□□	Learning detailed specifications on the basic instructions of an NY-series Industrial PC.	The instructions in the instruction set (IEC 61131-3 specifications) are described.
NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Motion Control User's Manual	W559	NY532-□□□□ NY512-□□□□	Learning about motion control settings and programming concepts of an NY-series Industrial PC.	The settings and operation of the Controller and programming concepts for motion control are described.
NY-series Motion Control Instructions Reference Manual	W561	NY532-□□□□ NY512-□□□□	Learning about the specifications of the motion control instructions of an NY-series Industrial PC.	The motion control instructions are described.
NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Built-in EtherCAT® Port User's Manual	W562	NY532-□□□□ NY512-□□□□	Using the built-in EtherCAT port in an NY-series Industrial PC.	Information on the built-in EtherCAT port is provided. This manual provides an introduction and provides information on the configuration, features, and setup.

Manual name	Cat. No.	Model numbers	Application	Description
NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Built-in EtherNet/IP™ Port User's Manual	W563	NY532-□□□□ NY512-□□□□	Using the built-in EtherNet/IP port in an NY-series Industrial PC.	Information on the built-in EtherNet/IP port is provided. Information is provided on the basic setup, tag data links, and other features.
NJ/NY-series NC Integrated Controller User's Manual	O030	NJ501-5300 NY532-5400	Performing numerical control with NJ/NY-series Controllers.	Describes the functionality to perform the numerical control. Use this manual together with the *NJ/NY-series G code Instructions Reference Manual* (Cat. No. O031) when programming.
NJ/NY-series G code Instructions Reference Manual	O031	NJ501-5300 NY532-5400	Learning about the specifications of the G code/M code instructions.	The G code/M code instructions are described. Use this manual together with the *NJ/NY-series NC Integrated Controller User's Manual* (Cat. No. O030) when programming.
NY-series Troubleshooting Manual	W564	NY532-□□□□ NY512-□□□□	Learning about the errors that may be detected in an NY-series Industrial PC.	Concepts on managing errors that may be detected in an NY-series Controller and information on individual errors are described.
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.
CNC Operator Operation Manual	O032	SYSMAC -RTNC0□□□D	Learning an introduction of the CNC Operator and how to use it.	An introduction of the CNC Operator, installation procedures, basic operations, connection operations, and operating procedures for main functions are described.
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.
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.
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

The following table provides terminologies related to the Controller functions.

Term	Description
absolute encoder home offset	This data is used to restore in the Controller 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 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.
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 Controller functions embedded in the Real-Time OS in the Industrial Panel PC or Industrial Box PC. In the NY-series System, it is also called "Controller" that includes the EtherCAT slaves (including general-purpose slaves and Servo Drives).
Controller error	Errors that are defined by the NY-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 NY-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 function	The functions to perform machine control that are embedded in the Real-Time OS in the Industrial Panel PC or Industrial Box PC.
Controller information	Information that is defined by the NY-series System that is not an error. It represents an information Controller event.
derivative data type	A data type that is defined by the user. Structures, unions, and enumerations are derivative data types.

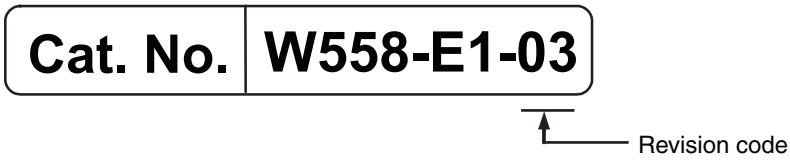
Term	Description
device variable	A variable that is used to access a specific device through an I/O port.
download	To transfer data from the Sysmac Studio to the Controller with the synchronization operation of the Sysmac Studio.
edge	One of the attributes of a variable. This attribute makes a BOOL variable 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 Controller.
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 Controller 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 Controller 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.
main memory	The memory inside the Controller that is used by the Controller to execute the OS and user program.

Term	Description
major fault level Controller error	An error for which all NY-series Controller control operations stop. The Controller 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.
minor fault level Controller error	An error for which part of the control operations for one of the function modules in the NY-series Controller stop. An NY-series Controller 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 NY-series Controller stop. An NY-series Controller 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 and sends commands to the Motion Control Function Module.
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.
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 Controller 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.

Term	Description
Servo Drive/encoder input slave	Any of the EtherCAT slaves that is assigned to an axis. In the NY-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 NY-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 trouble-shooting NY-series Controllers. It also provides operations for motion control and a Simulator.
system common processing	System processing that is performed by the Controller 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	Non-task related processing that is performed by the Controller. The system service includes communications processing, Virtual 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 Communications Coupler Unit.
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 Controller 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 NY-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.
Virtual SD Memory Card	A shared folder for the Controller functions to handle files. An SD Memory Card is used in the NJ/NX-series CPU Unit which is treated as the Controller for Sysmac. In the NY-series Controller, however, a shared folder with Windows is handled as a Virtual SD Memory Card. Therefore, the SD Memory Card is sometimes given without "virtual".

Revision History

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



Revision code	Date	Revised content
01	September 2016	Original production
02	April 2017	• Added information on the functions supported by unit version 1.14 of the Controllers. • Corrected mistakes.
03	October 2017	• Added information on the functions supported by unit version 1.16 of the Controllers. • Corrected mistakes.

1

Introduction to NY-series Controllers

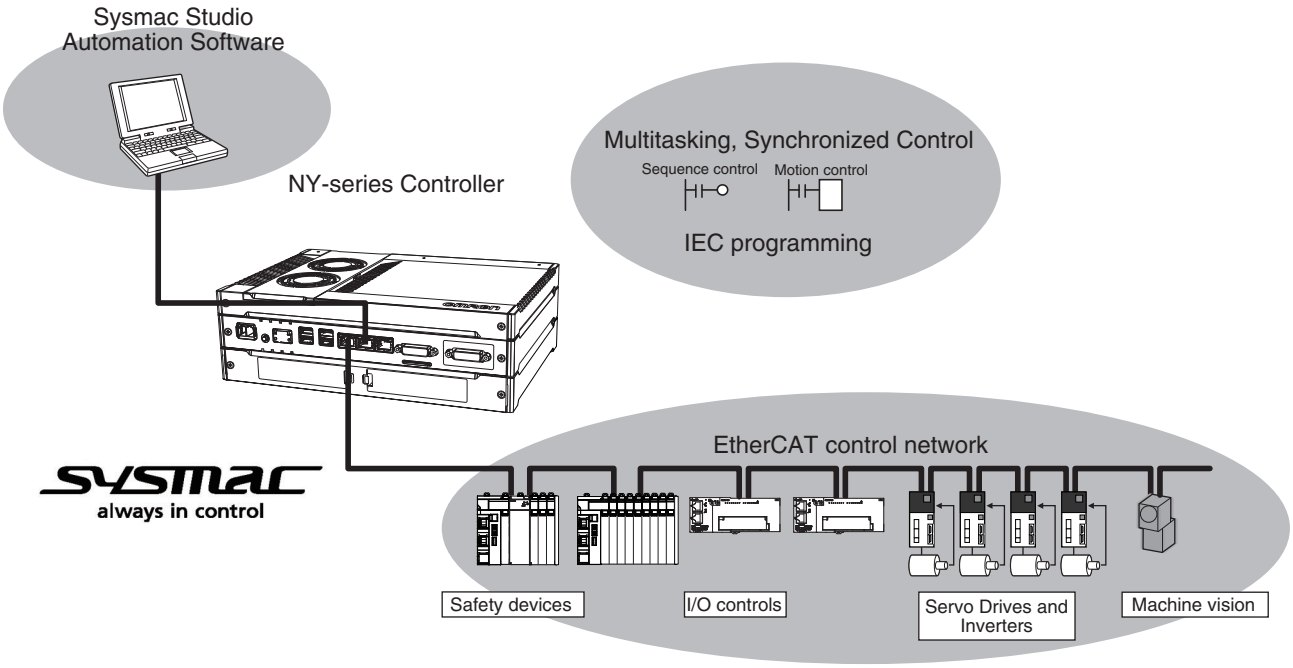
This section describes the features, basic system configuration, specifications, and overall operating procedure of an NY-series Controller.

1-1 The NY-series Controllers

The Industrial PC Platform NY-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 Industrial PC Platform NY-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 Industrial PC Platform NY-series 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

This section describes features of the Controller functions in the NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC.

Hardware Features

● Standard-feature EtherCAT Control Network Support

The NY-series Controllers 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.

● Standard-feature EtherNet/IP Communications Port

The NY-series Controllers 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.

● Highly Reliable Hardware

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

● Parallel Execution of Tasks with a Multi-core Processor

With a multi-core processor, the NY532-□□□□ and NY512-1□□□ can execute the task, the tag data link service, and system services in parallel. This enables high-speed control of even large-scale devices.

Software Features

Integrated Sequence Control and Motion Control

An NY-series Controller 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 NY-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 Controller 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 NY-series Controllers include operation authority settings and restriction of program execution with IDs.

Complete Controller Monitoring

The Controller monitors events in all parts of the Controller, including mounted EtherCAT slaves. Troubleshooting information for errors is displayed on the Sysmac Studio or on an NS/NA-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 and debugging on a virtual controller.

1-1-2 Introduction to the System Configurations

The NY Series supports the following system configurations.

● Basic System Configurations

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

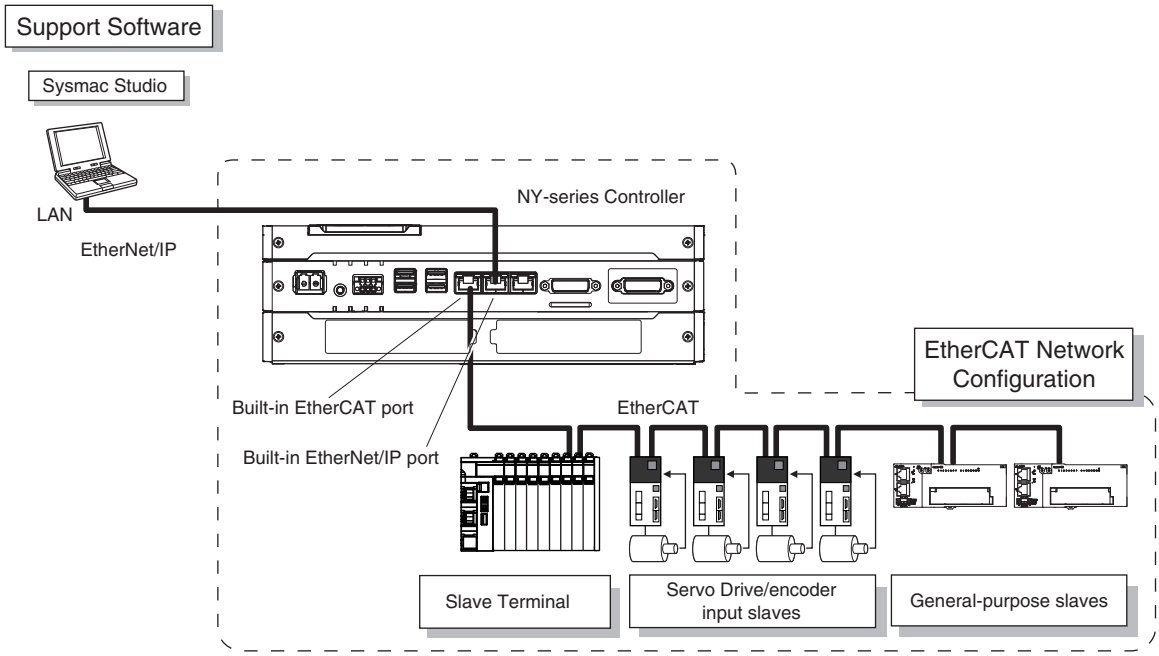
- EtherCAT Network Configuration

  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 built-in EtherNet/IP port with an Ethernet cable.

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

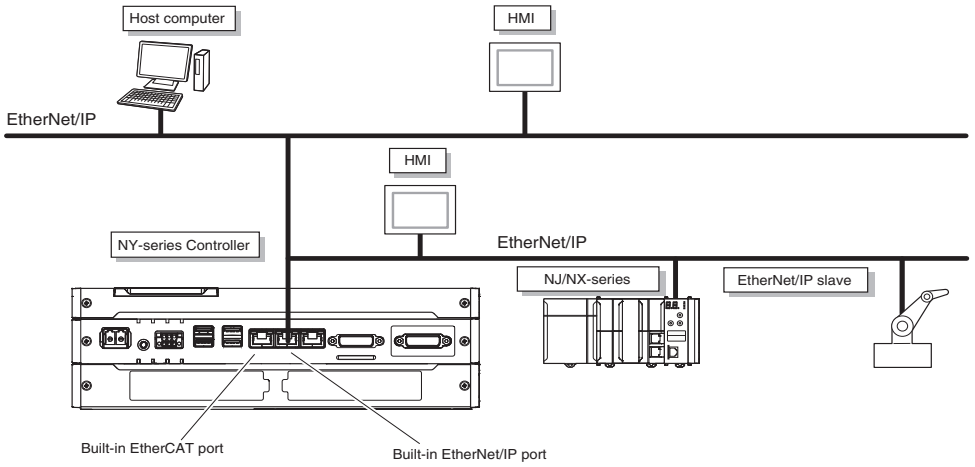


Additional Information

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/NX/NY-series Controllers are connected to the built-in EtherNet/IP port.



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 NY-series Controller functions.

- Sysmac Studio

  The Sysmac Studio is the main Support Software that you use for an NY-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-Designer	The CX-Designer is used to create screens for NS-series PTs.

*1 Use the Network Configurator if a CS/CJ-series PLC operates as the originator device.

Refer to the *NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat. No. W568) for information on Support Software other than the NY-series Controller functions.

1-2 Main Specifications

This section gives the main specifications of the NY-series Controller functions. Refer to *A-1 Specifications* for general specifications, performance specifications, and function specifications.

Item				NY5□2-15□□	NY5□2-□4□□	NY5□2-13□□
Programming	Program capacity*1	Size		40 MB		
		Quantity	Number of POU definitions	3,000		
			Number of POU instances	24,000		
	Memory capacity for variables	Retain attributes	Size	4 MB		
			Number of variables	40,000		
		No Retain attributes	Size	64 MB		
			Number of variables	180,000		
	Data types	Number of data types		4,000		
Motion control	Number of controlled axes	Maximum number of controlled axes*2		64 axes	32 axes	16 axes
		Maximum number of used real axes*3		64 axes	32 axes	16 axes
		Maximum number of axes for single-axis control		64 axes	32 axes	16 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			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		
		Maximum number of cam tables		640 tables		

Item			NY5□2-15□□	NY5□2-□4□□	NY5□2-13□□
Built-in EtherNet/IP port	Number of ports		1		
	Physical layer		10BASE-T, 100BASE-TX, or 1000BASE-T		
	Frame length		1,514 bytes max.		
	Media access method		CSMA/CD		
	Modulation		Baseband		
	Topology		Star		
	Baud rate		1 Gbps (1000BASE-T)		
	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	128		
		Packet interval*4	Can be set for each connection. 1 to 10,000 ms in 1-ms increments		
		Permissible communications band	20,000 pps*5 (including heartbeat)		
		Maximum number of tag sets	128		
		Tag types	Network variables		
		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		
		Maximum link data size per node (total size for all tags)	184,832 bytes		
		Maximum data size per connection	1,444 bytes		
		Maximum number of registrable tag sets	128 (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.)		
		Multi-cast packet filter*6	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		128		
Unit configuration	Maximum number of connectable Units	Maximum number of NX Units for entire controller	4,096 (On EtherCAT Slave Terminals)		

*1. This is the capacity for the execution objects and variable tables (including variable names).
*2. This is the total for all axis types.
*3. This is the total number of axes that are set as servo axes or encoder axes and are also set as used axes.
*4. Data will be refreshed at the set interval, regardless of the number of nodes.
*5. “pps” means packets per second, i.e., the number of communications packets that can be sent or received in one second.
*6. 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.

1-3 Product Model Numbers

You can identify the number of controlled axes and hardware configurations for the models of the NY-series Controllers.

NY 5 3 2 - 1 5 0 0 - 1 1 2 2 1 3 C 2 2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

No.	Item	Option
1	Series name	NY : NY-series Industrial PC Platform
2	Controller specifications	5 : Large scale, high speed and high precision control application for up to 64 axes.
3	Model type	1 : Industrial Box PC 3 : Industrial Panel PC
4	Sequential number	2 : Latest edition
5	Function module	1 : Standard
6	Number of axes for motion control	3 : 16 axes 4 : 32 axes 5 : 64 axes
7	Additional Function Software Module	0 : ---
8	Reserved.	0 : ---
9	Expansion slots	1 : 1 PCIe slot
10	Frame type	1 : Aluminum frame, black, and Projected Capacitive Touch type X : No display (Industrial Box PC)
11	Display size	1 : 12.1 inch 2 : 15.4 inch X : No display (Industrial Box PC)
12	OS	1 : Windows Embedded Standard 7 32bit 2 : Windows Embedded Standard 7 64bit
13	Processor	1 : Intel® Core™ i7-4700EQ Processor
14	Main memory	3 : 8 GB, non-ECC
15	Storage	8 : 32 GB, SSD SLC 9 : 64 GB, SSD SLC C : 320 GB, HDD K : 128GB, SSD iMLC
16	Optional interface	1 : RS-232C 2 : DVI-D
17	Logo	0 : OMRON 2 : Customization X : No display (Industrial Box PC)

Additional Information

This manual mainly describes the Controller functions, so that the model numbers given after No.9 may be omitted for which there are no influences to the Controller functions.

1-4 Overall Operating Procedure for the NY-series Controller

This section gives the overall operating procedure of the NY-series Controller functions and then describes it in more detail.

1-4-1 Overall Procedure

The overall procedure to use the NY-series Controller functions 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.

1-4-2 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	*NY-series Industrial Panel PC Hardware User's Manual* (Cat No. W557) *NY-series Industrial Box PC Hardware User's Manual* (Cat. No. W556)
⇩		
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 Creating the EtherCAT Slave 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*
	(To use motion control) Setting the initial settings for the Motion Control Function Module	**Configurations and Setup** – **Motion Control Setup**	*4-3 Initial Settings for the Motion Control Function Module*
	Setting the initial values for the EtherCAT Function Module	**Configurations and Setup** – **EtherCAT**	*4-4 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**	*4-5 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* *NY-series Instructions Reference Manual* (Cat. No. W560) and *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561)
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 on the Simulator (virtual controller)		*Section 7 Checking Operation and Actual Operation*

Step 3. Mounting and Setting Hardware

Step	Description	Reference
1. Mounting	• Mounting each part • Installation in a control panel	*NY-series Industrial Panel PC Hardware User's Manual* (Cat No. W557) *NY-series Industrial Box PC Hardware User's Manual* (Cat. No. W556)
2. Setting the Shared Folder	• Setting the shared folder used as a Virtual SD Memory Card in Windows.	*NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat No. W568)
3. Setting Hardware	• Setting the node addresses of the EtherCAT slaves	Operation manuals for the EtherCAT slaves

Step 4. Wiring

Step	Description	Reference
1. Connecting Ethernet Cable	• Connecting the built-in EtherCAT port • Connecting the built-in EtherNet/IP port	*NY-series Industrial Panel PC Hardware User's Manual* (Cat No. W557) *NY-series Industrial Box PC Hardware User's Manual* (Cat. No. W556)
2. Wiring I/O	• Wiring I/O to EtherCAT slaves	Operation manuals for the EtherCAT slaves *NY-series Industrial Panel PC Hardware User's Manual* (Cat No. W557) *NY-series Industrial Box PC Hardware User's Manual* (Cat. No. W556)
	• Checking wiring	*Sysmac Studio Version 1 Operation Manual* (Cat. No. W504)
3. Connecting the Computer That Runs the Sysmac Studio	• 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

NY-series Controller Operation

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

2-1 Overview of NY-series Controller Operation

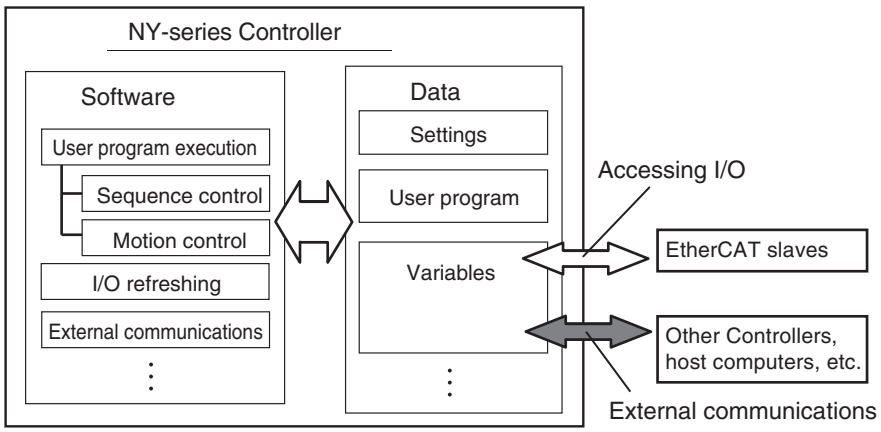
This section describes the operation of the Controller and gives an overview of how it operates depending on the status of the Controller.

2-1-1 Introduction to NY-series Controller

The NY-series Controller 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 Controller.

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

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



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

Item	Reference
Software	P.2-4
Accessing I/O	P.2-10
Sequence control and motion control	P.2-16
Overview of Controller data	P.2-21
Operation for Controller status	P.2-22

Additional Information

- Refer to the *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherNet/IP Port User's Manual* (Cat. No. W563) for details on the use of variables for external communications.
- Variables in the NY-series Controller can be accessed from Windows. Refer to the *NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat. No. W568) for information on accessing.

2-1-2 Overview of Operation According to NY-series Controller Status

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

The Controller operation according to the status of the Controller is described in *2-6 Operation for Controller Status*.

2-2 Software

This section describes the software configuration of the NY-series Controller functions, and how the software components operate.

2-2-1 Software Configuration

The software in the Controller 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, Virtual SD Memory Card services, and data trace processing.
Motion Control Function Module	Performs motion control processing.*1
EtherCAT Master Function Module	Performs communications with EtherCAT slaves as the EtherCAT master, including I/O refreshing*2 for the EtherCAT slaves, EtherCAT message communications*3, etc.
EtherNet/IP Function Module	Performs processing for EtherNet/IP communications, including tag data link processing, built-in EtherNet/IP port servicing, etc.

*1 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.

*2 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.

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

2-2-2 Operation of Software

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

Type of Controller processing	Status of Controller	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			Virtual SD Memory Card service

Refer to *2-6-1 Controller Status* for information on the Controller 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, and CPU Unit Watchdog Timer Error.*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 were not retained after a power interruption.
Recording Power Turned ON and Power Interrupted events	The Power Turned ON and Power Interrupted events are recorded.

*1 Refer to the *NY-series Troubleshooting Manual* (Cat. No. W564) for information on the following errors: Power Supply Error, CPU Unit Reset, and CPU Unit Watchdog Timer Error.

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 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.

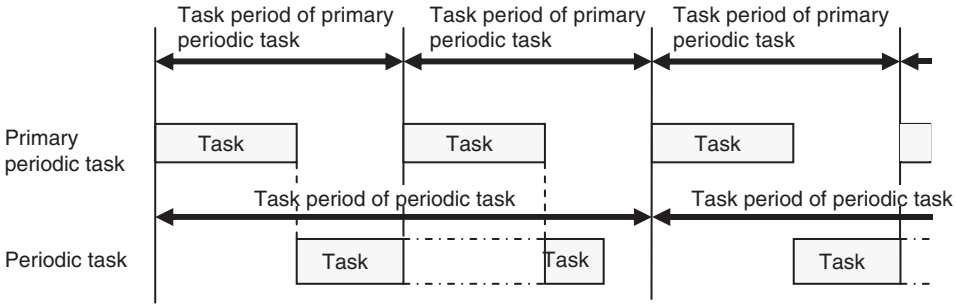
Refer to *5-3-3 Basic Operation of Tasks for NY-series Controllers* for details on the processing that is executed with tasks.

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	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, 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

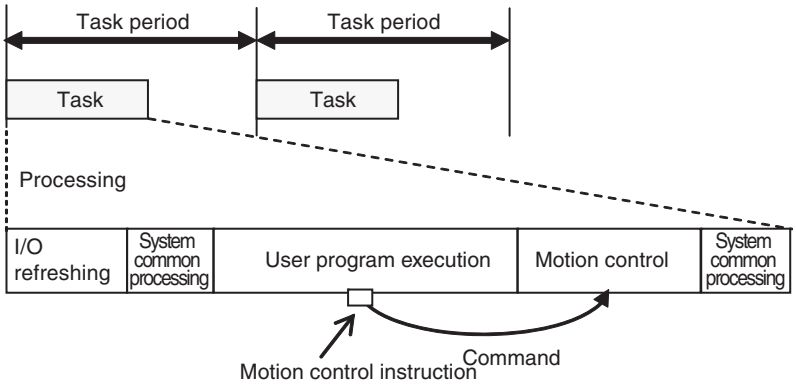
The Controller executes the task with the highest execution priority first. The following operation example is for the primary periodic task and a periodic task. If the primary periodic task is ready for execution while a periodic task is in execution, execution of the primary periodic task is prioritized.



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 Controller executes motion control in the primary periodic task. If the program that contains a motion control instruction is assigned to the periodic task, the execution process for the program in the periodic task will send a command to the motion control in the primary periodic task.
Refer to *5-9-3 System Input and Output Response Times* for details.

Tag Data Link Service

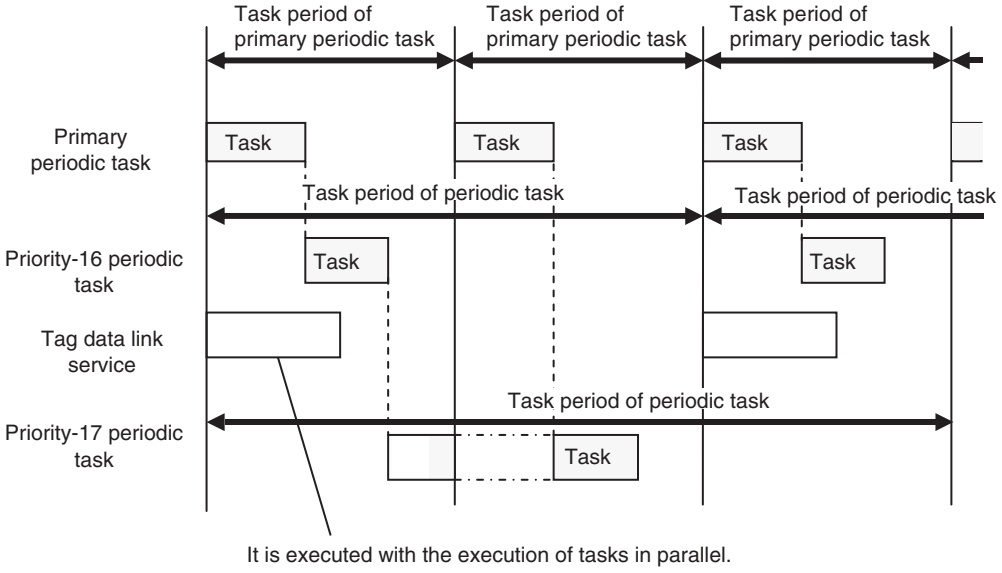
Processing Performed by the Tag Data Link Service

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 Controller to connect to an HMI.

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 Controller and on the tag data link settings.

The tag data link service is executed with the execution of tasks in parallel.



For details on the tag data link service, refer to *5-4 Tag Data Link Service and System Services*.

System Services

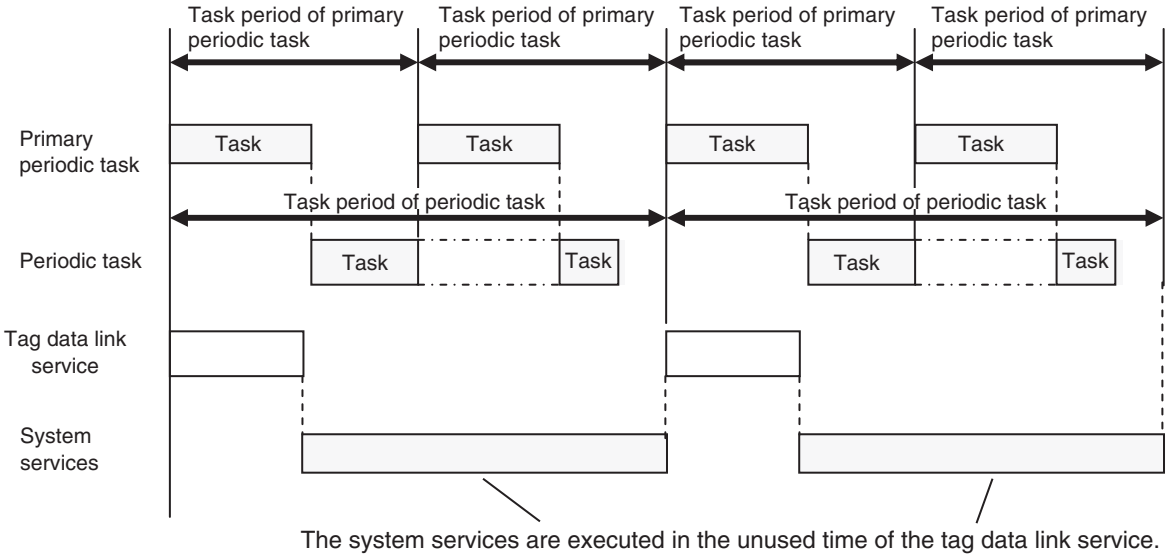
System Services

System services include the following processing.

Processing	Contents
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
Virtual 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 • Monitoring of Windows status

System Service Operations

For NY-series Controller, if a request comes from the hardware or from outside of the Controller, 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. However, the system services are not executed while the tag data link service is in progress.



Refer to *5-4 Tag Data Link Service and System Services* for details on the system services.

2-3 Accessing I/O

The NY-series Controller 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 are treated as I/O. Refer to the *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherNet/IP Port User's Manual* (Cat. No. W563) for details on how to access data in other Controllers with tag data links.

2-3-1 Types of Variables

In an NY-series Controller, you use variables in the user program to access I/O and memory in the Controller.

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>Controller</td></tr>
<tr><td rowspan="2">Semi-user-defined variables</td><td>Device variables</td><td>Device variables for EtherCAT slaves</td><td rowspan="2">Some attributes can be set.*1</td><td>EtherCAT slaves*2</td></tr>
<tr><td colspan="2">Cam data variables</td><td>Servo Drives, encoder input slaves, and Controller</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>Controller</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 Controller</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 3-7 for the attributes that can be set.

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

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

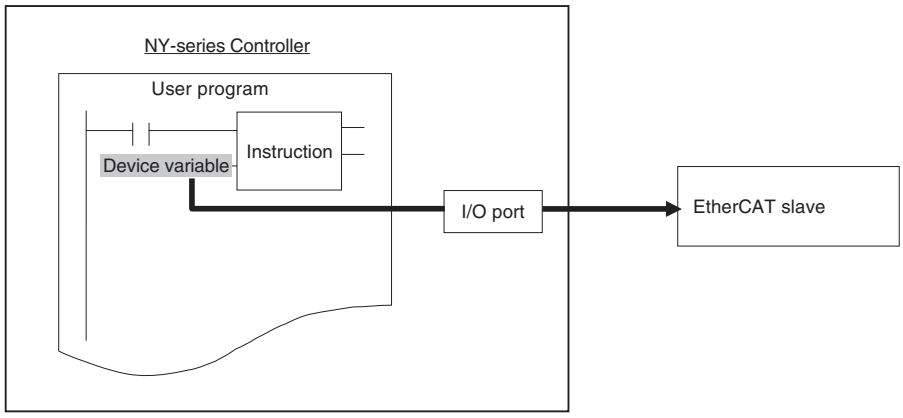
Device variables are used to access data in devices. A device is a general term for any Unit or slave that is refreshed by the I/O refreshing that is performed by the Controller. Specifically, it refers to EtherCAT slaves.

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	Process data for EtherCAT slaves*1

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

Device variables are used to access data for EtherCAT slaves 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 *NY-series Industrial Panel PC / Industrial Box PC Motion Control User's Manual* (Cat. No. W559).

System-defined Variables

System-defined variables are defined in advance in an NY-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

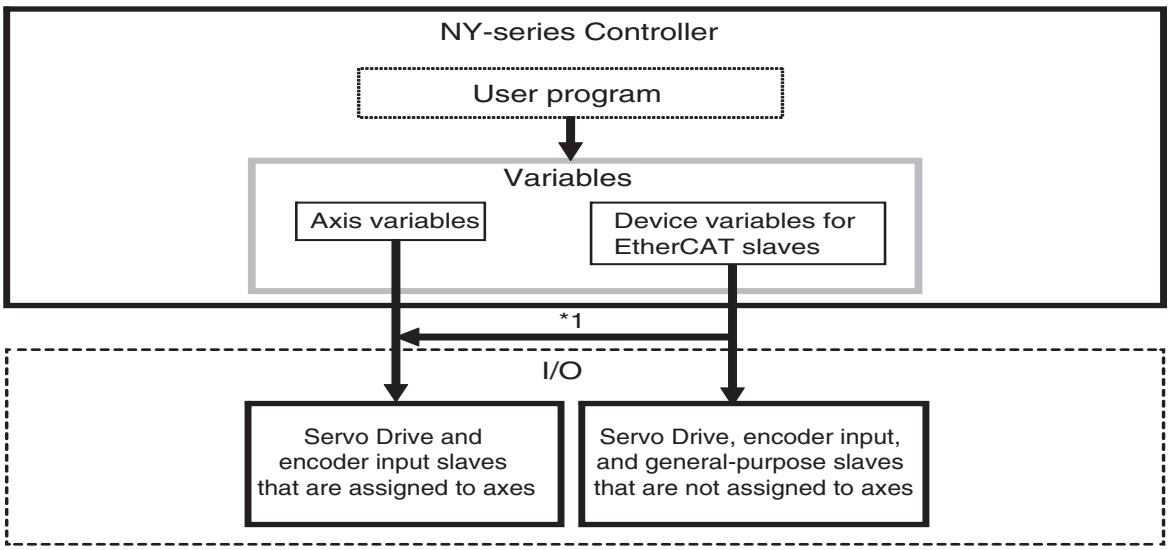
*1 An axes group consists of multiple axes. An axes group is used for interpolation.

Refer to *A-3 System-defined Variables* for details on system-defined variables.

2-3-2 Accessing I/O with Variables

In the NY-series Controller, 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 NY-series Controller.

I/O		Variables
EtherCAT slaves	EtherCAT slaves	Device variables for EtherCAT slaves
	EtherCAT slaves to which axes are assigned	Axis variables or device variables for EtherCAT slaves



Accessing EtherCAT Slaves

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

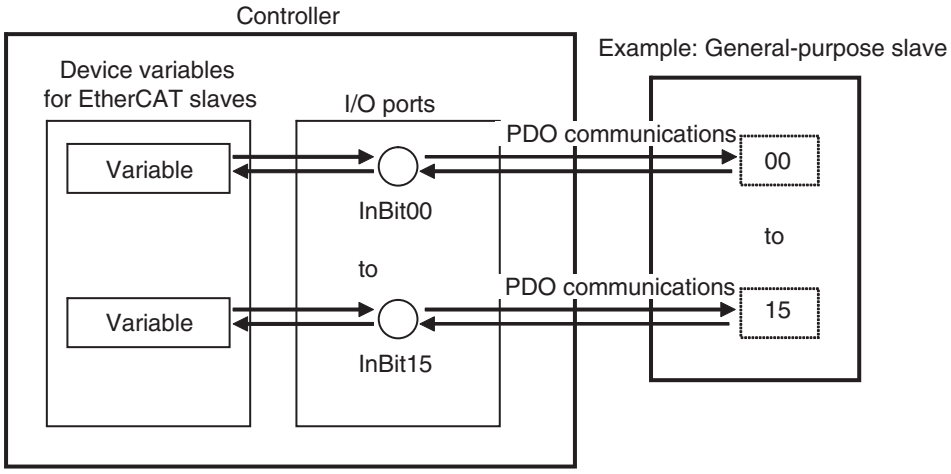
Type of EtherCAT slave	Access method
• Servo Drive and encoder input slaves • General-purpose slaves	These slaves are accessed through I/O ports by using device variables for EtherCAT slaves.
Servo Drive and encoder input slaves that are assigned to axes	These slaves are accessed directly with Axis variables.*1

*1 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 *NY-series Industrial Panel PC / Industrial Box PC Motion Control User's Manual* (Cat. No. W559) for details on Servo Drive and encoder input slaves.

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

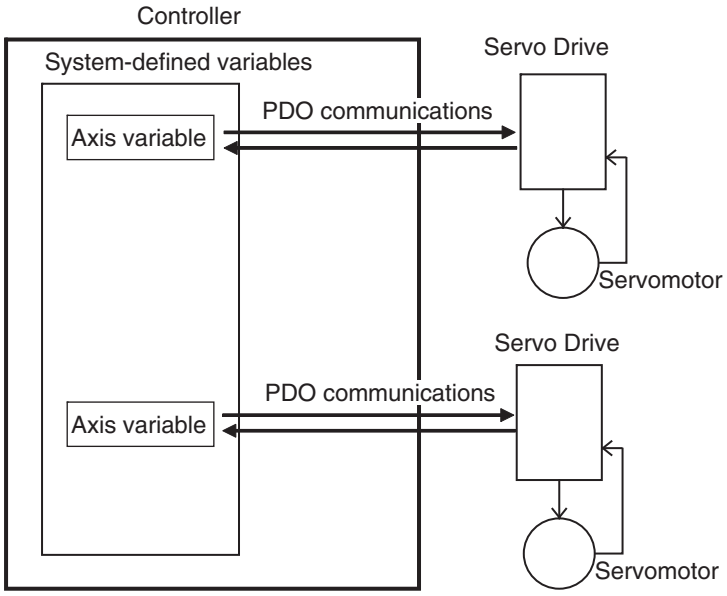
These slaves are accessed through I/O ports for device variables for EtherCAT slaves. PDO communications are used to access data from I/O ports.



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 Controller by using the Axis variable.



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

Precautions for Correct Use

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.

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 *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherCAT Port User's Manual* (Cat. No. W562) for details.

2-4 Sequence Control and Motion Control

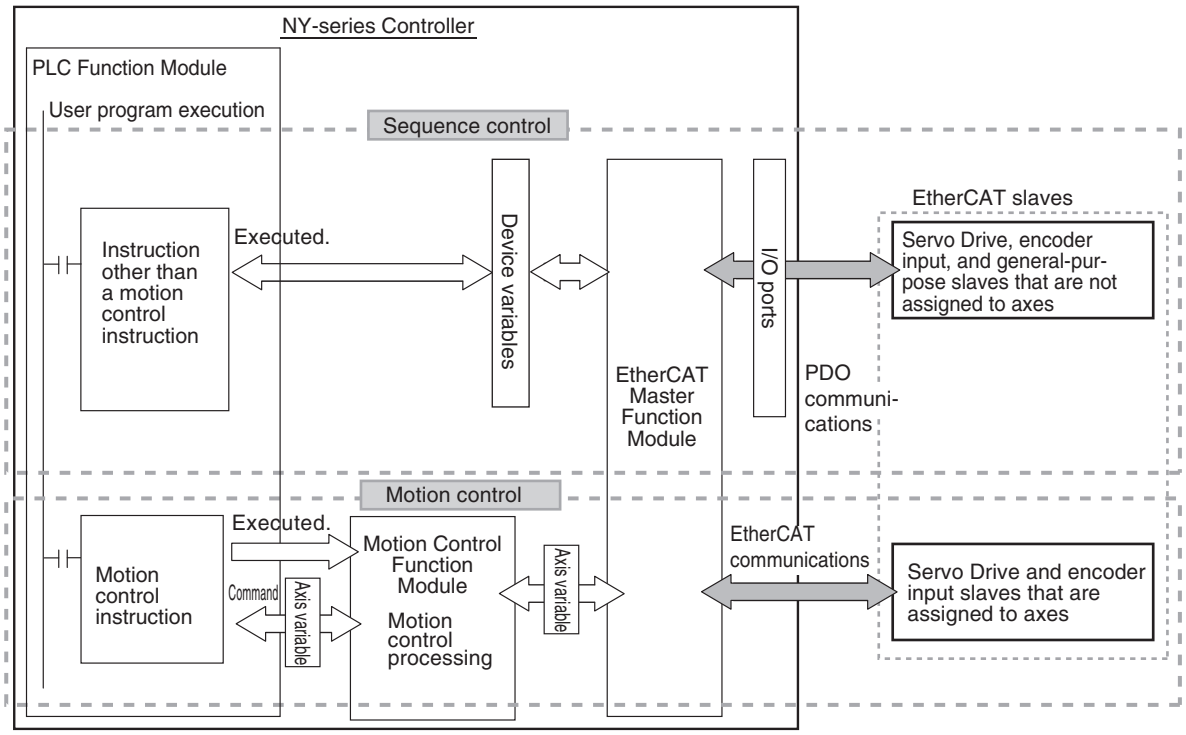
This section describes the sequence control and motion control systems that are used by the NY-series Controller.

2-4-1 Overview of Control

The NY-series Controller 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 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.



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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561). For details on other instructions, refer to the *NY-series Instructions Reference Manual* (Cat. No. W560).

Precautions for Correct Use

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.

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.

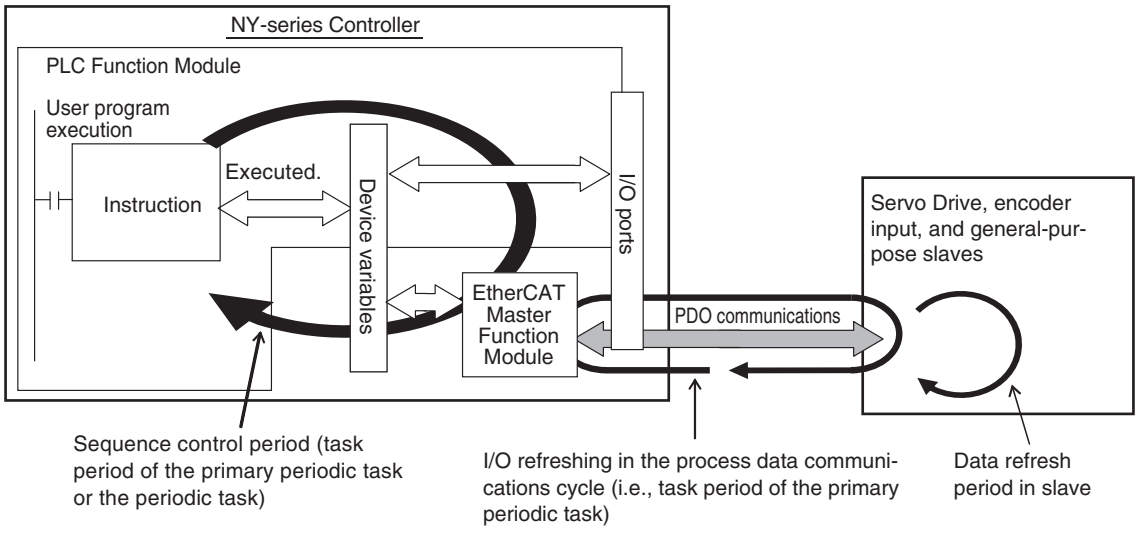
Device	Sequence Control System	
	Operation of the function module	Control period
Servo Drive, encoder input, and general-purpose slaves	• The PLC Function Module executes the user program and refreshes the device variables. • The EtherCAT Master Function Module exchanges data with the slaves through 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)*1

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

Servo Drive, encoder input, and general-purpose slaves 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. 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.

For details, refer to *5-9-3 System Input and Output Response Times.*



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

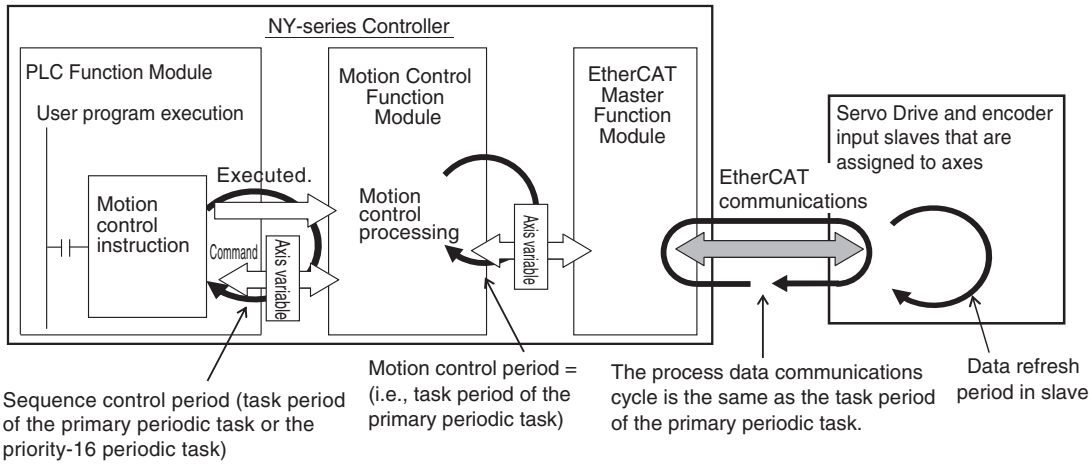
The motion control period is the task period of the primary periodic task.

Motion control processing in the Motion Control Function Module is executed in the task period of the primary 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. The process data communications cycle is synchronized with the primary periodic task.

This makes the motion control period the same as the task period of the primary 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.



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 *5-9-3 System Input and Output Response Times.*

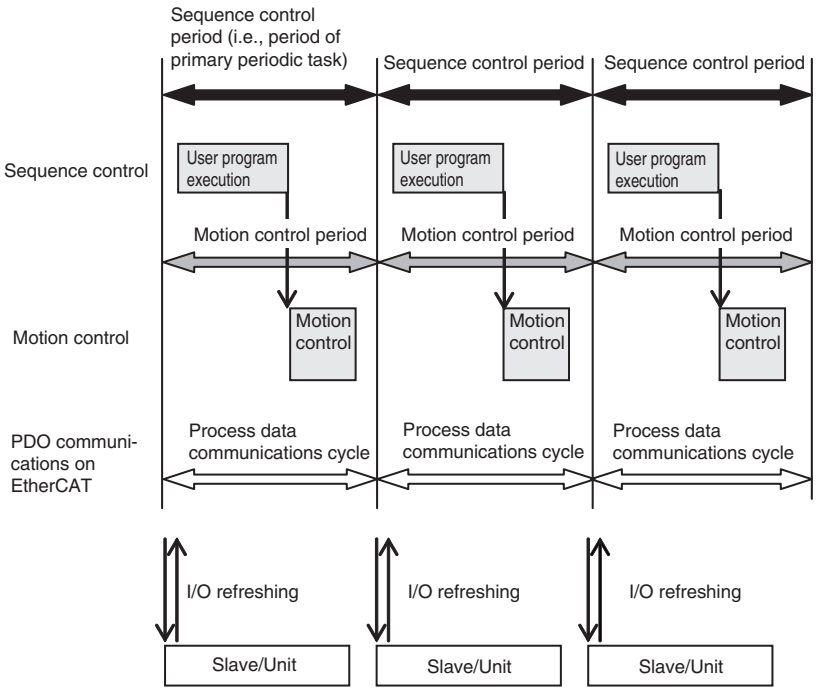
2-4-4 Synchronizing Sequence Control and Motion Control

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. The process data communications cycle for the EtherCAT slave to use for motion control is synchronized with the primary periodic task.

The NY-series Controller performs 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 primary periodic task, 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.



2-5 Overview of Controller Data

The NY-series Controller also contains settings, the user program, present values, and other data. The main data is described in the following table.

Refer to *A-5 Attributes of Controller Data* for information on all of the data in the NY-series Controller.

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 Controller 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, 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 Controller.
	Operation Authority Verification		This data contains the operation authority passwords to perform Sysmac Studio operations for the Controller.
	Built-in Clock	Set Time	This is the time information that is used inside the Controller.
		Time Zone Setting	This is the time zone that is set for the clock in the Controller.
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.
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.

2-6 Operation for Controller 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 NY-series Controller. It also describes the operating modes that change the execution status of the user program when the NY-series Controller is in the normal operation state.

2-6-1 Controller Status

The NY-series Controller 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 Controller in the normal operation state changes to the other states due to user interaction. This status is defined as follows:

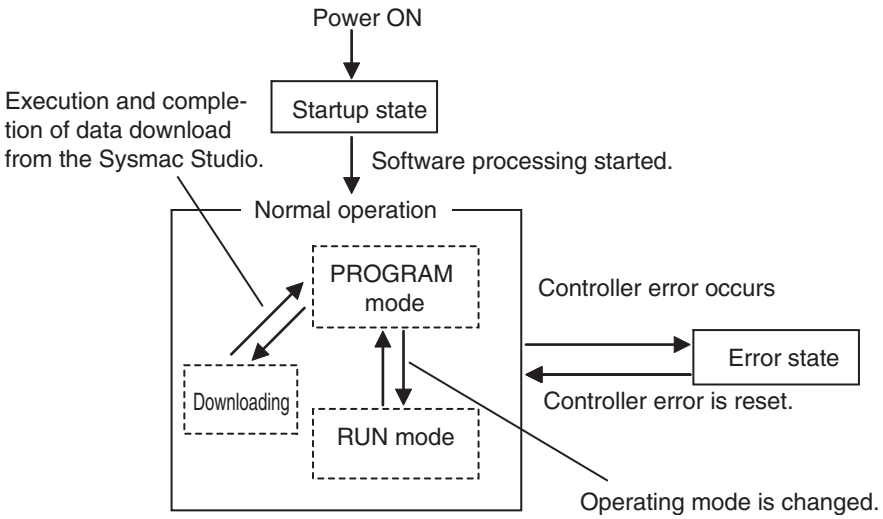
State	Definition
PROGRAM mode	The operating mode is PROGRAM mode.
RUN mode	The operating mode is RUN mode.
Downloading	Data is being downloaded from the Sysmac Studio.

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

Controller status

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

When the Controller 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 Controller to change between PROGRAM mode and RUN mode. If you download data from the Sysmac Studio during PROGRAM mode, the Controller will change to the downloading state. The Controller will return to PROGRAM mode when the download is completed.



Additional Information

- You can check the operating status of the Controller with the status indicators on the Industrial Panel PC or on the Industrial Box PC. Refer to the *NY-series Troubleshooting Manual* (Cat. No. W564) for troubleshooting procedures using the status indicators.
- Refer to *A-5 Attributes of Controller Data* for information on data operations when the Controller status changes.
- Refer to *6-3-9 Changes to Variables for Status Changes* for the values that variables take when the status of the Controller changes.

2-6-2 Operation for Controller Status

Changes in the status of the NY-series Controller 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 the *NY-series Troubleshooting Manual* (Cat. No. W564) for information on the error state.

Controller 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*2
External communications	Stopped.	Executed.		Executed. *3

*1 Only the input values are refreshed.

*2 I/O refreshing is executed when the device output hold configuration is set to enable (16#A5A5) in the *_DeviceOutHoldCfg* (Device Output Hold Configuration) system-defined variable. Refer to *Device Output Hold Configurations* on page 6-66 for the device output hold configurations. A Controller with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required to use the *_DeviceOutHoldCfg* (Device Output Hold Configuration) system-defined variable.

*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.		Controlled by the slave settings. *2 *3

*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 Device outputs are retained even when the operating mode changes or when downloading if the device output hold configuration is set to enable (16#A5A5) in the *_DeviceOutHoldCfg* (Device Output Hold Configuration) system-defined variables. Refer to *6-3-9 Changes to Variables for Status Changes* for details. A Controller with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required to use the *_DeviceOutHoldCfg* (Device Output Hold Configuration) system-defined variable.

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

Refer to *Device Output Hold Configurations* on page 6-66 for details on the device output hold configurations.

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 Controller 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 Controller will enter RUN mode when the Controller changes from startup state to normal operation.

Additional Information

The Controller 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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561).

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.

Operation		RUN mode	PROGRAM mode
Sysmac Studio	Synchronization	Not possible.	Possible.
	Online editing	Possible.	
	Forced refreshing	Possible.	
	Changing the values of variables	Possible.	
HMI	Changing the values of variables	Possible.	

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 6-61.

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

Setting and Changing the Operating Mode

When operation starts after the power supply is turned ON, the Controller 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 Controller starts operating after the power supply is turned ON, the Controller 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 Correct 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 Industrial PC or the Sysmac Studio.

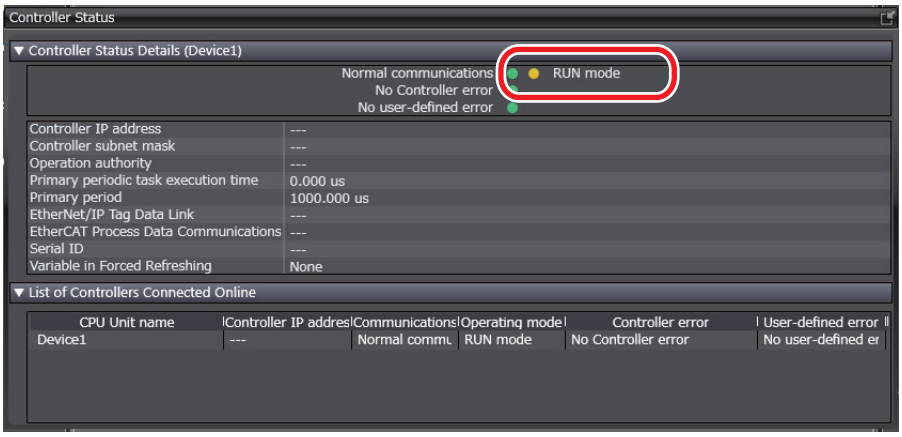
Checking the RUN Indicator

The RUN indicator on the Industrial PC 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 Controller is in RUN mode.



2-7 Monitoring and Changing Windows Status

In an NY-series Controller, you can monitor and change the status of Windows from the Controller.

2-7-1 Windows Status

The following table describes five states of Windows installed in an NY-series Controller.

State	Description
Booting	Windows is starting up.
Running	Windows is running.
ShuttingDown	Windows is shutting down.
Halted	Windows is stopped. (Processing to shut down Windows is completed and Windows is not running.)
Error	An error occurred in Windows.

Additional Information

You can use the system-defined variables or instructions to identify the status of Windows.

Running, *Halted*, and *Error* states can be identified. *Booting* and *ShuttingDown* states cannot be identified.

2-7-2 Monitoring Windows Status

You can use the system-defined variables or instructions to monitor Windows status.

System-defined Variables Related to Monitoring of Windows Status

Of the states of Windows, *Running*, *Halted*, and *Error* states are indicated with the system-defined variables.

Refer to *Functional Classification: OS (Windows)* on page A-61 for details on system-defined variables.

Variable name	Meaning	Function	Data type	R/W
_OSRunning	OS Running Flag	TRUE when the Controller observes that OS (Windows) is running.	BOOL	R
_OSHalted	OS Halted Flag	TRUE when the Controller observes that OS (Windows) is stopped.	BOOL	R
_OSErrorState	OS Error State Flag	TRUE when the Controller determines that an error occurred in OS (Windows).	BOOL	R

Instructions Related to Monitoring of Windows Status

The following instruction is supported to monitor Windows status.

Refer to the *NY-series Instructions Reference Manual* (Cat. No. W560) for details on the instruction.

Instruction	Instruction name	Introduction
IPC_GetOSStatus	Read OS Status	Reads OS (Windows) status. Of the states of Windows, *Running*, *Halted*, and *Error* states are read.

2-7-3 Changing Windows Status

You can use the instructions to change Windows status.

Instructions Related to Changing of Windows Status

The following instructions are supported to change Windows status.

Refer to the *NY-series Instructions Reference Manual* (Cat. No. W560) for details on the instructions.

Instruction	Instruction name	Introduction
IPC_RebootOS	Restart OS	Restarts OS (Windows).
IPC_Shutdown	Shut Down	Notifies the shutdown request to OS (Windows).

2-7-4 Changes in Windows Status

Windows status changes to *Booting* when an Industrial PC is started. Windows status changes to *Running* when the startup is ended.

If power is not provided due to the IPC_Shutdown instruction, a UPS power interruption detection, or other reasons, or the power supply button is pressed while Windows is running, Windows status changes to *ShuttingDown*. Windows status changes to *Halted* when the shutdown is ended.

Or, if the IPC_RebootOS instruction is executed while Windows is running, Windows status changes to *ShuttingDown*. Windows status changes to *Booting* when the shutdown is ended and changes to *Running* when the startup is ended.

Also execute the IPC_RebootOS instruction to recover from *Error* state.

If Windows status changes to *Running*, *Halted*, or *Error*, Controller events that indicate the changes are recorded.

However, if Windows status does not change to *Halted* even the time specified in the **OS** of the **Shutdown Wait Time Setting** is elapsed from the Controller notifies the shutdown request to Windows by executing the IPC_Shutdown instruction, a Controller event is not recorded.

2-8 Shutdown

You must connect a UPS (uninterruptible power supply) to an NY-series Industrial PC. By connecting a UPS, even if an unexpected lose of power (i.e., a power interruption) occurs, the shutdown can be properly performed to prevent data or file loss.

When the system-defined variable that indicates a power interruption detection in the UPS or a pressing of the power supply button changes to TRUE, the processing prepared for a power interruption can be executed in the user program of the Controller.

Program the following processing as required.

- Copy the variables without a Retain attribute that are used in the user program of the Controller to variables with a Retain attribute.
- Stop the file operations for the Virtual SD Memory Card in the Controller.
- Safely stop the operation of motion control during execution in the Controller.

After the processing in the user program is completed, the Controller system executes the shutdown such as a holding of variables with a Retain attribute and other processing. If the shutdown in the Controller is completed, the Controller notifies the shutdown request to Windows.

In Windows, the notification from the Controller is used as a trigger and the shutdown such as an application end and other processing can be executed.

Refer to *Details on Shutdown* on page 2-34 for details on shutdown.

Precautions for Correct Use

- If a UPS is not connected or the battery voltage of a UPS is dropped, the shutdown at the power interruption is not executed. Therefore, variables with a Retain attribute, absolute encoder home offsets, and event logs cannot be retained.
- Select a UPS with a battery capacity that the backup time is longer than the shutdown time of the Controller and Windows. The shutdown is not normally ended if the battery backup time is short.

Additional Information

Refer to the *NY-series Industrial Panel PC Hardware User's Manual* (Cat. No. W557) or the *NY-series Industrial Box PC Hardware User's Manual* (Cat. No. W556) for how to connect a UPS.

2-8-1 Processing When Power Is Interrupted or Power Supply Button Is Pressed

This section describes the processing of the Controller when power is interrupted, the power supply button is pressed, or the IPC_Shutdown instruction is executed while the Controller is operating. The processing is different according to the UPS connection or the time to press the power supply button.

Operation	UPS connection	Controller processing
Power is interrupted.	A UPS is connected.	The signal from a UPS is used as a trigger and the shutdown is executed. *1 *2
	A UPS is not connected or the UPS battery voltage is dropped.	The shutdown is not executed. *3
The power supply button is pressed within 5 seconds.	---	A pressing of the power supply button is used as a trigger and the shutdown is executed. *1 *2
The IPC_Shutdown instruction is executed.	---	The Controller system executes the shutdown. *1
The power supply button is pressed in 5 seconds or more.	---	A forced shutdown is started. The shutdown is not executed. *3

*1 Refer to *2-8-3 Contents of Shutdown* for information on shutdown.

*2 The Controller system executes the shutdown by executing the IPC_Shutdown instruction or when the time that is specified in the **User program** of the **Shutdown Wait Time Setting** is elapsed.

*3 If the shutdown is not executed and the power supply is turned ON next time, an Event Log Save Error observation event or a Present Values of Retained Variables Not Saved major fault event will occur.
Also, the following will happen to data in the Controller.
Values of variables with a Retain attribute and absolute encoder home offsets will be the values that the power supply is turned ON previously.
Event logs from the previous startup until the shutdown is not executed are disappeared.

Additional Information

If you press the power supply button when the Controller is not operated, the normal mode of an Industrial PC is started. Press the power supply button in a disconnection status of the power connector, connect the power connector while pressing the power supply button, and 10 seconds is elapsed, the Controller starts operation in PROGRAM mode no matter the Startup Mode setting in the Controller Setup.
In this case, a Safe Mode observation event will occur.
Refer to the *NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat. No. W568) for information on startup and end.

2-8-2 Settings of Shutdown

Set the following parameters in the Controller Setup.

Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
Shutdown Wait Time Setting	User program	Sets the time to wait when the signal is ON from a UPS which indicates a power interruption detection, or when a shutdown command is accepted by pressing the power supply button until the Controller starts the shutdown.	1 to 30 s	5 s	When downloaded to Controller	Not allowed.
	OS	Sets the time to wait when the Controller notifies the shutdown request to Windows until the shutdown completion is detected in Windows.	30 to 300 s	60 s	When downloaded to Controller	Not allowed.

● User Program

The signal from a UPS which indicates a power interruption detection is monitored in the user program of the Controller. The signal from a UPS is used as a trigger and the processing prepared for a power interruption is executed in the user program.

The Controller system waits the completion of processing in the user program only for the time that is specified in the **User program** setting. Therefore, specify a time in the **User program** setting which is longer than the processing time in the user program. If the specified time is short, the processing in the user program will be interrupted.

● OS

If the shutdown in the Controller system is completed, the Controller system notifies the shutdown request to Windows. In Windows, the notification from the Controller is used as a trigger and the shutdown such as an application end and other processing can be executed.

According to the Controller system monitors the status in Windows, the normal shutdown completion in Windows can be detected.

When the shutdown in Windows is completed before the time specified in the **OS** setting, a Controller event that indicates the state is changed to *Halted* is recorded. Therefore, specify a time in the **OS** setting which is longer than the processing time in Windows. If the specified time is short, the processing in Windows will be interrupted and applications in Windows cannot be normally ended.

If a Controller event that indicates the state is changed to *Halted* is not recorded, the time specified in the **OS** setting may be shorter than the processing time in Windows.

2-8-3 Contents of Shutdown

When a UPS power interruption is detected or the power supply button is pressed, the system-defined variable which corresponds to each status changes to TRUE. By monitoring the status of these system-defined variables, it is possible to copy the necessary data to variables with a Retain attribute and safely stop the operation of motion control during execution. A monitoring of the system-defined variables and processing prepared for a power interruption are created in the user program of Controller.

Execute the IPC_Shutdown instruction after the completion of processing prepared for a power interruption in the user program.

The Controller system performs the shutdown by executing the IPC_Shutdown instruction.

System-defined Variables That Are Used As Triggers of Shutdown

The following table gives the system-defined variables that are used as triggers to start the processing in the user program.

Variable name	Meaning	Function	Data type	R/W
_SelfTest_UPSSignal	UPS Signal Detection Flag	TRUE when a temporary power interruption signal from UPS is detected.	BOOL	R
_RequestShutdown	Request Shutdown Flag	TRUE when the power supply button is pressed while running.	BOOL	R

Refer to *Functional Classification: Power Supply* on page A-60 for details on system-defined variables.

Instructions That Notify Shutdown Requests

The following table gives an instruction that is used to notify the processing completion in the user program to the Controller system and notify the shutdown request from the Controller system to Windows.

Instruction	Instruction name	Introduction
IPC_Shutdown	Shut Down	Starts the shutdown in the Controller system and notifies the shutdown request to Windows after the shutdown is completed.

Refer to the *NY-series Instructions Reference Manual* (Cat. No. W560) for details on the instruction.

Details on Shutdown

The following describes the details on the shutdown in any of three states of the Controller: RUN mode, PROGRAM mode, or startup state.

● RUN Mode

The user program in the Controller is executed.

In RUN mode, a monitoring of the system-defined variables and processing prepared for a power interruption created in the user program are executed to prevent loss of operation data in the equipment managed in the Controller. The Controller system performs the shutdown after the processing in the user program is completed. Then, the shutdown such as an application end and other processing is executed in Windows.

The following describes the details of shutdown in RUN mode.

1 Monitor the *_SelfTest_UPSSignal* (UPS Signal Detection Flag) and *_RequestShutdown* (Request Shutdown Flag) system-defined variables in the user program.

2 When the *_SelfTest_UPSSignal* (UPS Signal Detection Flag) or *_RequestShutdown* (Request Shutdown Flag) system-defined variable changes to TRUE, execute the processing prepared for a power interruption in the user program as required.

3 Execute the IPC_Shutdown instruction in the user program after the completion of processing prepared for a power interruption.

- The Controller system starts the shutdown when the IPC_Shutdown instruction is executed.
- Values of variables with a Retain attribute, absolute encoder home offsets, and event logs are retained.
- The Controller notifies the shutdown request to Windows and Windows starts the shutdown.
- Even if the IPC_Shutdown instruction is not executed, the Controller system starts the shutdown when the time that is specified in the **User program** of the **Shutdown Wait Time Setting** is elapsed.
- When the shutdown in Windows is completed, a Controller event that indicates Windows status changes to *Halted* is recorded.
  However, if Windows status does not change to *Halted* even the time specified in the **OS** of the **Shutdown Wait Time Setting** is elapsed from the Controller notifies the shutdown request to Windows, a Controller event is not recorded.
- If the signal from a UPS is used as a trigger and the shutdown is started, the Power Status Output of the I/O connector on the Industrial PC turns ON after the shutdown is completed.
- The UPS stops providing power when the Power Status Output turns ON.

● PROGRAM Mode

The user program in the Controller is not executed.

In PROGRAM mode, the processing prepared for a power interruption in the user program is not necessary because the Controller does not perform control operations in the equipment.

The shutdown such as an application end and other processing is executed in Windows after the Controller system completes the shutdown.

The following describes the details of shutdown in PROGRAM mode.

- When the *_SelfTest_UPSSignal* (UPS Signal Detection Flag) or *_RequestShutdown* (Request Shutdown Flag) system-defined variable changes to TRUE, the Controller system immediately starts the shutdown.
- Values of variables with a Retain attribute, absolute encoder home offsets, and event logs are retained.
- The Controller notifies the shutdown request to Windows and Windows starts the shutdown.
- When the shutdown in Windows is completed, a Controller event that indicates Windows status changes to *Halted* is recorded.
  However, if Windows status does not change to *Halted* even the time specified in the **OS** of the **Shutdown Wait Time Setting** is elapsed from the Controller notifies the shutdown request to Windows, a Controller event is not recorded.
- If the signal from a UPS is used as a trigger and the shutdown is started, the Power Status Output of the I/O connector on the Industrial PC turns ON after the shutdown is completed.
- The UPS stops providing power when the Power Status Output turns ON.

● Startup State

The Controller is initializing the system.

In startup state, the processing prepared for a power interruption in the user program is not necessary because the Controller does not perform control operations in the equipment.

The shutdown such as an application end and other processing is executed in Windows after the Controller system completes the shutdown.

The following describes the details of shutdown in startup state.

- When the *_SelfTest_UPSSignal* (UPS Signal Detection Flag) or *_RequestShutdown* (Request Shutdown Flag) system-defined variable changes to TRUE, the Controller system immediately starts the shutdown.
- Values of variables with a Retain attribute, absolute encoder home offsets, and event logs are retained.
- The Controller notifies the shutdown request to Windows and Windows starts the shutdown.
- When the shutdown in Windows is completed, a Controller event that indicates Windows status changes to *Halted* is recorded.
  However, if Windows status does not change to *Halted* even the time specified in the **OS** of the **Shutdown Wait Time Setting** is elapsed from the Controller notifies the shutdown request to Windows, a Controller event is not recorded.
- If the signal from a UPS is used as a trigger and the shutdown is started, the Power Status Output of the I/O connector on the Industrial PC turns ON after the shutdown is completed.
- The UPS stops providing power when the Power Status Output turns ON.

2-9 Resetting the Controller

The Controller can be reset only by resetting the Controller from the Sysmac Studio or Industrial PC Support Utility.

● Resetting the Controller from the Sysmac Studio

The Controller can be reset when the operating mode of the Controller is PROGRAM mode.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for information on resetting the Controller from the Sysmac Studio.

● Resetting the Controller from the Industrial PC Support Utility

The Controller can be reset when the operating mode of the Controller is PROGRAM mode.

Refer to the *NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat. No. W568) for information on resetting the Controller from the Industrial PC Support Utility.

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

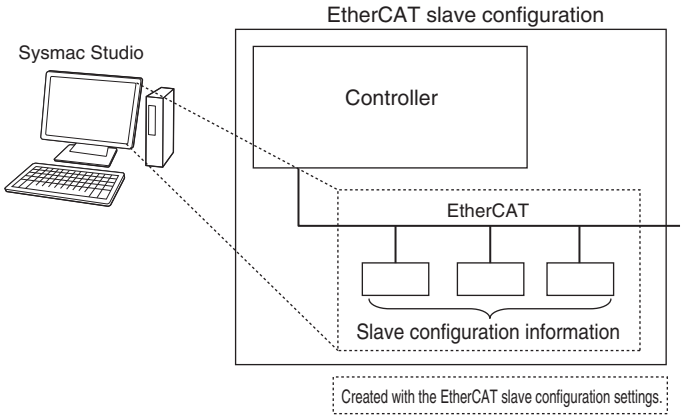
- Step 2-1 Slave and Unit Configurations
- Step 2-2 Controller Setup
- Step 2-3 Programming
- Step 2-4 Offline Debugging

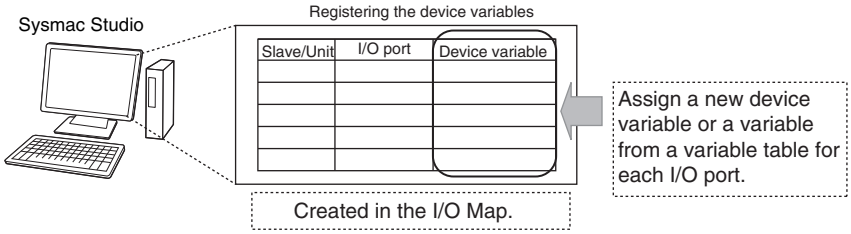
Step 3. Mounting and Setting Hardware

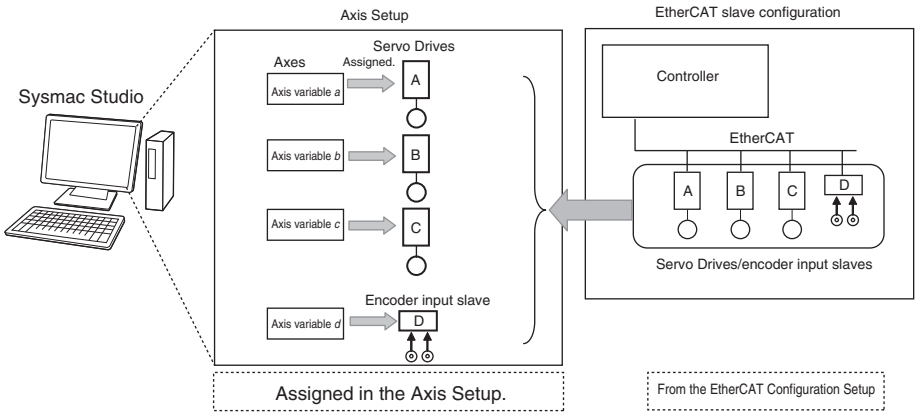
Step 4. Wiring

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

For details, refer to *1-4 Overall Operating Procedure for the NY-series Controller*.

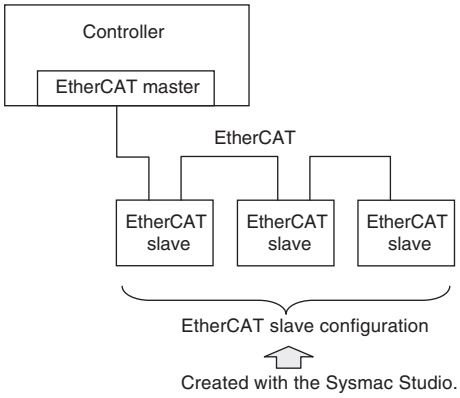
Step 1 Create the EtherCAT Slave Configuration (if EtherCAT is used).	Reference
• Create the EtherCAT Slave Configuration. 	*3-2 Creating the EtherCAT Slave Configuration*

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 the EtherCAT Slave Configuration

To enable accessing the slaves that are connected to the NY-series Controller, you create a slave configuration on the Sysmac Studio. In the EtherCAT Tab Page of the Sysmac Studio, create the EtherCAT slave configuration that is detected as “correct” by the NY-series Controller.



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.

Additional Information

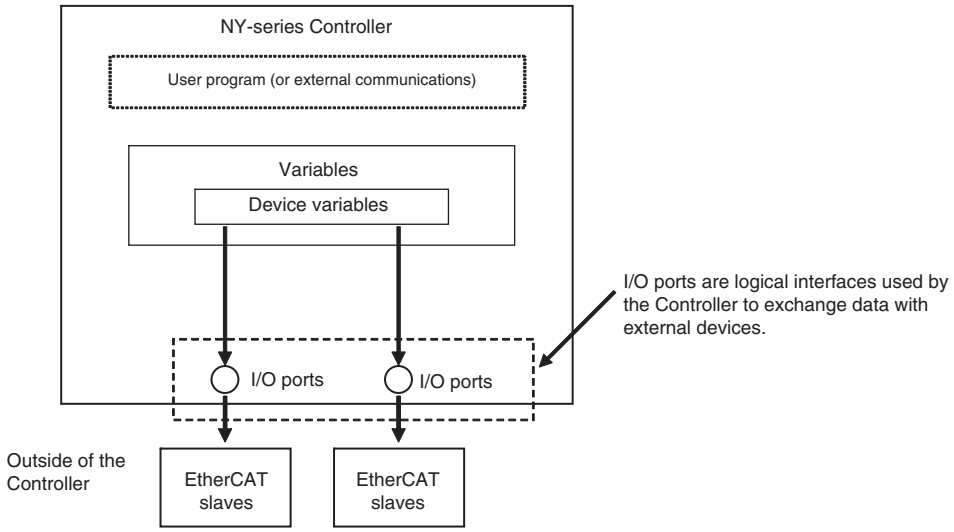
If the EtherCAT slaves are Servo Drives or encoder input slaves, after they are registered in the EtherCAT slave configuration, Axis Variables are registered automatically by creating the axes. Refer to *3-5 Creating the Axes and Assigning Them to the Servo Drives/Encoder Input Slaves* for details.

3-3 I/O Ports and Device Variables

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

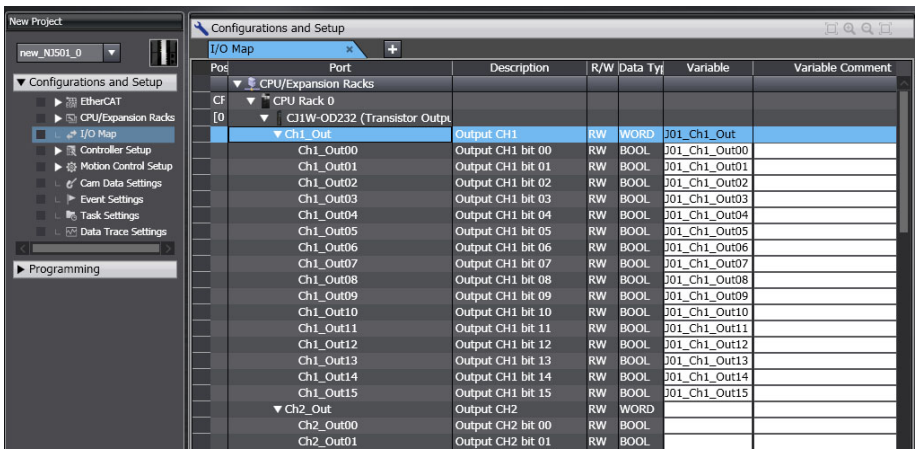
3-3-1 I/O Ports

An I/O port is a logical interface that is used by the NY-series Controller 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 following I/O port name is used when the device (i.e., Unit or slave) is an EtherCAT slave.

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

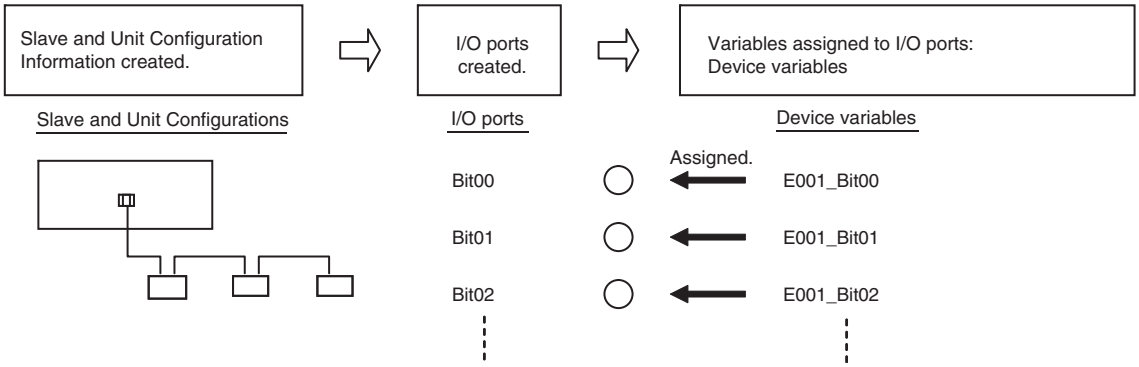
For other slaves, all or part of the object names that are defined in the EtherCAT object dictionary are used.

Example for Analog Input Unit: CH0_input16-bit

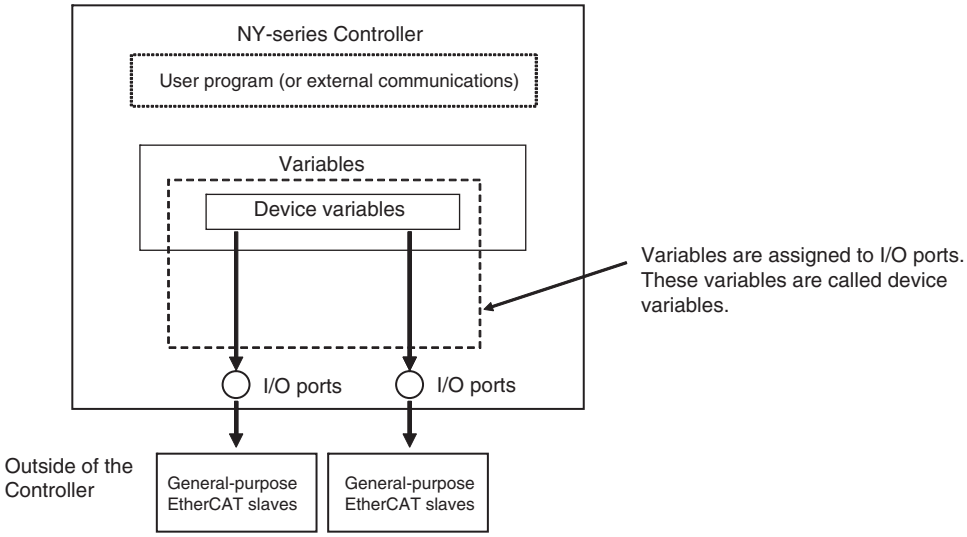
Examples for the R88D-KN50H-ECT: Position actual value and Digital inputs

3-3-3 Device Variables

In an NY-series Controller, external devices (slaves and Units) are not assigned to specific memory addresses in the Controller. 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*] For the default device names of EtherCAT slaves, an E followed by a sequential number starting from 001. 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]*	Not allowed.
Retain	Device variables for EtherCAT slaves: Not retained.	Not allowed.
Initial Value	None	Allowed.
Constant	None	Allowed.
Network Publish	Do not publish.	Allowed.
Edge	None	Not allowed.

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.

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

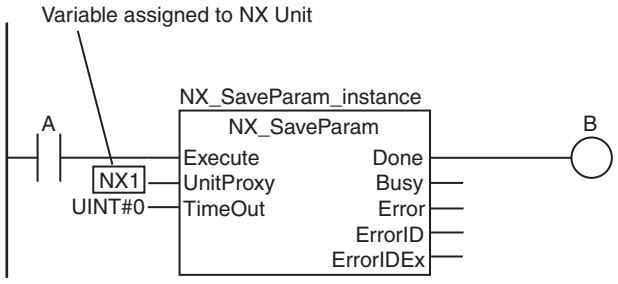
This section describes how to use the variables assigned to the Units in the user program. In any cases, the variable with the same name as the variable assigned to the Unit on the I/O Map must be registered in the variable table in advance. The data type of the variable is _sNXUNIT_ID structure.

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

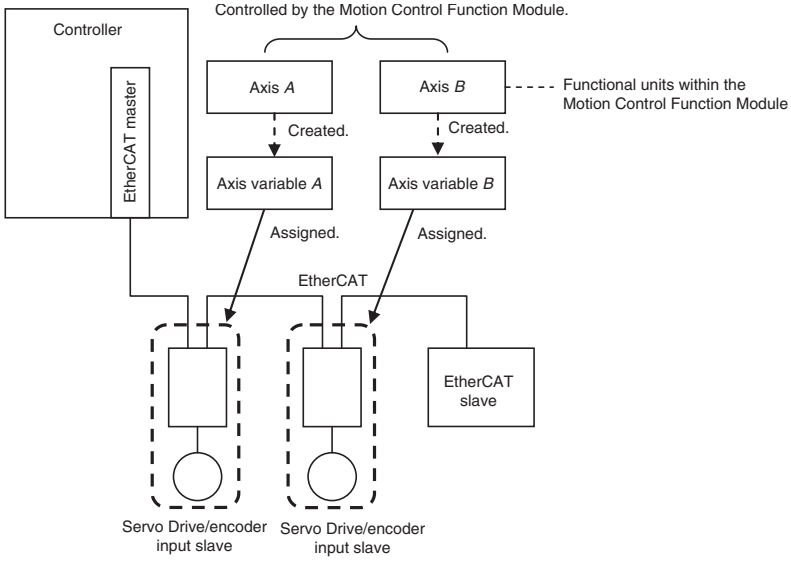
```
FOR i:= 0 TO 2 DO
  NX_ChangeWriteMode_instance[i](Execute:=FALSE);
END_FOR;

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 NY-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 drives or encoder input slaves. 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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561) 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[***] (The array element numbers are assigned in ascending order from 0 in the order the variables are created.)	_MC_GRP[***] (The array element numbers are assigned in ascending order from 0 in the order the variables are created.)

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.

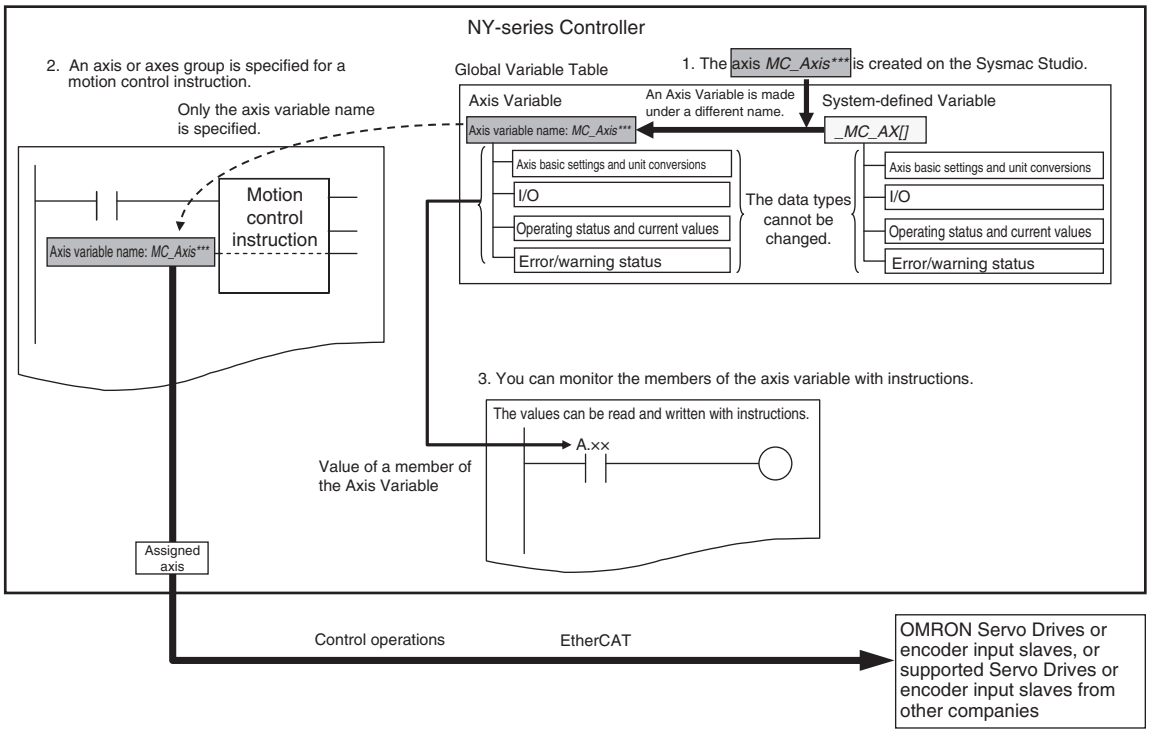
Additional Information

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.

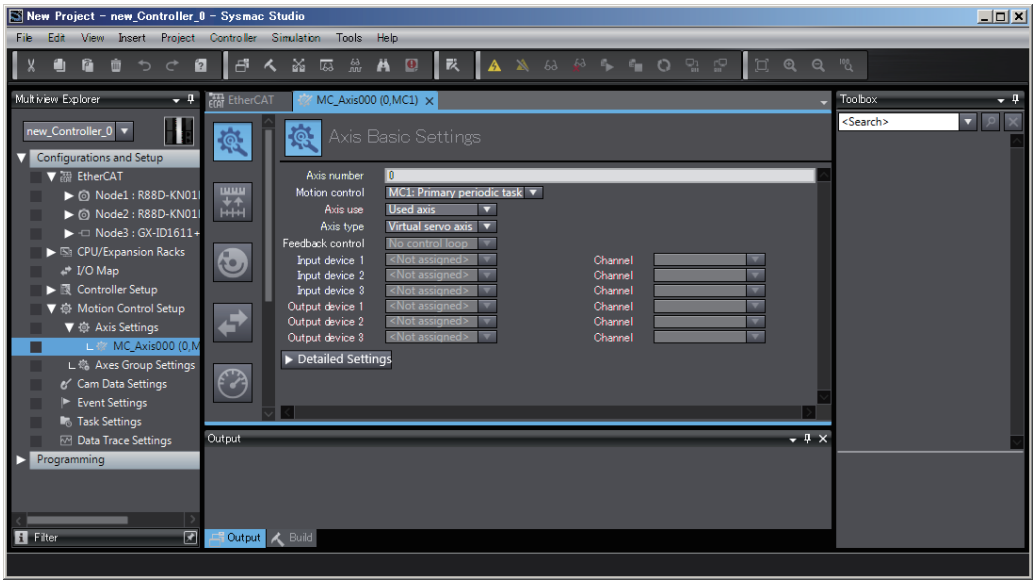
1 Right-click **Axis Settings** under **Configurations and Setup** – **Motion Control Setup** in the Multiview Explorer and select **Add** – **Axis Settings** from the menu.

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.
	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.

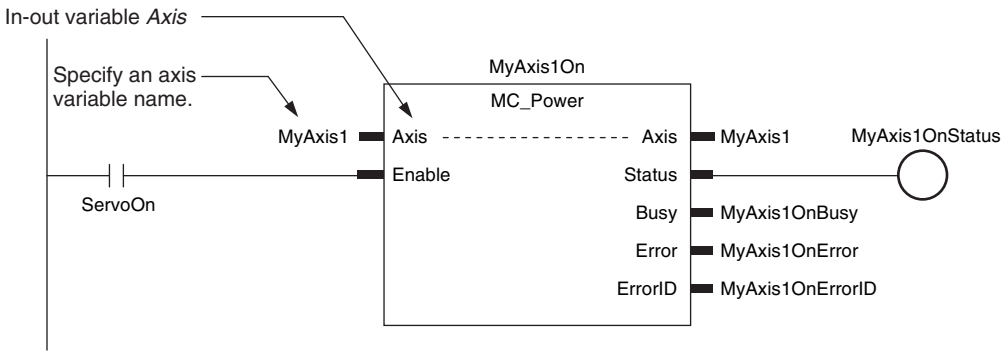


3 Use the Sysmac Studio to specify the settings required for Test Mode operation (Unit Conversion, Count Mode, Limits, etc.) and the settings required for actual system operation. Then transfer the settings to the Controller with the project.

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.

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 *1-4 Overall Operating Procedure for the NY-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 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	*4-3 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	*4-4 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	*4-5 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

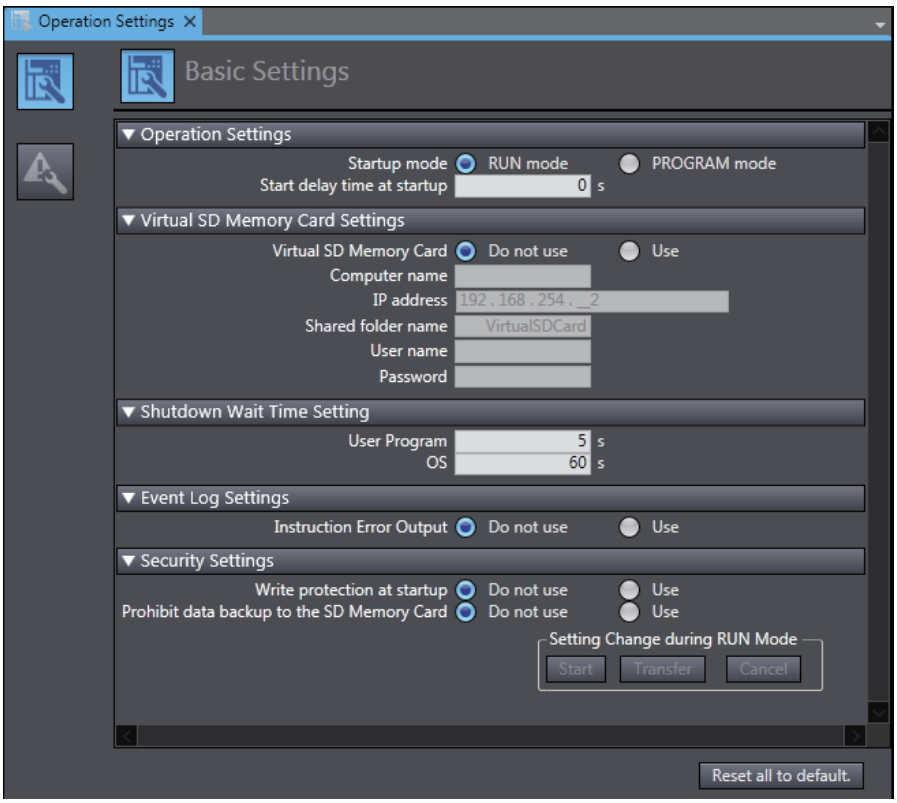
Select **Configurations and Setup** – **Controller Setup** and **Configurations and Setup** – **Task Settings** on the Sysmac Studio to make these settings.

4-2-2 Controller Setup

Operation Settings Tab Page

● Basic Settings

The Operation Settings are for functions supported by the Controller, 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 Controller	Not allowed.
	Start delay 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 Controller	Not allowed.
Virtual SD Memory Card Settings	Virtual SD Memory Card	Sets whether to use the shared folder as a Virtual SD Memory Card.	Use. Do not use.	Disabled.	When downloaded to Controller	Not allowed.
	Computer name	Sets the computer name on Windows.	Text string*3	None	When downloaded to Controller	Not allowed.
	IP address	Sets the IP address of an internal port in Windows.	IPv4*4	192.168.254.2	When downloaded to Controller	Not allowed.
	Shared folder name	Sets a name of the shared folder that is used as a Virtual SD Memory Card.	Text string*5	VirtualSDCard	When downloaded to Controller	Not allowed.
	User name	Sets a shared user name of the shared folder that is used as a Virtual SD Memory Card.*6	Text string*7	None	When downloaded to Controller	Not allowed.
	Password	Sets a password of the shared user that is used as a Virtual SD Memory Card.	Text string*8	None	When downloaded to Controller	Not allowed.
Shutdown Wait Time Setting	User program	Sets the time to wait when the signal is ON from a UPS which indicates a power interruption detection, or when a shutdown command is accepted by pressing the power supply button until the Controller starts the shutdown.	1 to 30 s	5 s	When downloaded to Controller	Not allowed.
	OS	Sets the time to wait when the Controller sends the shutdown command to Windows until the shutdown completion is detected in Windows.	30 to 300 s	60 s	When downloaded to Controller	Not allowed.
Event Log Settings	Instruction Error Output	Sets whether to output events to an event log when instruction errors occur.	Do not use. Use.	Do not use.	When downloaded to Controller	Not allowed.

Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
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	Allowed.
	Prohibit data backup to the SD Memory Card*9	Sets whether to enable or disable SD Memory Card backups.	Do not use. Use.	Do not use.	When downloaded to Controller	Allowed.

*1 This setting is enabled when an NS-series PT is connected to the built-in EtherNet/IP port on the Controller, and the power supplies for these devices are turned ON simultaneously.

*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 Controller gives priority to the tag verifications of the NS-series PT for approximately 10 seconds during startup before the Controller 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 Controller changes the state to the normal operation state is increased because the Controller 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 Controller simultaneously.

*3 The usable characters are 0 to 9, A to Z, and a to z. The number of characters is 1 to 15 (not including NULL). The text is not case sensitive.

*4 You cannot set the following IP addresses.
IP addresses start with 127 (decimal)
Class-D IP addresses (224.0.0.0 to 239.255.255.255)
Class-E IP addresses (240.0.0.0 to 255.255.255.255)

*5 The usable characters are 0 to 9, A to Z, and a to z. The number of characters is 1 to 32 (not including NULL). The text is not case sensitive.

*6 Use <Domain name>\<User name> when you specify the domain. You can omit specifying the domain name if the user name in the domain and one in the local are not same.

*7 The usable characters are 0 to 9, A to Z, and a to z, as well as single-byte code (` ~ ! # $ ^ & ( ) _ - { } \ '). The number of characters is 1 to 274 (not including NULL). The text is not case sensitive.

*8 The usable characters are 0 to 9, A to Z, and a to z, as well as single-byte code (` ~ ! @ # $ % ^ & * ( ) _ - + = { } [ ] \ | : ; " ' < > . ? /). The number of characters is 8 to 32 (not including NULL). The text is not case sensitive.

*9 A Virtual SD Memory Card is applicable in the NY-series Controller.

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.

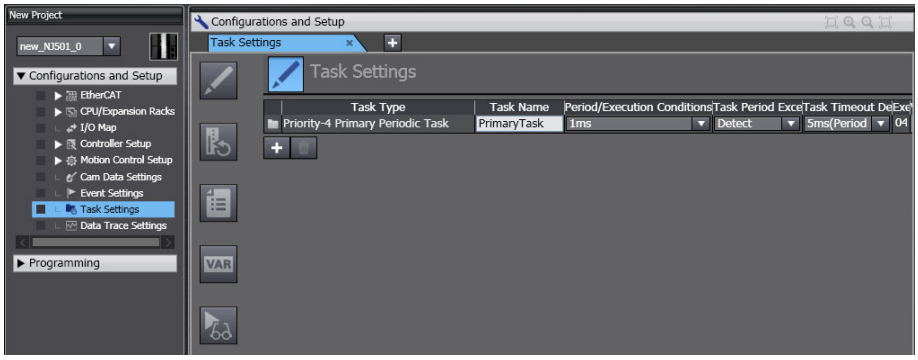
Precautions for Correct Use

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.

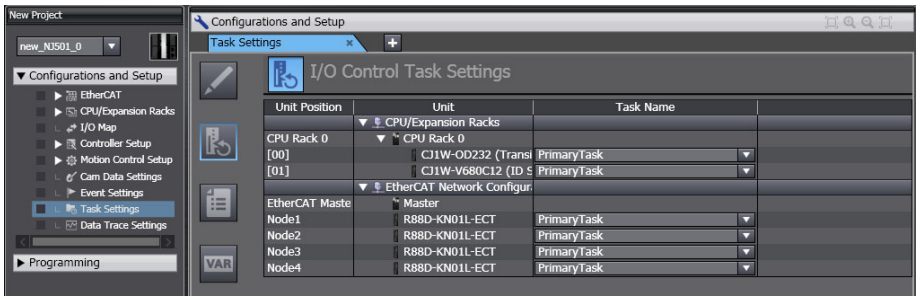


Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
Task Type		Sets the task type.	Primary periodic task Priority-16 periodic task Priority-17 periodic task Priority-18 periodic task Priority-8 event task Priority-48 event task	Primary periodic task	When downloaded to Controller	Not allowed.
	Execution Priority	Sets the task execution priority.	Automatically set according to the task type.	Primary periodic task: 4	When downloaded to Controller	Not allowed.
Task Name		Sets the task name.	Text string	Primary periodic task: PrimaryTask Periodic tasks: PeriodicTask0 Event tasks: EventTask0	When downloaded to Controller	Not allowed.
Period/Execution Conditions		Sets the task period.	Primary periodic task: 500 μs to 8 ms (in 250-μs increments) Periodic tasks: 1 ms to 100 ms (in 250-μs increments) Event tasks: Executed with an instruction or when a variable expression is satisfied.	Primary periodic task: 1 ms Periodic tasks: 10 ms (Priority 16, 17, and 18) Event tasks: Execution with an instruction	When downloaded to Controller	Not allowed.

Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
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 Controller	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 Controller	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 Controller	Not allowed.

I/O Control Task Settings

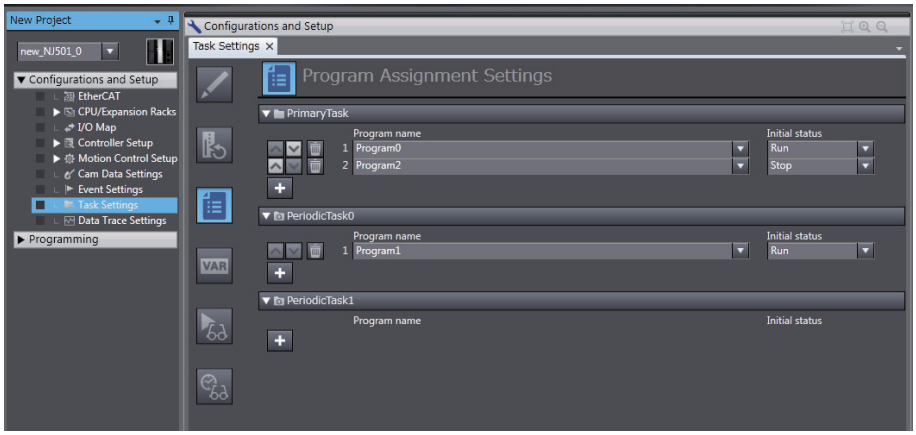
The I/O Control Task Settings are used to set the timing of refresh execution of inputs and outputs.



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	PrimaryTask	When downloaded to Controller	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.



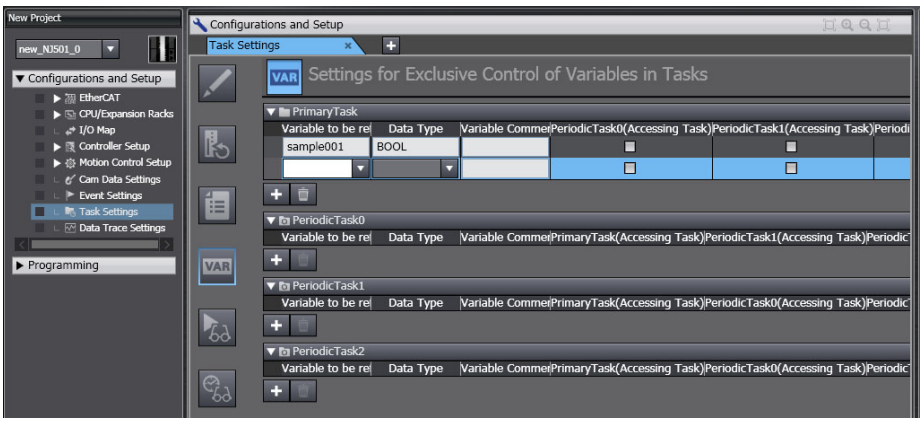
Parameter	Description	Set value	Default	Update timing	Changes in RUN mode
Program Execution Order	Assigns the programs to the specified tasks and sets the order of program execution within the tasks.	Assign the programs in the order to execute them from top to bottom.	Program0	When downloaded to Controller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

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.

Settings for Exclusive Control of Variables in Tasks

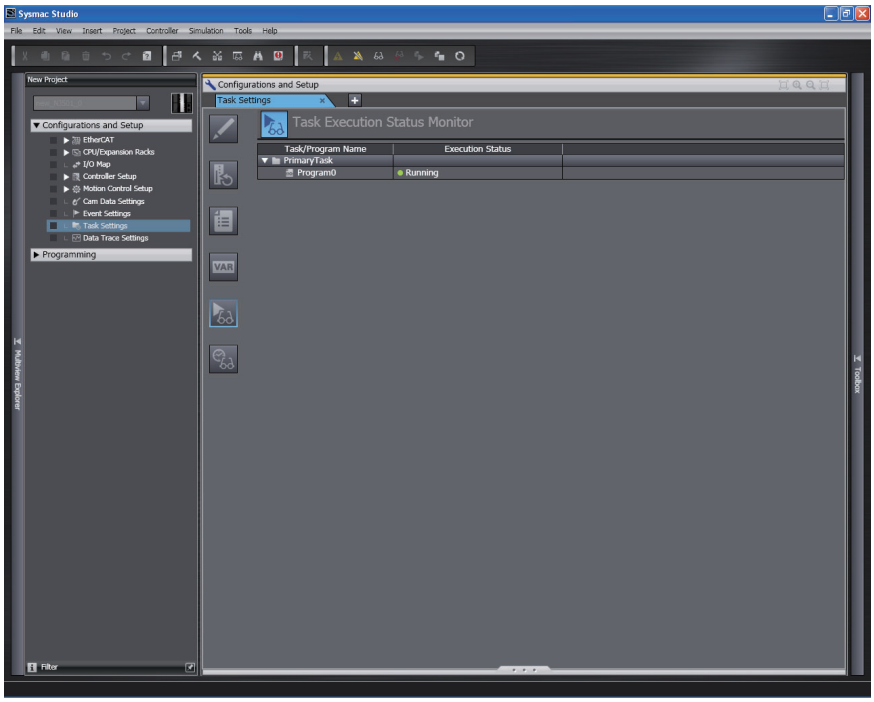
The Settings for Exclusive Control of Variables in Tasks are used to set the tasks that refresh specified global variables and the tasks that access specified global variables.



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 Controller	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.				

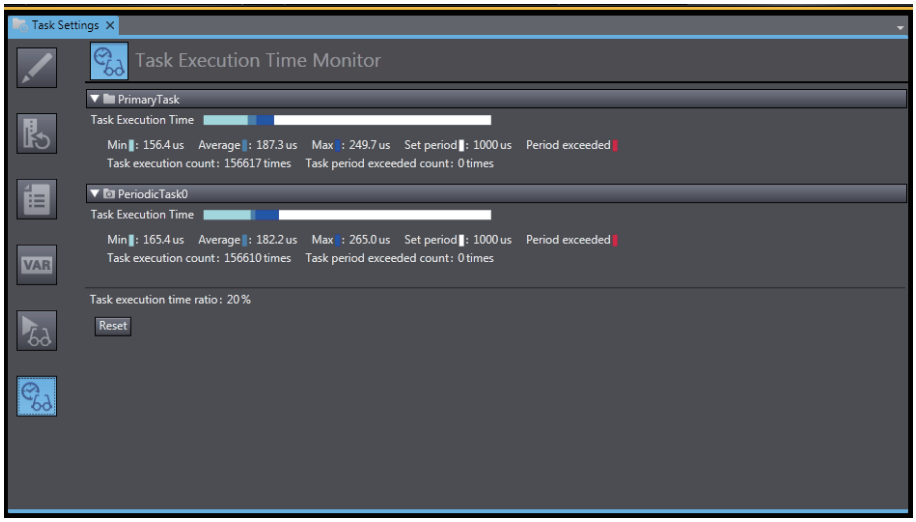
Task Execution Status Monitor

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.



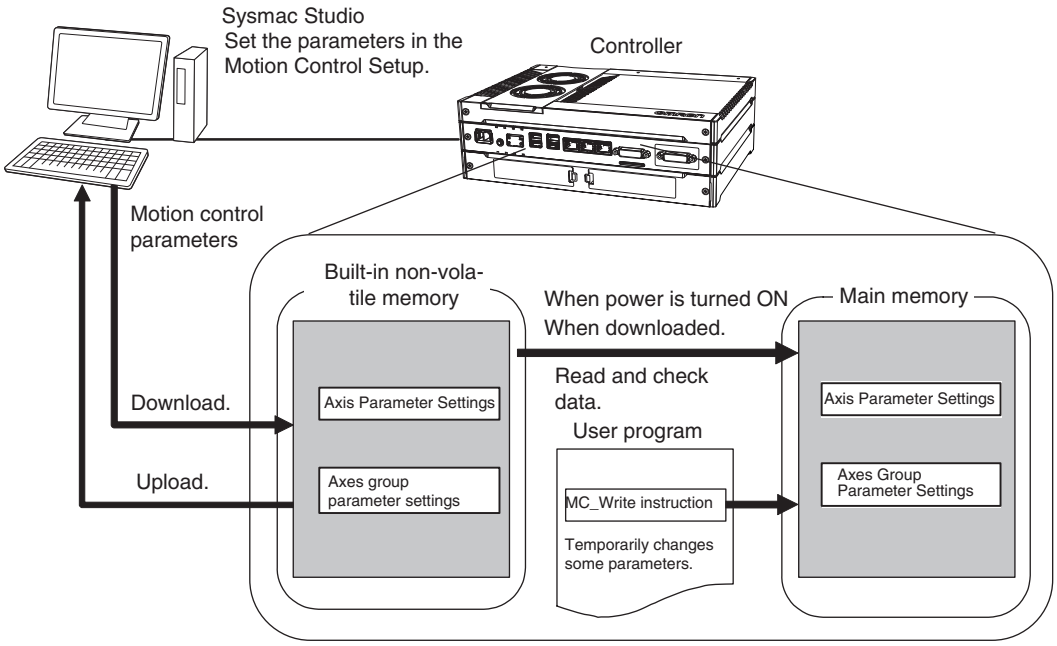
4-3 Initial Settings for the Motion Control Function Module

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

4-3-1 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

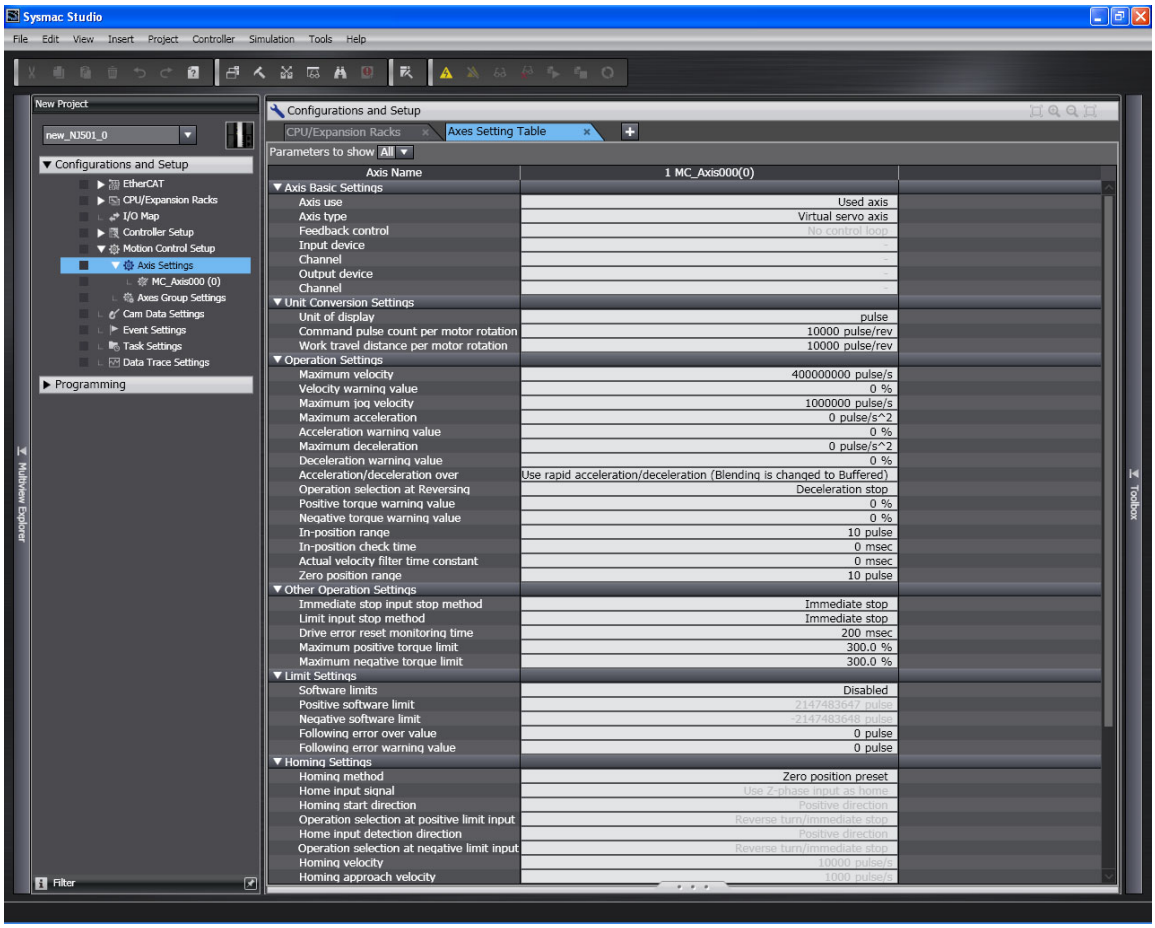


4-3-2 Setting Methods

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 Controller to save them in the non-volatile memory in the Controller. 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 *NY-series Industrial Panel PC / Industrial Box PC Motion Control User's Manual* (Cat. No. W559) for details.

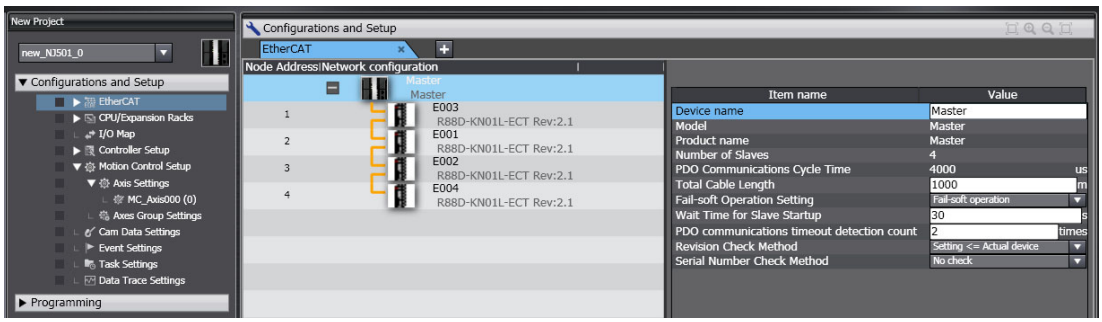
4-4 Initial Settings for the EtherCAT Master Function Module

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 *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherCAT Port User's Manual* (Cat. No. W562) for details.

4-5 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
- SNMP Settings
- SNMP Trap Settings

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

Refer to the *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherNet/IP Port User's Manual* (Cat. No. W563) 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 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

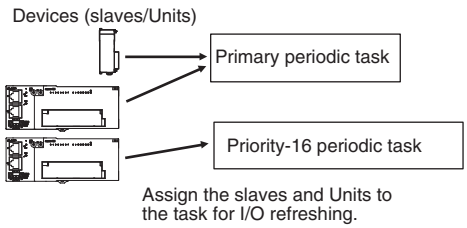
↓

Step 4. Wiring

↓

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

Refer to *1-4 Overall Operating Procedure for the NY-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 NY-series Controllers* *5-9 Task Design Methods and I/O Response Times*
● **I/O refreshing of slaves and Units is assigned to the tasks. You can only assign I/O refreshing to the primary periodic task in an NY-series Controller.** 	*5-5-1 Assigning I/O Refreshing to Tasks*
● **Determine which programs to assign to the primary periodic task, to the priority-16, priority-17, and priority-18 periodic tasks, and to the priority-8 and priority-48 event tasks.**	*5-5-2 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.	*5-6-1 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 Controller 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.	*5-6-2 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*

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.

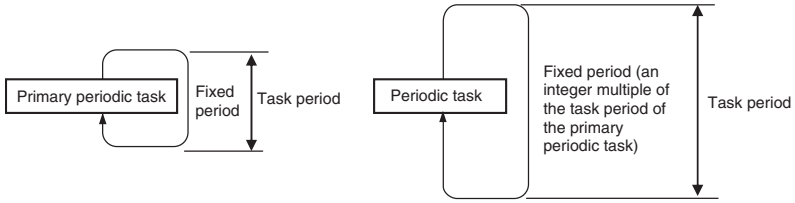
There are three kinds of tasks, as shown in the following table. They are defined by their execution conditions and execution priorities.

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 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, 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

Primary Periodic Task and Periodic Tasks

The NY-series Controller periodically executes both the primary periodic task and periodic tasks.

(The interval in which the NY-series Controller 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 EtherCAT slaves is called I/O refreshing.

You can only assign I/O refreshing for each slave and Unit 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 *NY-series Instructions Reference Manual* (Cat. No. W560) for details on the basic instructions that cannot be used in event tasks.

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.

Instruction	Instruction name	Introduction	
GetMyTaskStatus	Read Current Task Status	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	
GetMyTaskInterval	Read Current Task Period	Reads the task period of the current task.	
Task_IsActive	Determine Task Status	Determines if the specified task is currently in execution.	
Lock	Lock Tasks	Starts a lock between tasks.	Execution of any other task with a lock region with the same lock number is disabled.
Unlock	Unlock Tasks	Stops a lock between tasks.	
ActEventTask	Activate Event Task	Activates the specified event task.	

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.

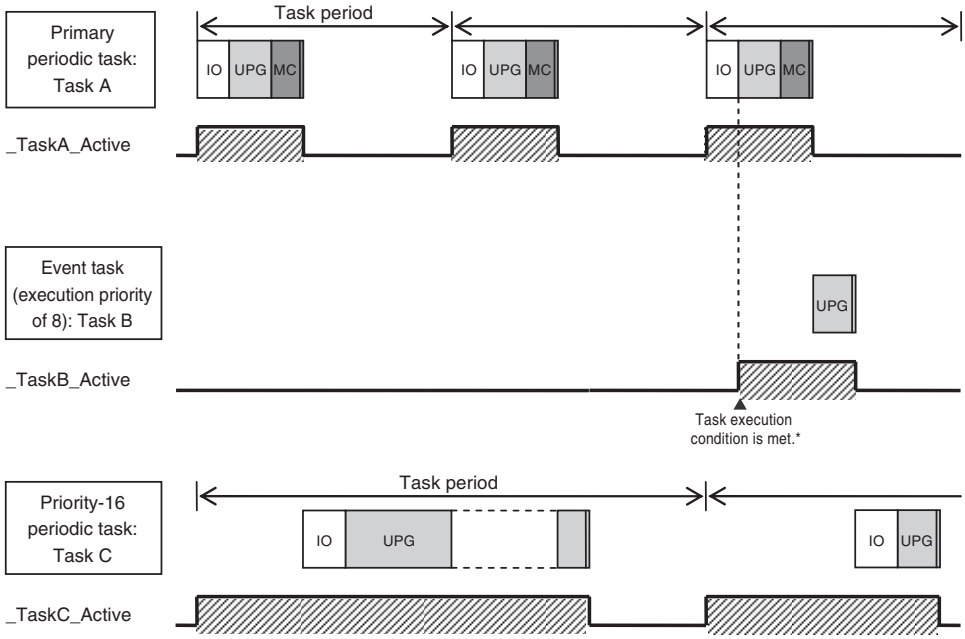
You can also use the GetMyTaskStatus and Task_IsActive instructions to read task status from the user program.

Variable name	Meaning	Function	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
_*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*.

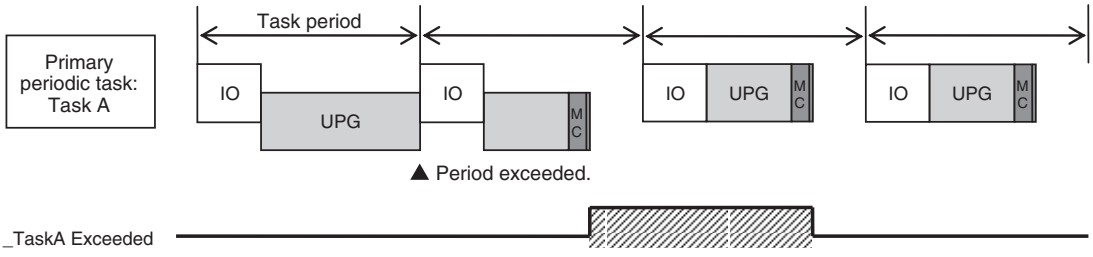
Flag Operation

● 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 NY-series Controllers

This section describes the specifications and basic operation of tasks for NY-series Controllers with a multi-core processor.

5-3-1 Specifications of Tasks for NY-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 to 8 ms (in 250-μs increments)
Task period of periodic task	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: 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.

5-3-2 Guidelines for Separating Tasks for NY-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-16 periodic task	• Programs that contain controls for some slaves and Units whose I/O refreshing is assigned to the primary periodic task. • Programs with a relatively low execution priority that require periodic user program execution or system common processing. • 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 NY-series Controllers

With a multi-core processor, the NY-series Controller can execute multiple tasks, the tag data link service, and system services in parallel. The order in which tasks are executed depends on the execution priority that is set for each task.

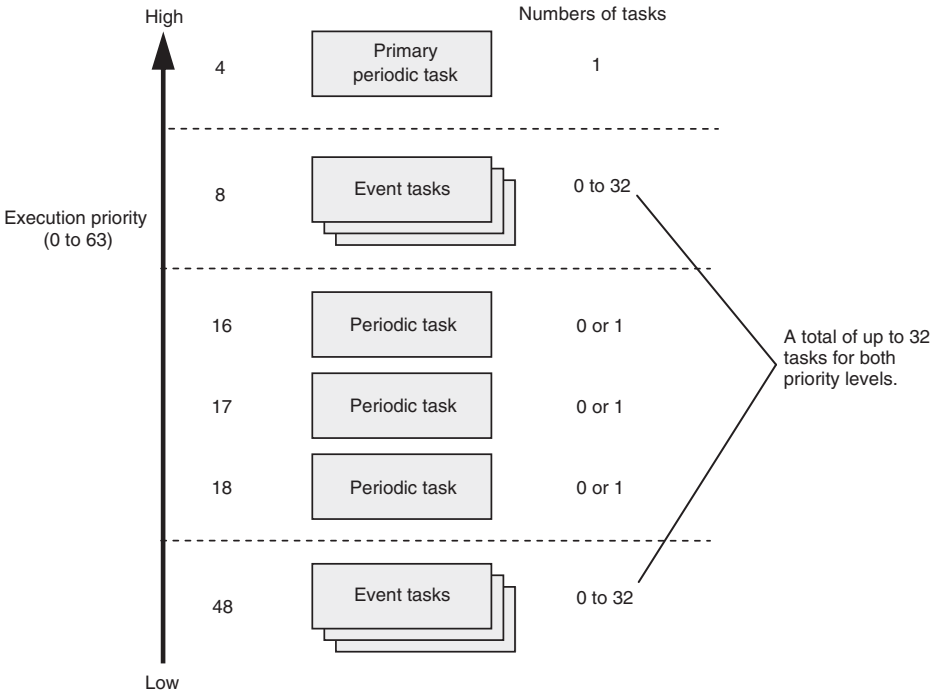
Task Execution Priority

The type of the task determines its execution priority. The NY-series Controller 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 NY-series Controller 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	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 NY-series Controllers* for the order of execution.



Task Periods for the Primary Periodic Task and Periodic Tasks

The NY-series Controller 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 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

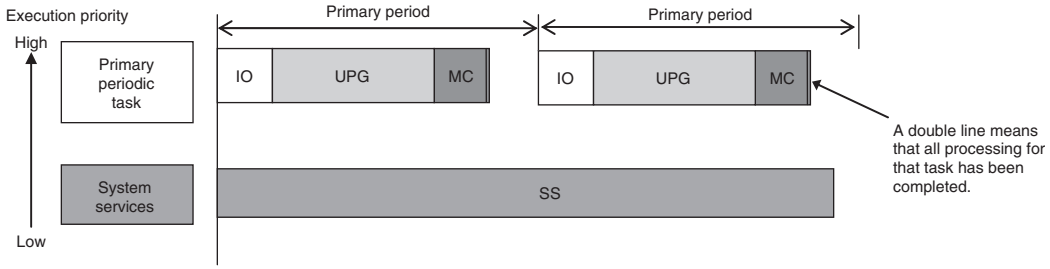
This section gives a few examples of the execution order for the primary periodic task and periodic tasks.

Refer to *5-3-5 Event Task Execution Timing for NY-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.

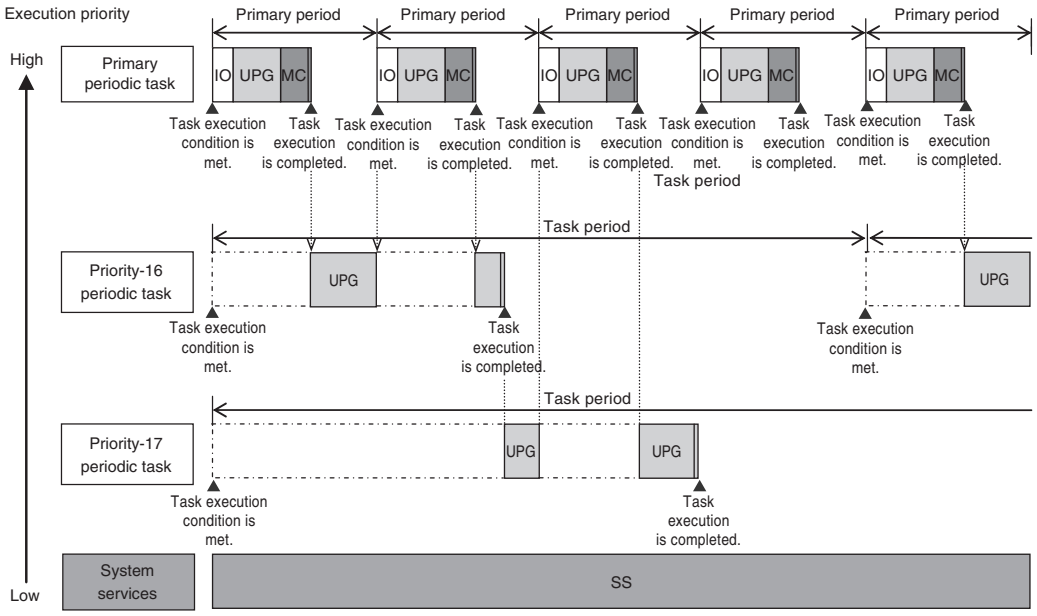
The system service shown in this figure refers to non-task related processing, such as communications processing, that is performed by the Controller. Refer to *Processing Performed in System Services* on page 5-32 for details on the system services.



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 is executed after execution of the primary periodic task is completed.
- 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 ever four primary periods, execution of the primary periodic task and the priority-16 periodic task will start at the same time.
- 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 *5-6-1 Ensuring Concurrency of Variable Values between Tasks* for details.

Tasks and Operating Modes

The relationship between the operating modes and tasks of an NY-series Controller 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 task	
Event task	Event tasks are executed only in RUN mode.

Precautions for Correct Use

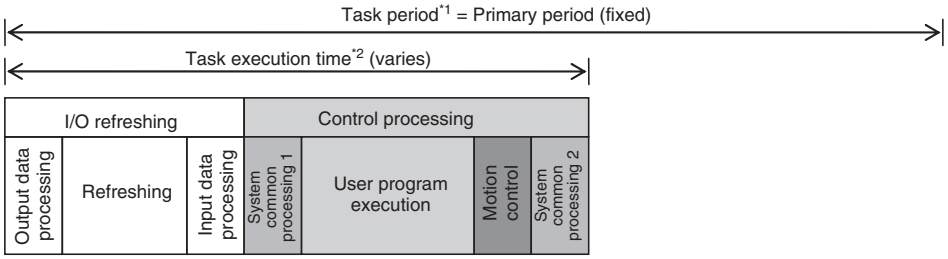
- Even if the execution condition for an event task is already met when you change the operating mode to RUN mode, the event task will not be executed. An event task is executed only when its execution condition changes from not met to met during RUN mode.
- Even in RUN mode, an event task is not executed 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 Controller executes tasks in this fixed period. This is a preset, fixed time.

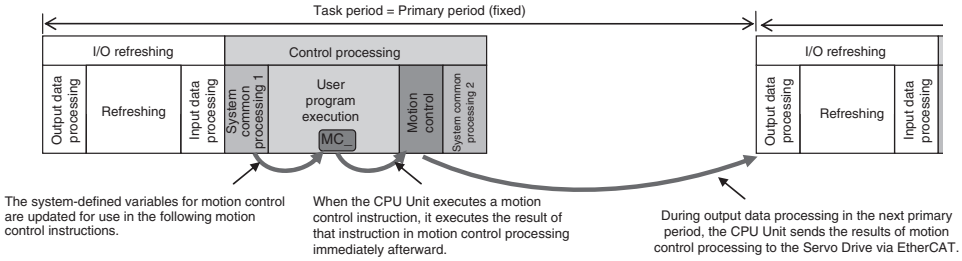
*2 : Task execution time — This is the actual time it takes from the point that the execution condition is met until execution is completed.

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 is performed (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 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 are updated.

*2 When there are motion control instructions in user program execution in the primary periodic task, the Controller executes the results from those instructions immediately afterward in motion control processing as shown below. The Controller 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.

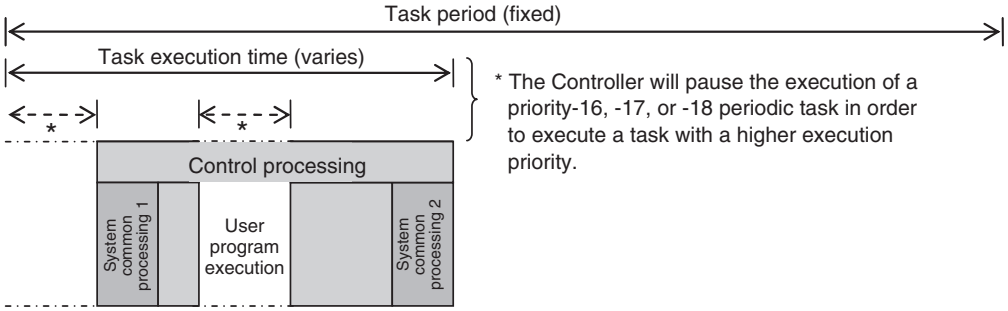


When there is a motion control instruction in user program execution in the priority-16 periodic task, the Controller executes the result from that instruction in the motion control processing (MC) of the next primary periodic task.

Refer to *5-9-3 System Input and Output Response Times* for details.

● 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.



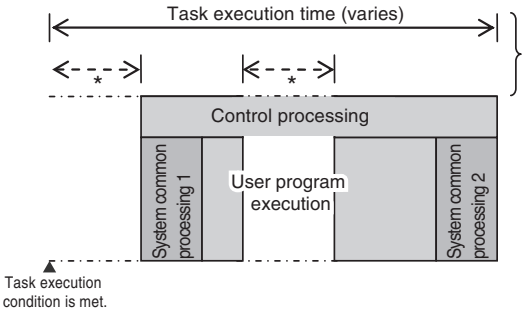
Processing	Processing contents
System common processing 1	• Processing for exclusive control of variables in tasks is performed (when accessing tasks are set). • Data trace processing (sampling and trigger checking) is performed.
User program execution	• Programs assigned to tasks are executed in the order that they are assigned.
System common processing 2	• Processing for exclusive control of variables in tasks 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.

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 *5-6-1 Ensuring Concurrency of Variable Values between Tasks* for details on exclusive control.

5-3-4 Event Task Execution Conditions for NY-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*1	When you want to simplify the user program by executing event tasks without user programming

*1 Refer to *Execution Timing When the Execution Condition Is a Condition Expression for a Variable* on page 5-22 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

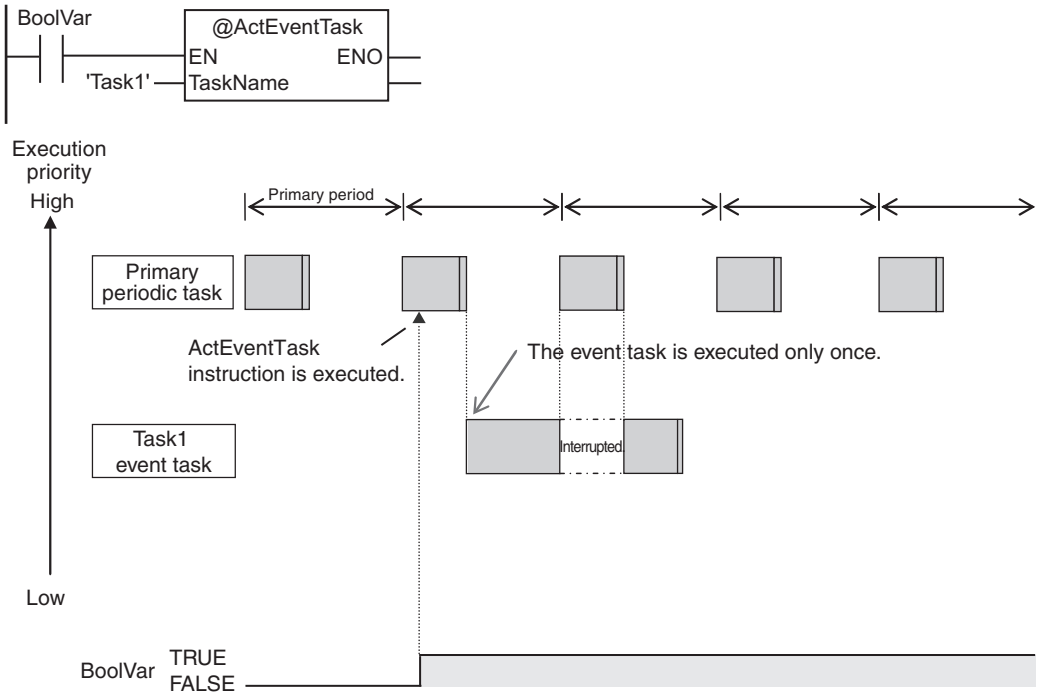
When the ActEventTask (Execute Event Task) instruction is executed in the user program, the specified event task is executed once. Refer to the *NY-series Instructions Reference Manual* (Cat. No. W560) for the detailed specifications of the ActEventTask instruction.

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

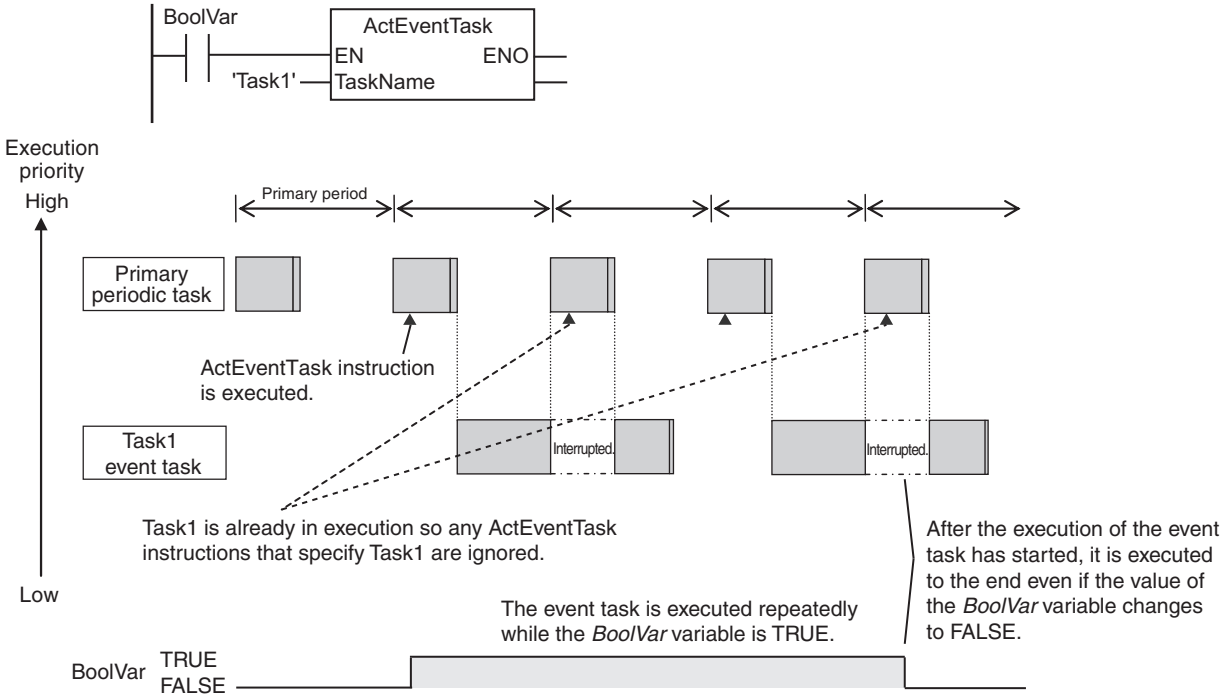
Example 1: Executing an Event Task Only Once When the Value of a Variable Changes

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 types	**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.

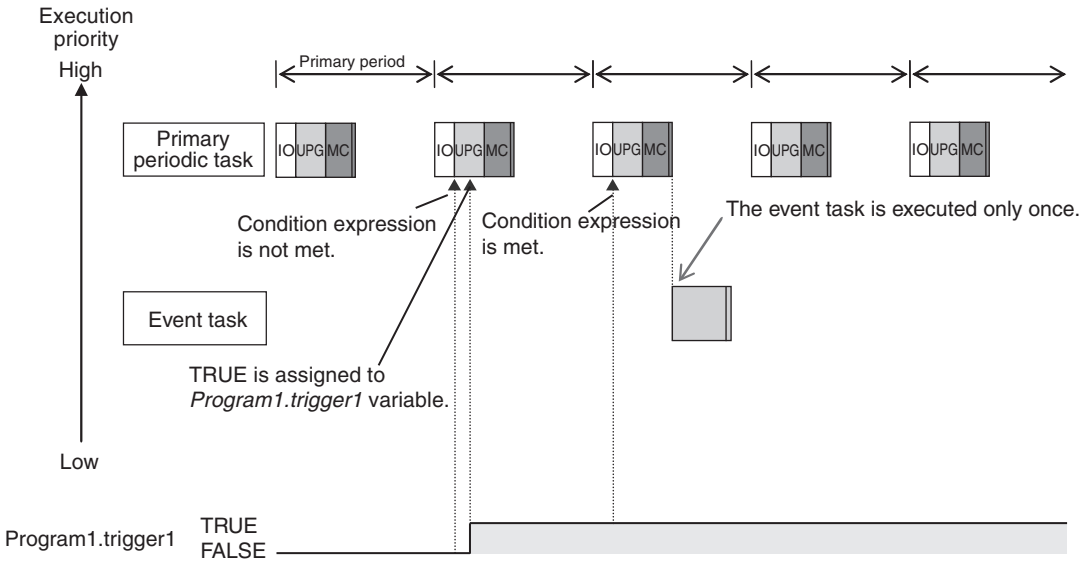
Refer to *Basic Data Types and Derivative Data Types* on page 6-30 for the valid range of values for each data type.

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

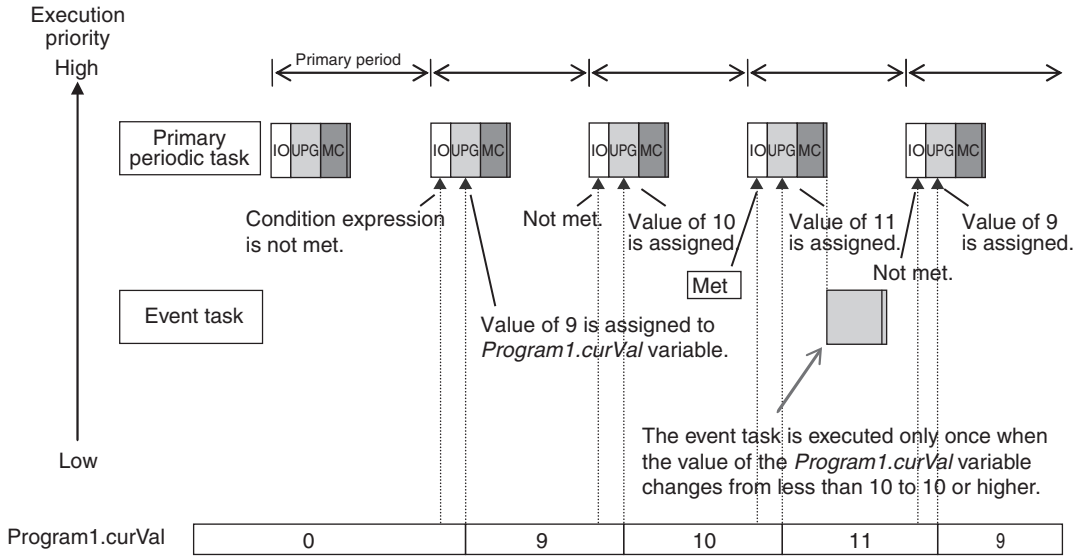
Example 1: Execution Condition for Event Task Set to a Change to TRUE of the *Program1.trigger1* Boolean Variable

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 Controller evaluates whether the condition expression is met in the next primary periodic task. This means that the event task will be executed on completion if this evaluation against the condition expression in the next primary periodic task after the Controller evaluates that the condition expression is met.

5-3-5 Event Task Execution Timing for NY-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.

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 *5-6-1 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.

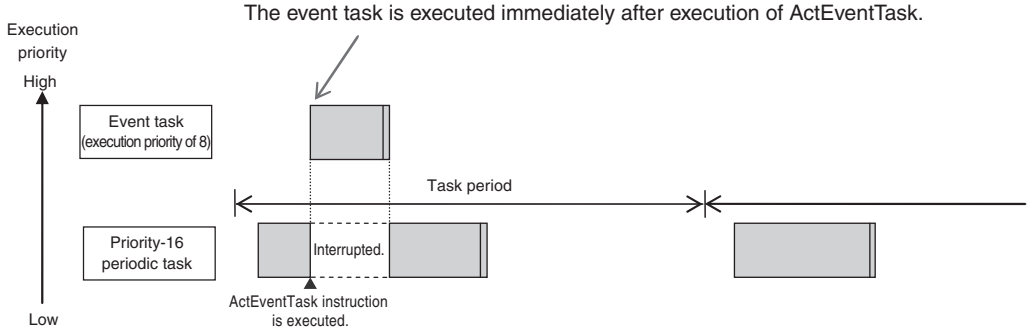
Differences in Execution Timing Based on the Execution Conditions of Event Tasks

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.

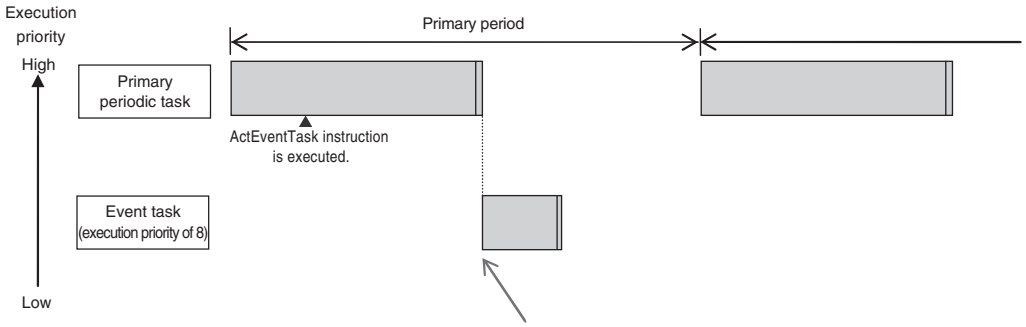
● Execution Timing When the Execution Condition Is an ActEventTask Instruction

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 NY-series Controller 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

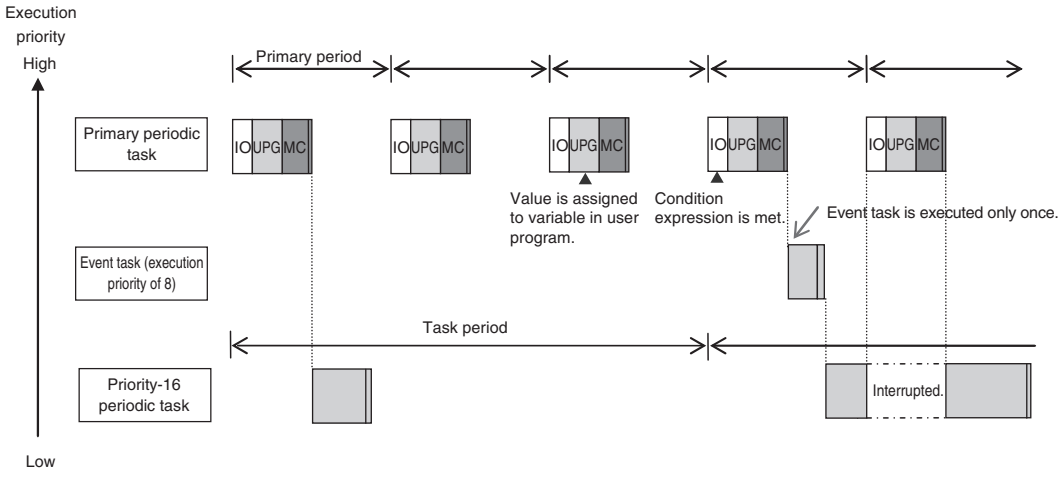


Event task is executed when the primary periodic task is completed.

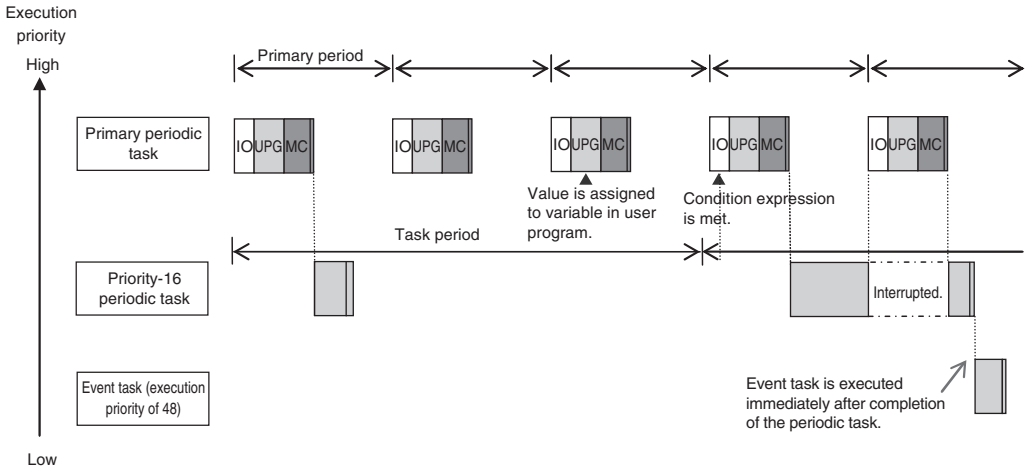
● Execution Timing When the Execution Condition Is a Condition Expression for a Variable

The condition expression is evaluated for a match inside the primary periodic task. The execution condition for an event task is met when it is evaluated to match the condition expression. The NY-series Controller 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
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

- For NY-series Controllers, 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.

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

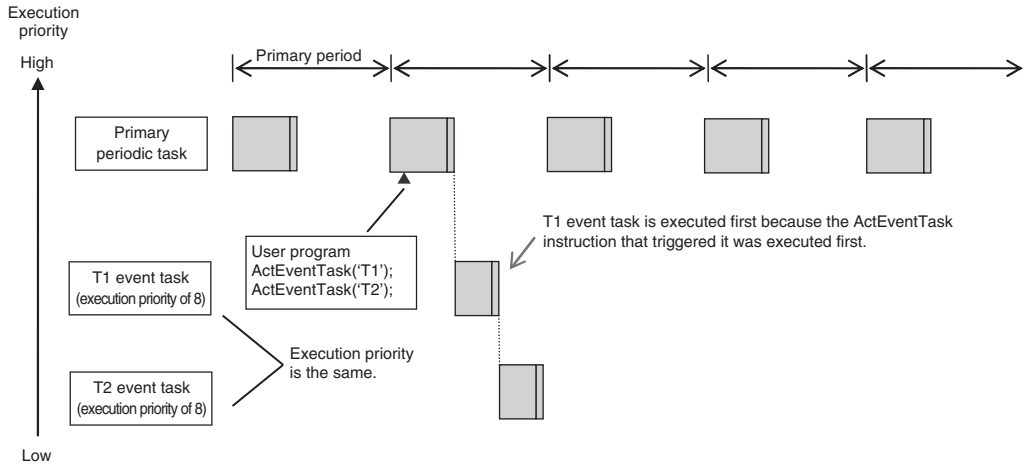
- In order for an event task to be executed, the condition expression must be met in the evaluation after the previous evaluation where the condition expression was not met. This means that even if the status of the condition expression changes from not met to met, if the condition returns to not met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

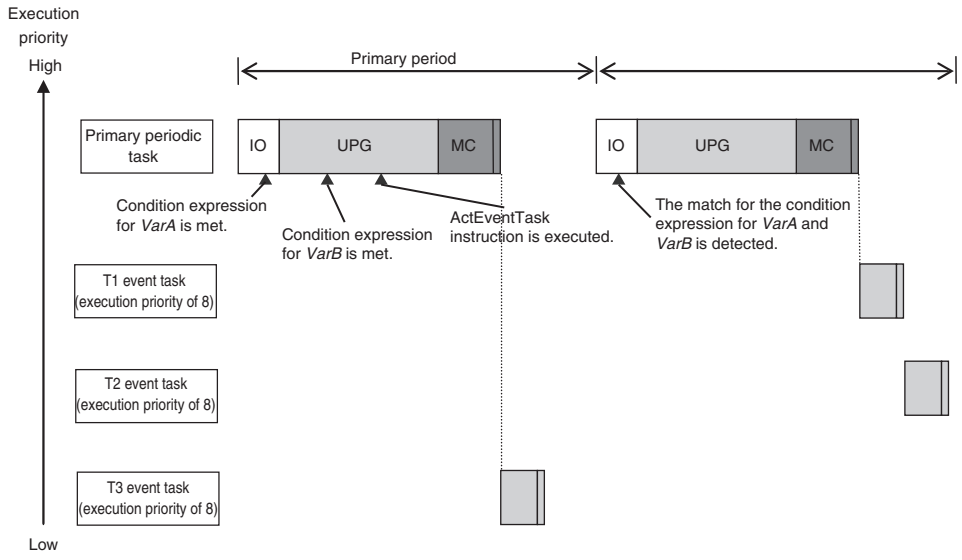
(1) The condition expression for *VarA* is met during I/O refreshing.

(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.

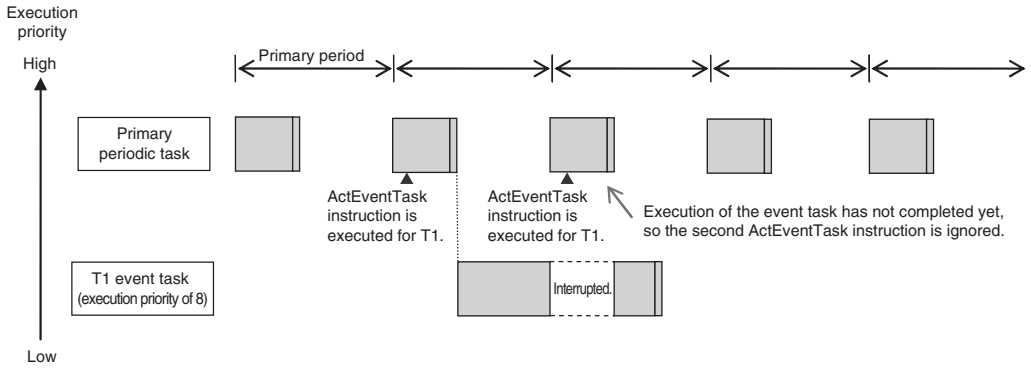
(4) The ActEventTask instruction is executed in the user program, so T3 is executed.

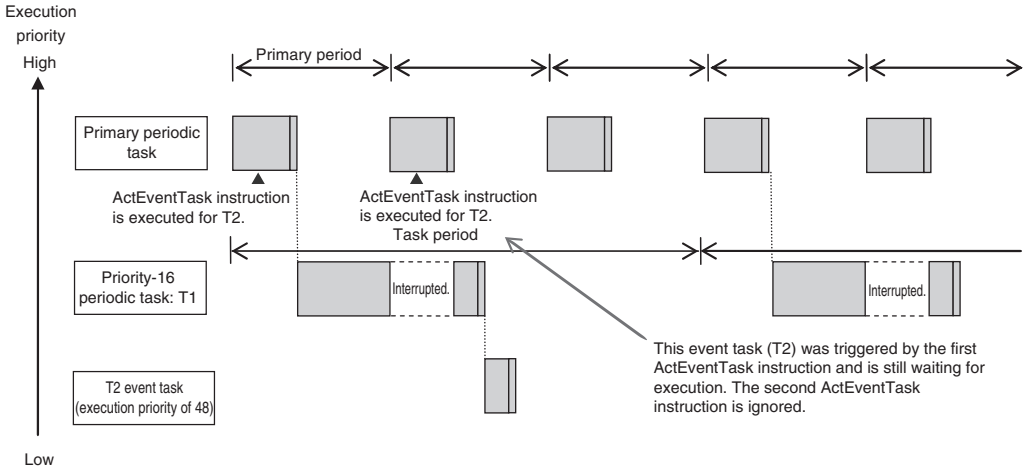
(5) When I/O refreshing is executed in the primary periodic task, the match is detected that the condition expressions for *VarA* and *VarB* are met, so T1 and T2 are executed. If the match is detected that more than one condition expression is met in the same execution period, the order of execution 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 Tag Data Link Service and System Services

The NY-series Controller 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 on the NY-series Controller.*1	System services include services with low execution priority, such as communications processing and other services.
Execution timing	The execution interval and the time that is required for each execution depend on the model of the NY-series Controller and on the tag data link settings.*2	For NY-series Controllers, the system services are executed at the required time without being affected by the tasks.

*1 Refer to the *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherNet/IP Port User's Manual* (Cat. No. W563) for details on the processing that is performed for tag data links.

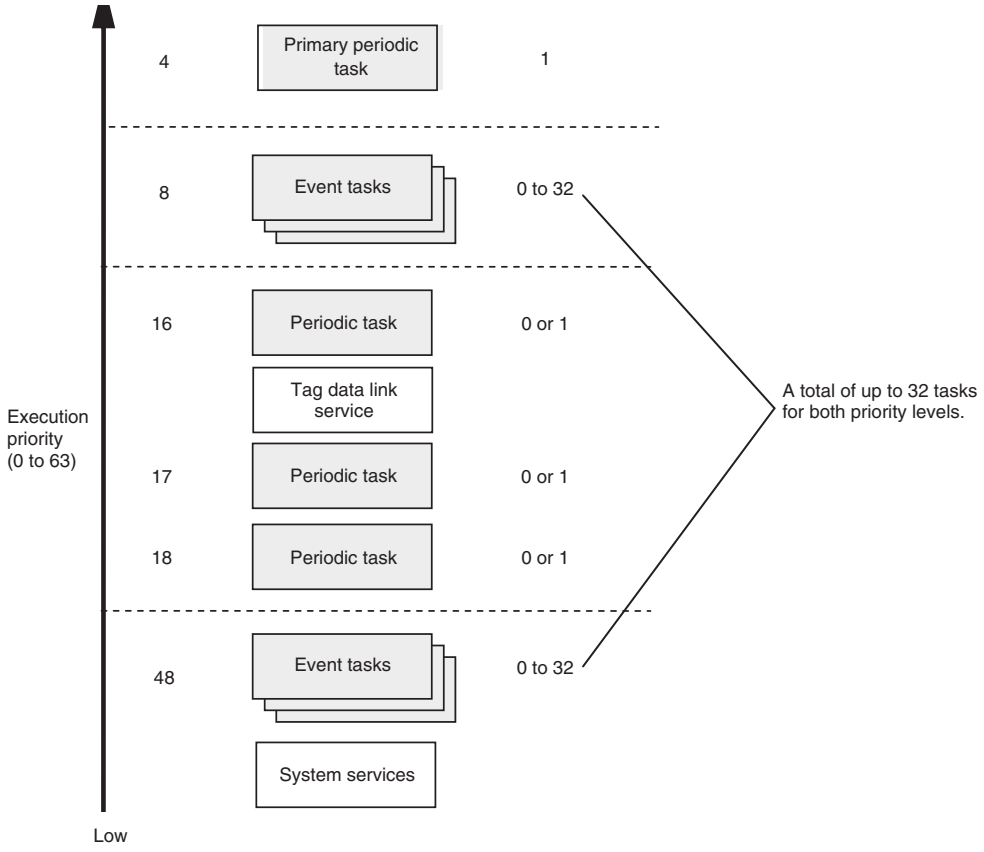
*2 Refer to *5-4-2 Processing Performed in and Execution Timing of the Tag Data Link Service* for details on the execution interval and the time that is required for execution of tag data links.

5-4-1 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.

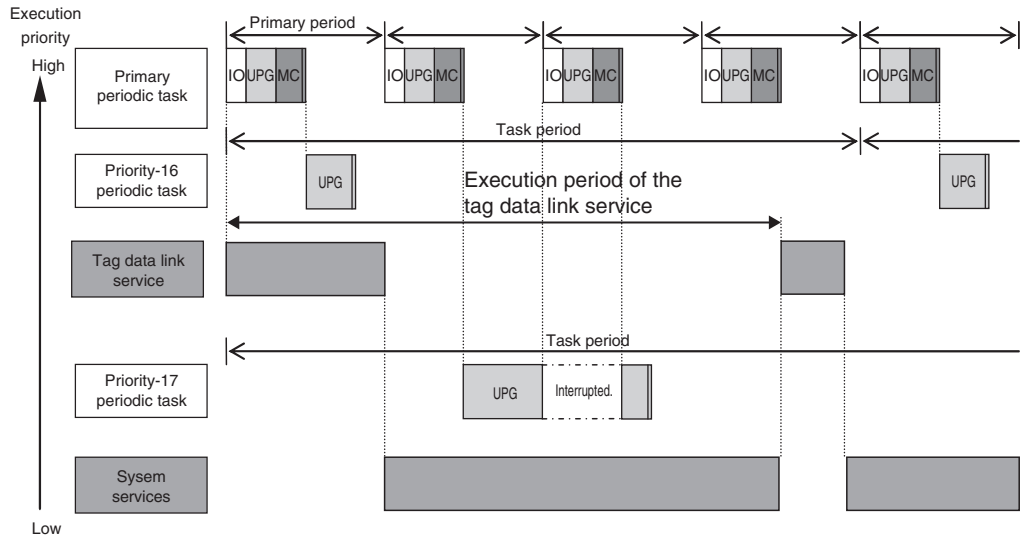


Examples of the Order of Execution for the Tag Data Link Service and System Services

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.

● NY-series Controllers

- You can execute multiple tasks, the tag data link service, and system services in parallel.
- The system services are not executed while the tag data link service is in progress.



The execution interval and the execution time for one execution depend on the model of the NY-series Controller and on the tag data link settings. Refer to *5-4-2 Processing Performed in and Execution Timing of the Tag Data Link Service* for details.

5-4-2 Processing Performed in and Execution Timing of the Tag Data Link Service

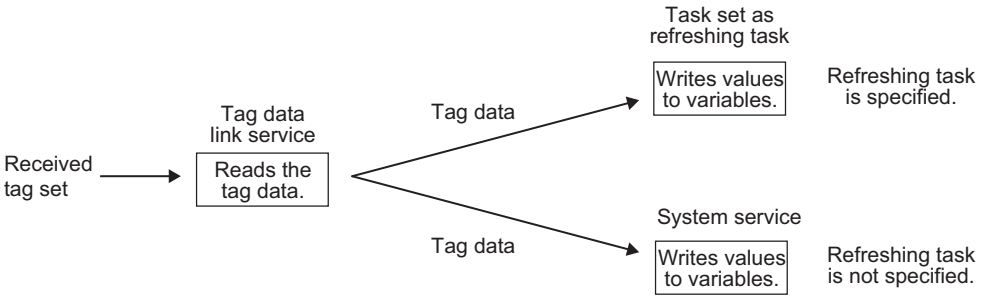
This section describes the processing that is performed in and the execution timing of the tag data link service.

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 NY-series Controller is used.

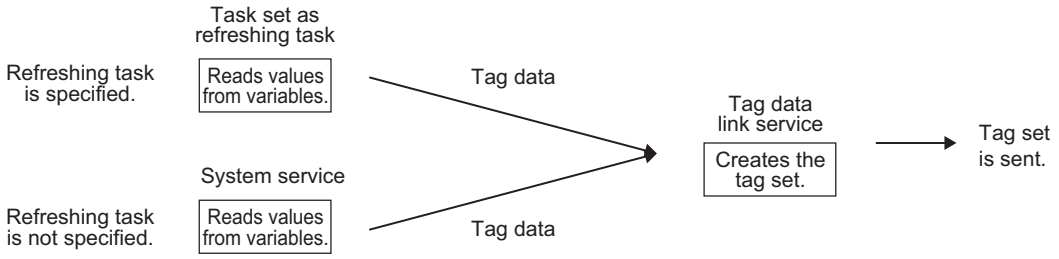
● Flow of Tag Data Reception Processing

- 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.



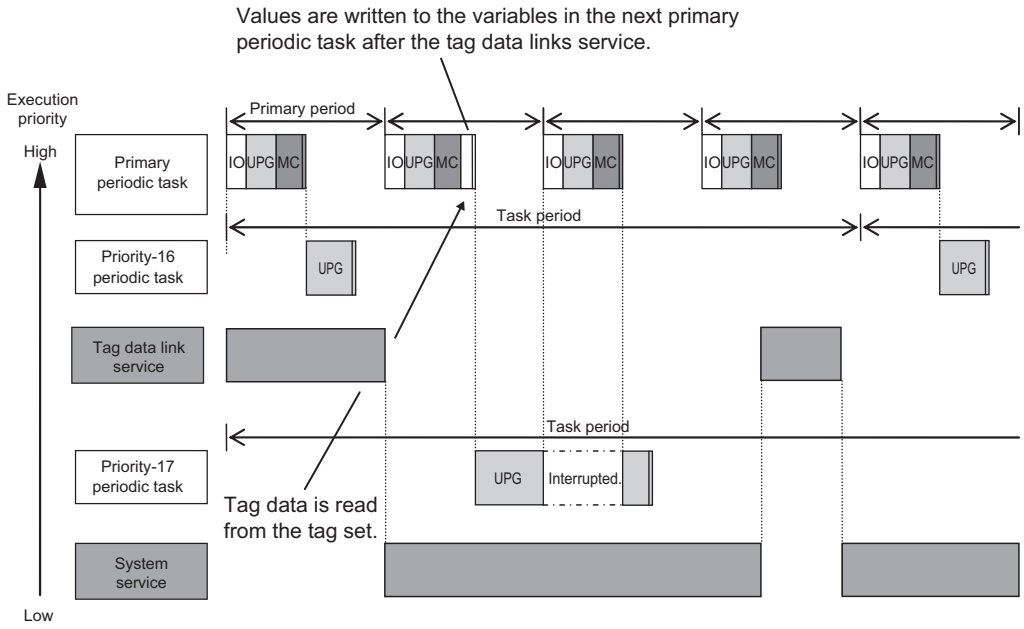
Additional Information

Differences in the Timing of Processing Tag Data Links Depending on Whether a Refreshing Task Is Set

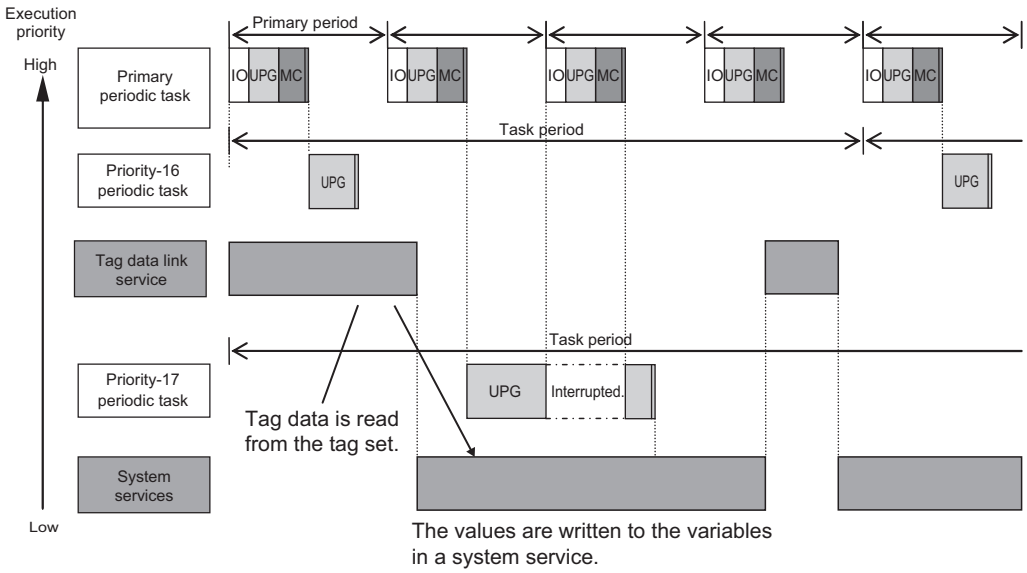
The process to write values to and read values from variables is different depending on whether a refreshing task is set. If a refreshing task is set, the values are read and written in that task. If a refreshing task is not set, the values are read and written in a system service. This means there is 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 NY-series Controller 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 link service.



- When a Refreshing Task Is Not Specified
  The values are written to the variables in a system service.



5-4-3 Processing Performed in and Execution Timing of the System Services

This section describes the processing that is performed in and the execution timing of the system services.

Processing Performed in System Services

System services include the following processing.

System service	Description
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
Internal port service	• Processing of message service requests, such as CIP commands and HTTP commands, from Windows inside the NY-series Controller.
Built-in EtherCAT port service	• Execution of EtherCAT message communications
Virtual 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

Execution Timing of the System Services

For NY-series Controllers, the system services are executed without being affected by the tasks. However, the system services are not executed while the tag data link service is in progress.

There is no priority in the processing of system services. All of the processing is executed in parallel with time slicing.

5-5 Assignment and Settings Related to Tasks

This section describes the assignment and setting related to tasks.

5-5-1 Assigning I/O Refreshing to Tasks

I/O refreshing of the EtherCAT slaves is assigned to the tasks. Unit of assignment is 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 5-34.

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	Primary periodic task	Slave Terminal
EtherCAT slaves	Primary periodic task	Slave

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 **Configuration and Setup - Task Settings** on 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** on the Sysmac Studio. The tasks to perform I/O refreshing are set.

Refer to the *NY-series Industrial Panel PC / Industrial Box PC Motion Control User's Manual* (Cat. No. W559) for details.

Timing of I/O Refreshing

The table below shows when I/O is refreshed for EtherCAT slaves.

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

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-ECC20□	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.

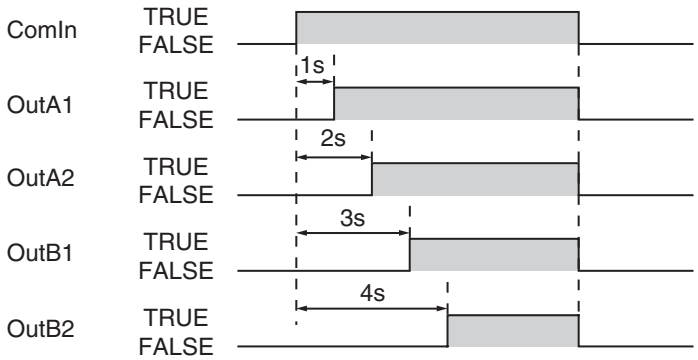
Position	Port	Description	R/W	Data type	Variable	Variable comment	Variable type
Unit1	NX-ID3317						
	Input Bit 00	Input bit 00	R	BOOL	ComIn	Common input value	Global variable
Unit2	NX-OD3256						
	Output Bit 00	Output bit 00	W	BOOL	OutA1	Output value A1	Global variable
	Output Bit 01	Output bit 01	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

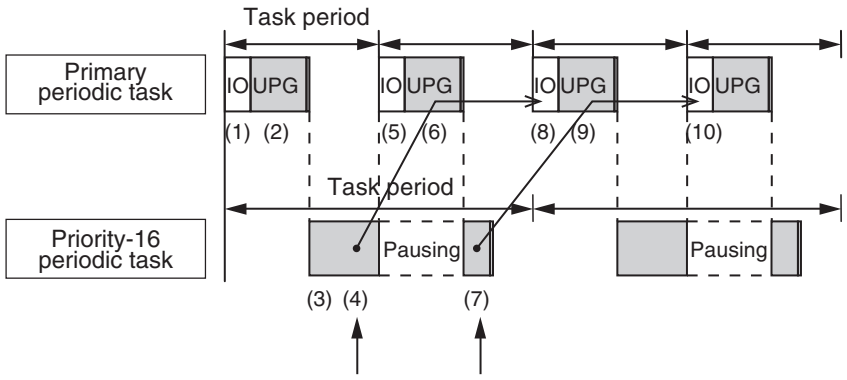
The primary periodic task and priority-16 periodic task are used. The I/O refreshing is assigned to the primary periodic task.

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 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 *5-6-1 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.

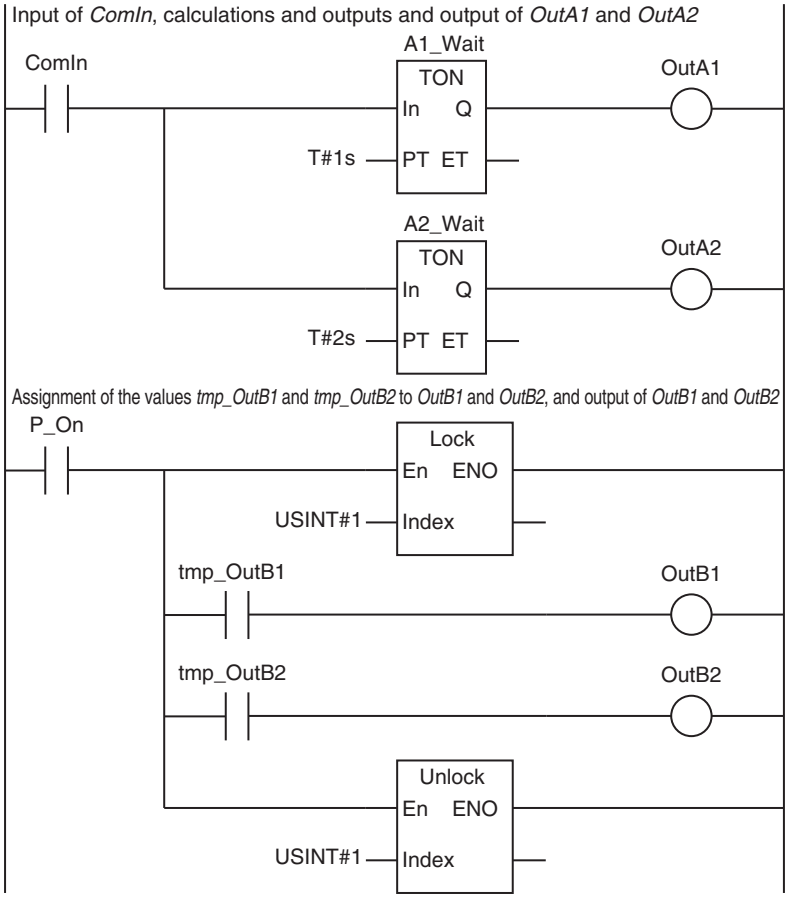
Internal variable table

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.

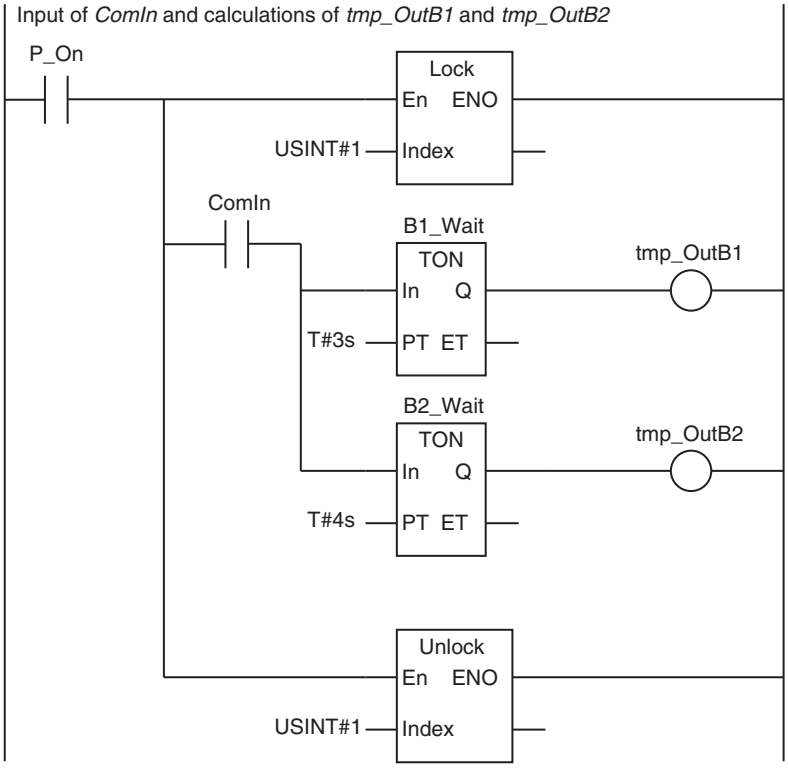
Internal variable table

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



5-5-2 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 Controller 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** on the Sysmac Studio. Refer to *Program Assignment Settings* on page 4-9 for details.

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.

5-5-3 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 downloaded to Controller	Not allowed.
	Execution priority	Always 4.	---		
Task Name		Text string	Primary-Task		
Period/ Execution Conditions	Task period*1	500 μs to 8 ms (in 250-μs increments)	1 ms		
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 occurred). • Do not detect (an observation is recorded in event log). Refer to *Task Period Exceeded* on page 5-53 for details.	Detect.		
Task Timeout Detection Time		Set the time to detect a timeout 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 5-54 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 5-51 for details.	3%		

*1 The process data communications cycle (process data communications cycle) 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 down-loaded to Controller	Not allowed.
	Execution priority	Automatically set to 16, 17, or 18.	---		
Task Name		Text string	Periodic Task0		
Period/ Execution Conditions	Task period	Refer to *5-3-1 Specifications of Tasks for NY-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 down-loaded to Controller	Not allowed.
Task Name	Text string	EventTask0		
Execution Condition	Select either Execution by instruction or When a variable expression is satisfied.	Execution with an instruction		
Task Timeout Detection 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.

5-6 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.

5-6-1 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. These are collectively called the exclusive control of variables in 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.

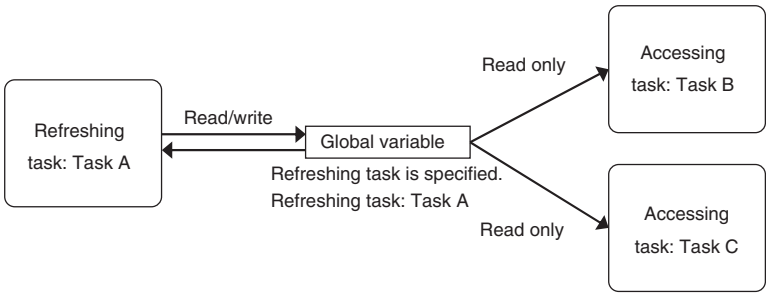
Method 2: Write the global variable from more than one task.
Use the task exclusive control instructions.

Method 1: Settings for Exclusive Control of Variables in Tasks

● Introduction

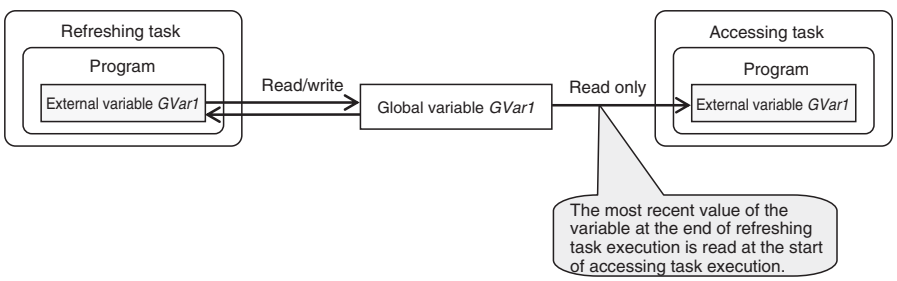
You can specify the task that refreshes a global variable and the tasks that access the global variable. This ensures the concurrency of the value of the global variable from the point of view of the tasks that access the variable.

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.



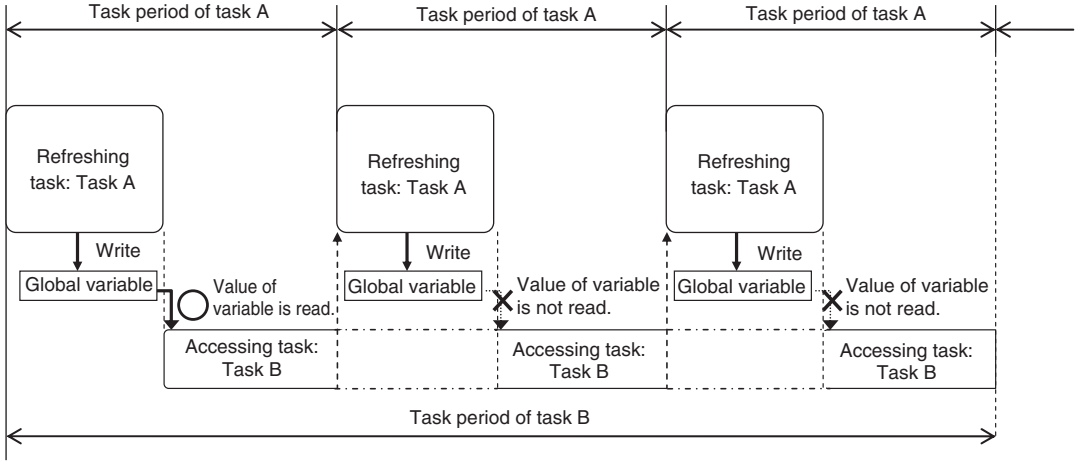
● Application Example

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.

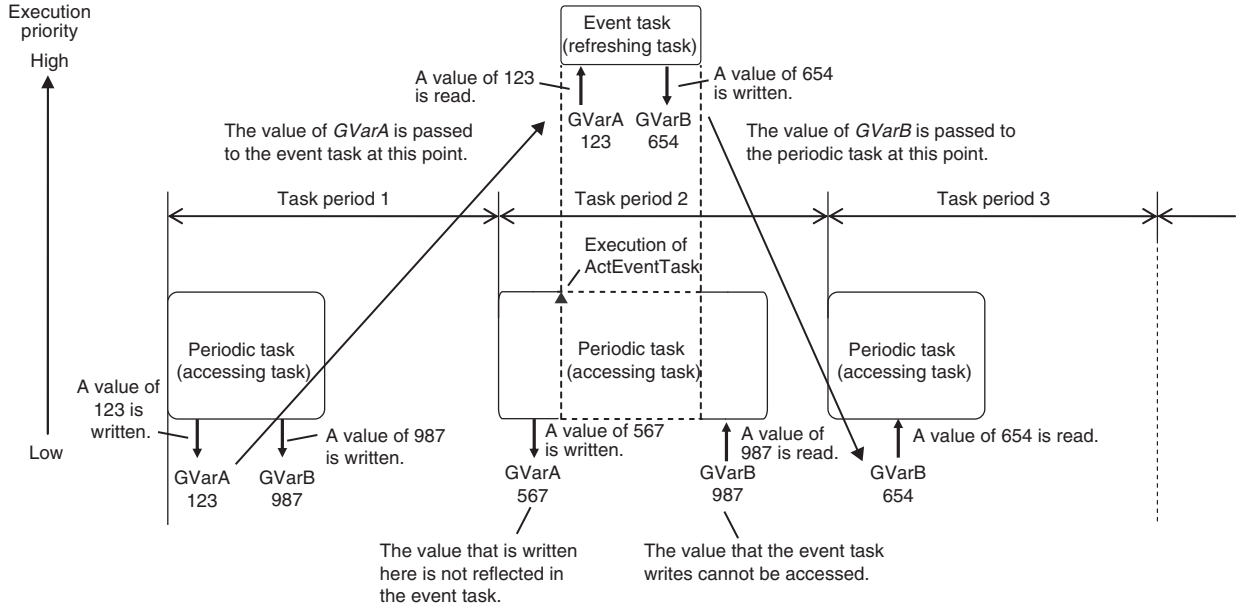


This will allow you to maintain the concurrency of the values of global variables within the tasks without performing any special programming.

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 variables 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 *NY-series Instructions Reference Manual* (Cat. No. W560) 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 *8-4-4 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 Settings for **Exclusive Control of Variables in Tasks under Configurations and Setup** – **Task Settings** on the Sysmac Studio.

Refer to *Settings for Exclusive Control of Variables in Tasks* on page 4-10 for details.

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.

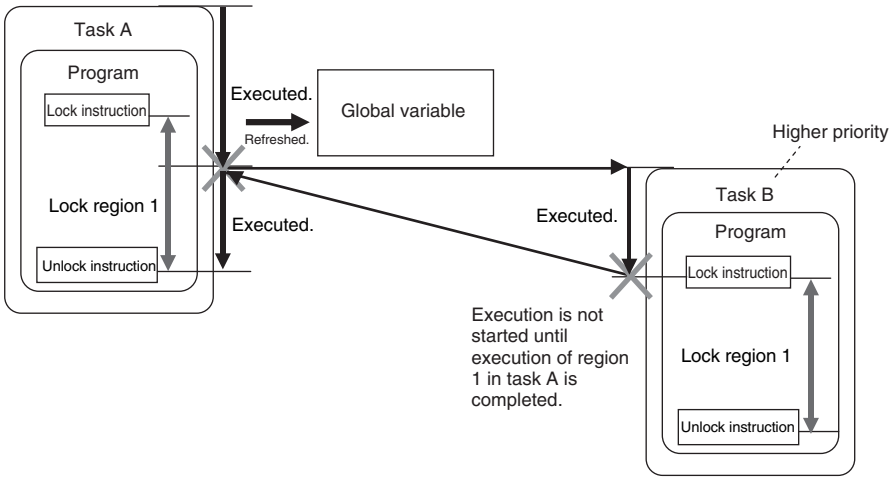
The task exclusive control instructions create a lock region from one Lock instruction to the next Unlock instruction. If a lock region in one task is being executed, the lock regions with the same lock number in other tasks are not executed. If you place the instructions that write to the global variable 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 *NY-series Instructions Reference Manual* (Cat. No. W560) for details.

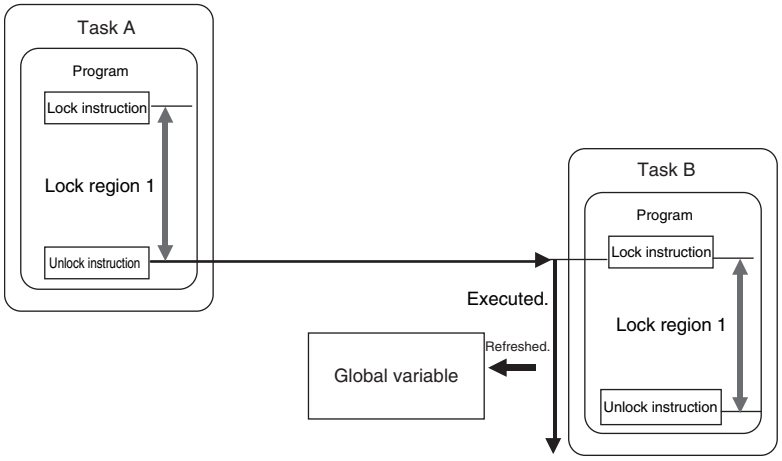
Example:

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 together as a set 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 ReqTrg=TRUE AND gReq=FALSE THEN            (* Access the exclusive flag. *)
  ReqBusy:=TRUE;
  UserDefFB_ins.PutData:=gPar1;               (* Read the parameter from MainTask. *)
  UserDefFB_ins.PutData:=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;
```

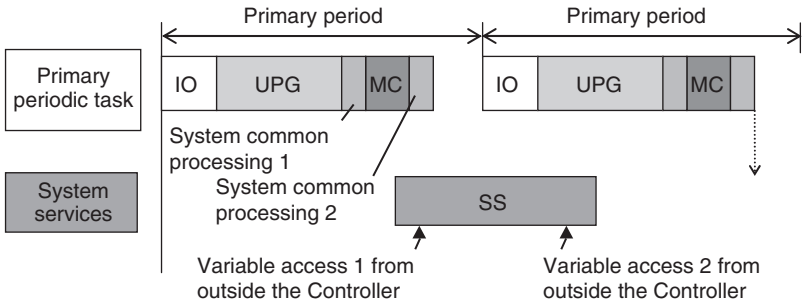
5-6-2 Variable Access from Outside the Controller

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.

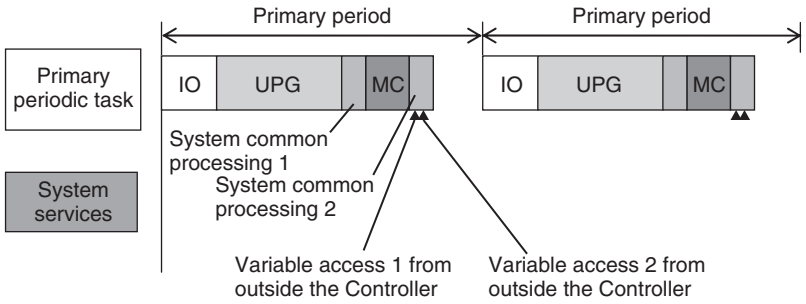
● Accessing Variables from Outside the Controller during the System Service

The access to multiple variables may not be completed in the same task period.



● Accessing Variables from Outside the Controller during the System Common Processing 2

The access to multiple variables is securely executed in the same task 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	Task that update 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.

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.

Refer to *Settings for Exclusive Control of Variables in Tasks* on page 4-10 for the details on the exclusive control of variables in tasks.

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 are given in the following table for the NY-series Controller.

Controller model number	Constant value [μs]			
	a	b	c	d
NY5□2-□□□□	0.0006	0.100	1.40	6.60

● Setting Variable Access Time

Set the variable access time in the **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 NY5□2-□□□□.

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 * 0.0006 + 8 * 0.100 + 1 * 1.40 + 6.60

= 9.16 [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	9.16
2	8.52
3	8.7

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 9.16 μs. Therefore, when the task period is 500 μs, the variable access time is set to 9.16/500 ≈ 2%.

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) * 0.0006 + (8 + 4 + 1) * 0.100 + (1 + 1 + 1) * 1.40 + 6.60

= 13.18 [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.

5-7 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

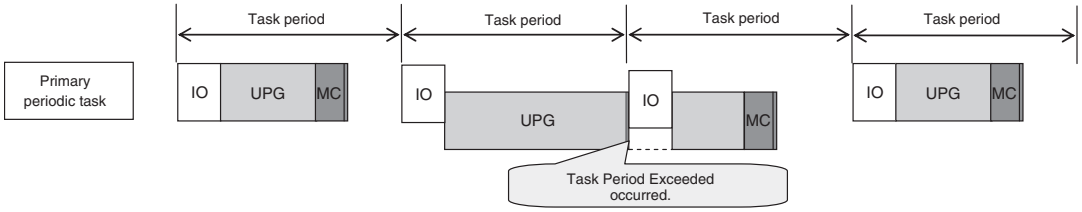
Task Period Exceeded

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.

*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.

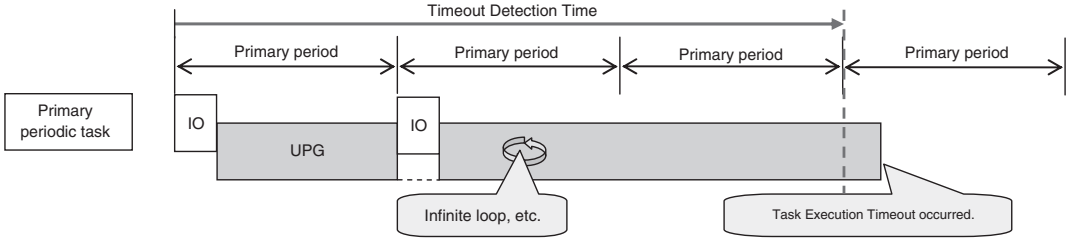
Error name	Error level	Correction
Motion Control Period Exceeded	Partial fault	Reduce the amount of processing in the programs or increase the control period within the range that does not adversely affect operation.

Task Execution Timeout

A Task Execution Timeout error occurs if task processing is not completed within the specified timeout detection 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.

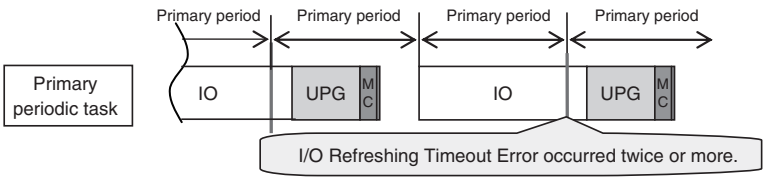


Error name	Error level	Correction
Task Execution Timeout	Major fault	Review the task settings and download the user program again. The power supply must be cycled or the Controller reset.

I/O Refreshing Timeout Error

An I/O Refreshing Timeout Error occurs when I/O refreshing for the primary 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 Controller reset.

5-8 Monitoring Task Execution Status and Task Execution Times

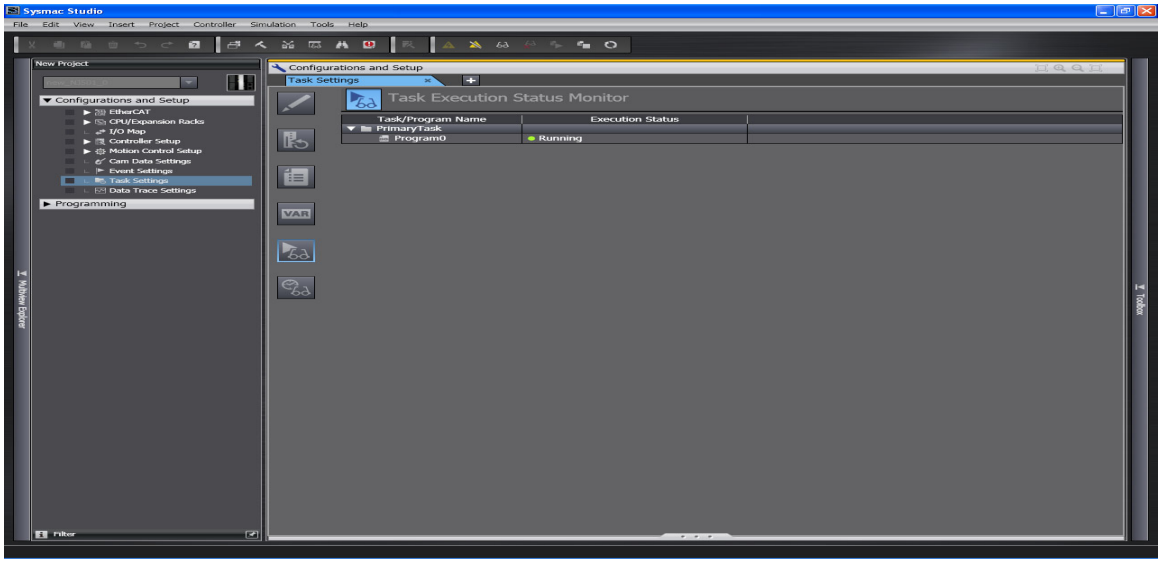
You can use online operations from the Sysmac Studio to monitor the task execution status and task execution times.

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

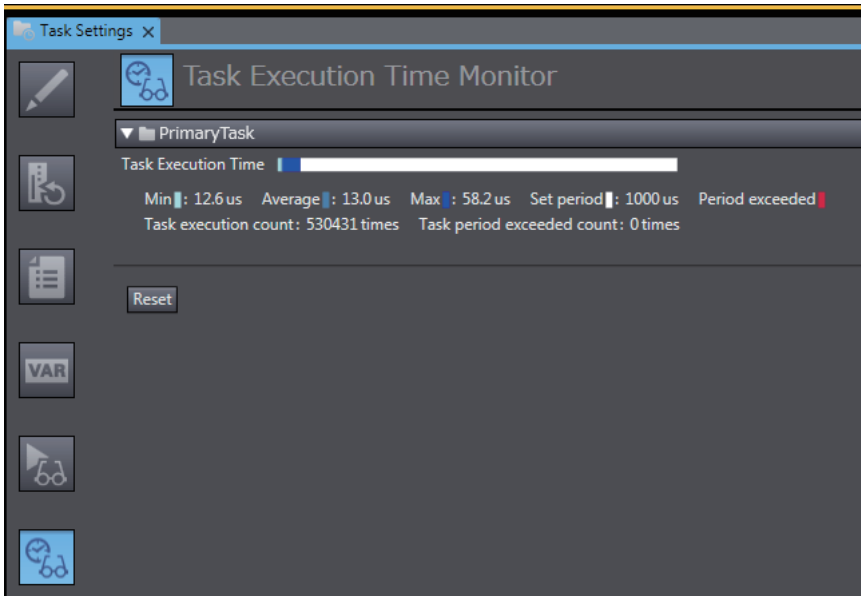
Place the Sysmac Studio online with the Controller and select **Configurations and Setup** – **Task Settings**. Click the **Task Execution Status Monitor** Button to display the following window.



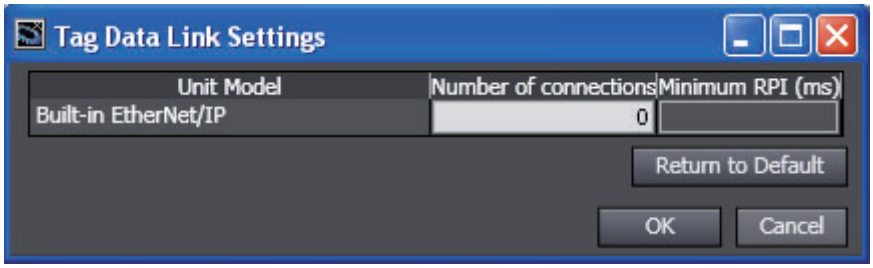
Task Execution Time Monitor

You can monitor the execution time of each task from the Sysmac Studio.

● Values You Can Monitor from the Sysmac Studio Connected to the Controller



Built-in EtherNet/IP Dialog Box for Simulator Connection



The parameters are listed in the following table.

Port that is used	Parameter	Description	Set values	Default
Built-in EtherNet/IP port on NY-series Controller	Number of connections	This is the number of tag data link connections.	0 to 128	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	---

You can monitor the following items.

Monitor item		Description	Connected to the Controller
Task execution time*1	Min.	The minimum value of the task execution time.	Displayed.
	Average	The average value of the task execution time.	
	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.	

*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.

*2 This item is not displayed for event tasks.

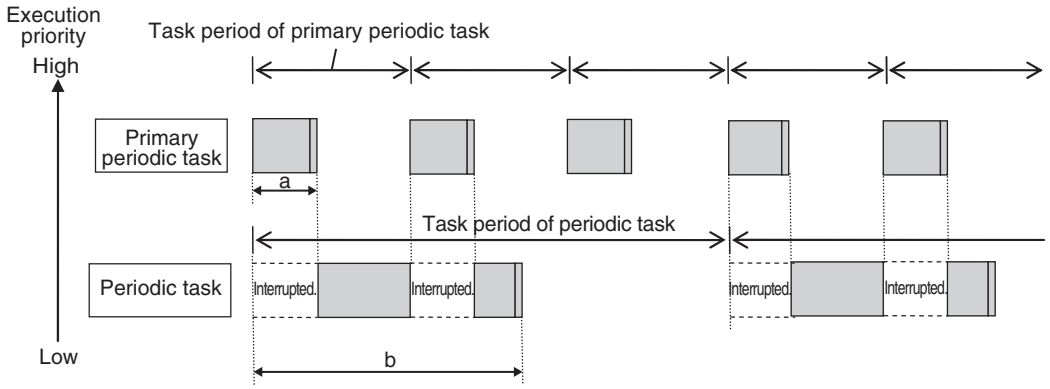
Precautions for Correct Use

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.

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.



a: Task execution time of primary periodic task
b: Task execution time of periodic task

5-9 Task Design Methods and I/O Response Times

This section provides guidelines for designing tasks, information on estimating task execution times, an example of task designing, and information on I/O response times.

The primary periodic task and periodic tasks of an NY-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 Correct 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.

5-9-1 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 NY-series System* to make a rough estimate of the average task execution time on paper.

You cannot estimate the maximum value on paper.

● 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
- Task execution count
- Number of times the task period was exceeded (Task Period Exceeded Count)

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.

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.

5-9-2 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 to estimate the average of the task execution time.
4. 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 the Specified Task Period, Consider Separating It into Periodic Tasks As Follows.

- 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. Refer to the *NY-series Industrial Panel PC / Industrial Box PC Motion Control User's Manual* (Cat. No. W559) for details.
- 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.

5-9-3 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.

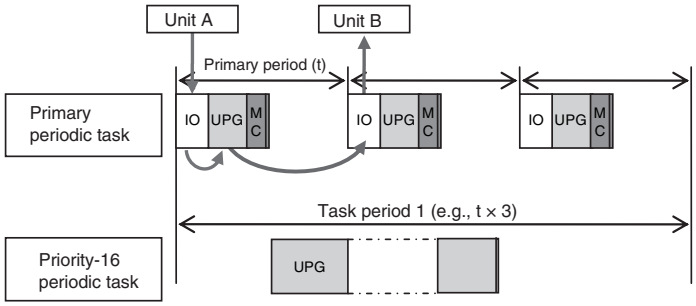
The I/O response times that include EtherCAT communications times are given below.

● 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.

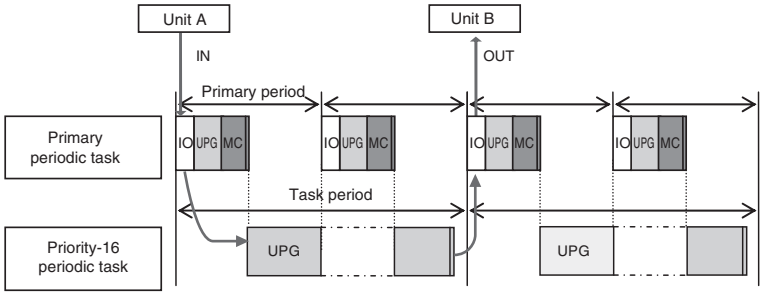
Maximum I/O response time = Primary period × 2

● 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.

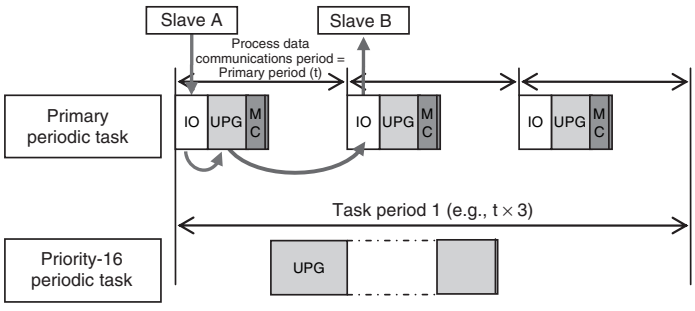
The I/O response times that include EtherCAT communications times are given below.

● 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.

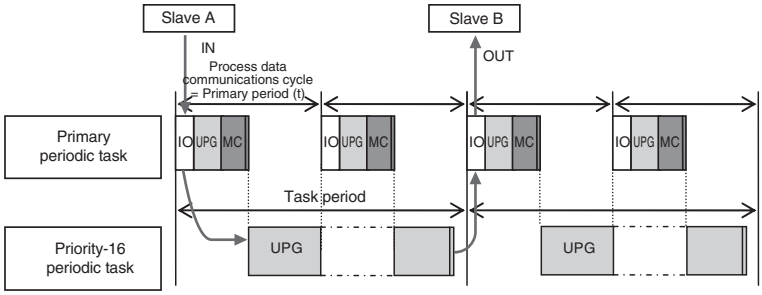
Maximum I/O response time = Primary period (= process data communications cycle) × 2

● Performing Control with the Programs in the Priority-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 Drives and encoder input slaves to which axes are assigned.

For NY-series Controllers, 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.

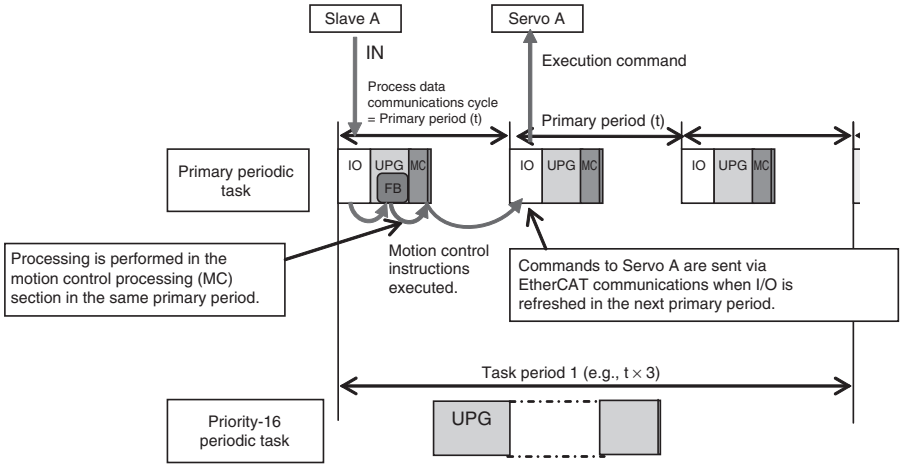
The I/O response times that include EtherCAT communications times are given below.

● 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.

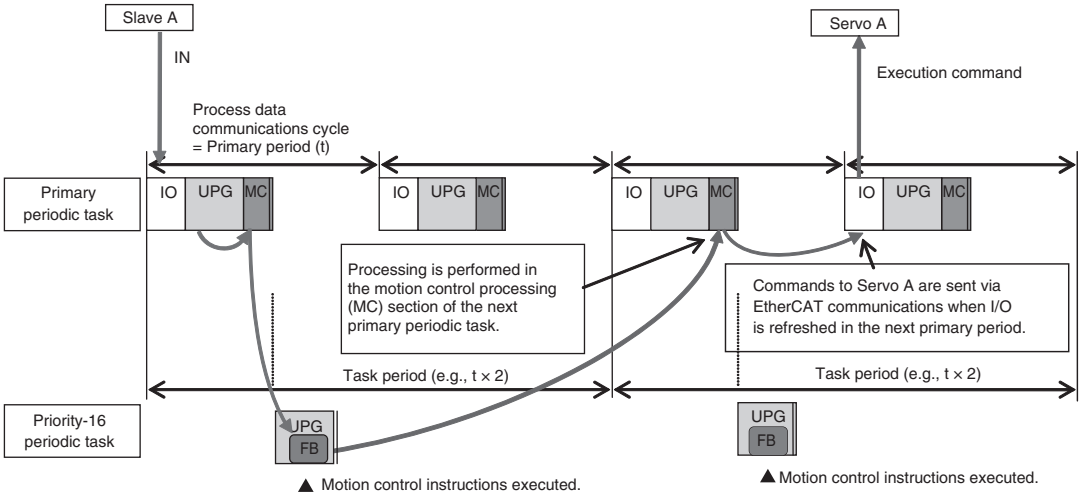
Maximum I/O response time = Primary period (= process data communications cycle) × 2

Programming Motion Control Instructions in the Priority-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)



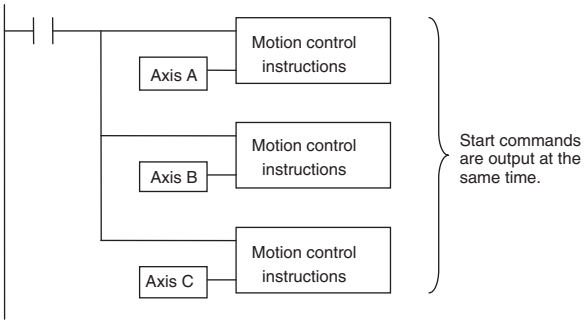
Note: The above diagram shows only one input and one output.

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) x 2

Simultaneous Execution of More Than One Axis

If more than one axis is controlled 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 *NY-series Industrial Panel PC / Industrial Box PC Motion Control User's Manual* (Cat. No. W559).

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 *1-4 Overall Operating Procedure for the NY-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: User I/O allocations • 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 NY-series Controller 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

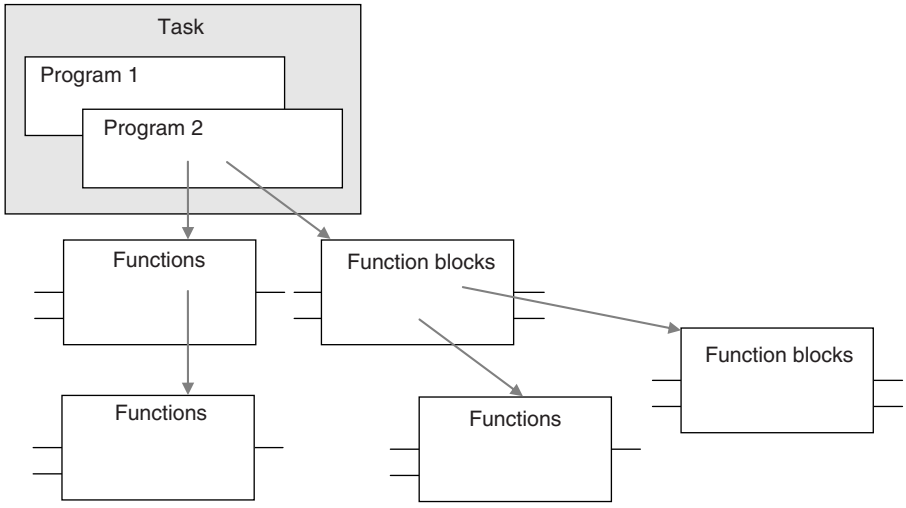
- You cannot use FB instructions or user-defined function blocks in algorithms.
- The values of internal variables are not retained. Therefore, the output value remains constant if the input values are the same.
- There are both user-defined and system-defined function blocks.
  User-defined functions 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.

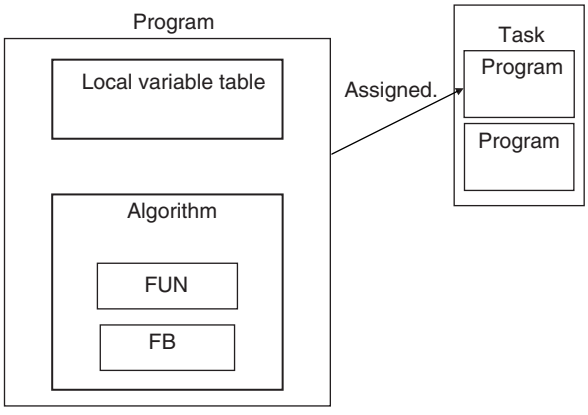
The hierarchical relationships between programs, functions, and function blocks are shown in the following figure.



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_Run-Mode	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 Controller begins operation.	BOOL	Read
P_First_Run	First Program Period Flag	This flag is TRUE for one task period after execution of the program starts.*1 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 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.

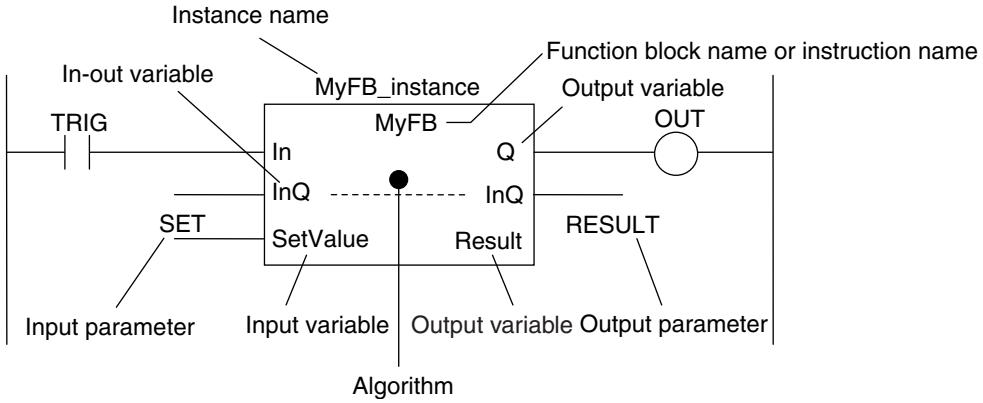
2 Placing an Instance of the Function Block Definition in a Program

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.

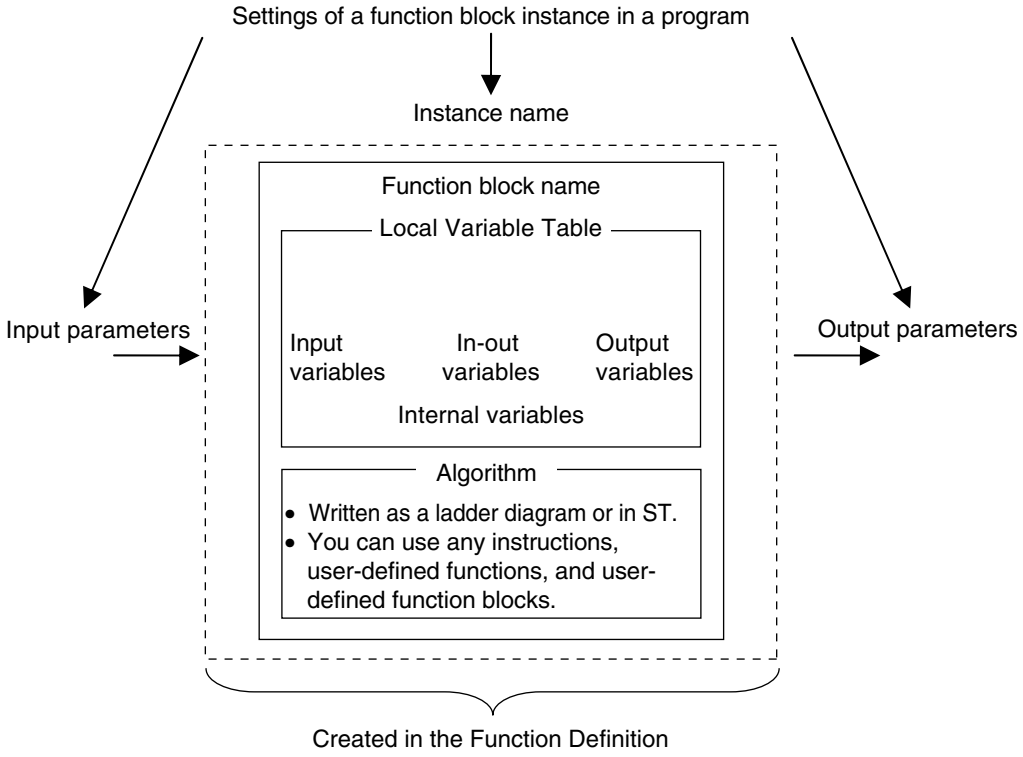
Structure of Function Blocks

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

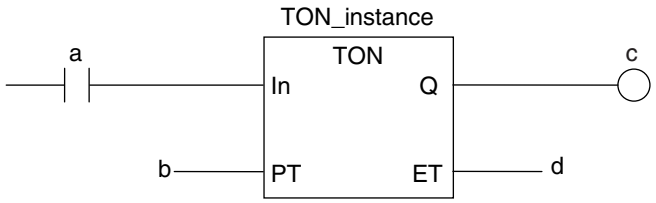
You can omit input and output parameters. Refer to information on operation when parameters are omitted in *Operation When Parameters Are Omitted* on page 6-23 for details.

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.

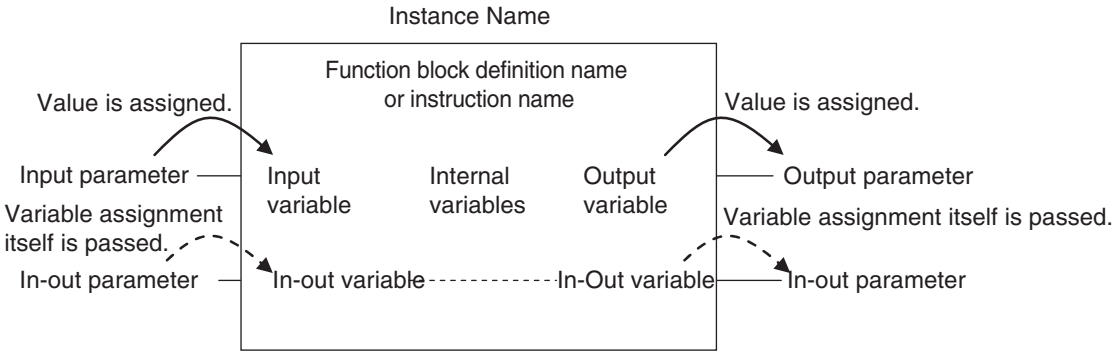


Function Blocks Expressed in ST:

```
   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 6-117 for details.

Variable Designations for Function Blocks



The specifications for variables in function blocks are given below.

Variables	Number	Specification
Input variables	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*.
Output variables[*1]	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.Output-VariableName*. 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 At least one BOOL output variable (including ENO) is required 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.

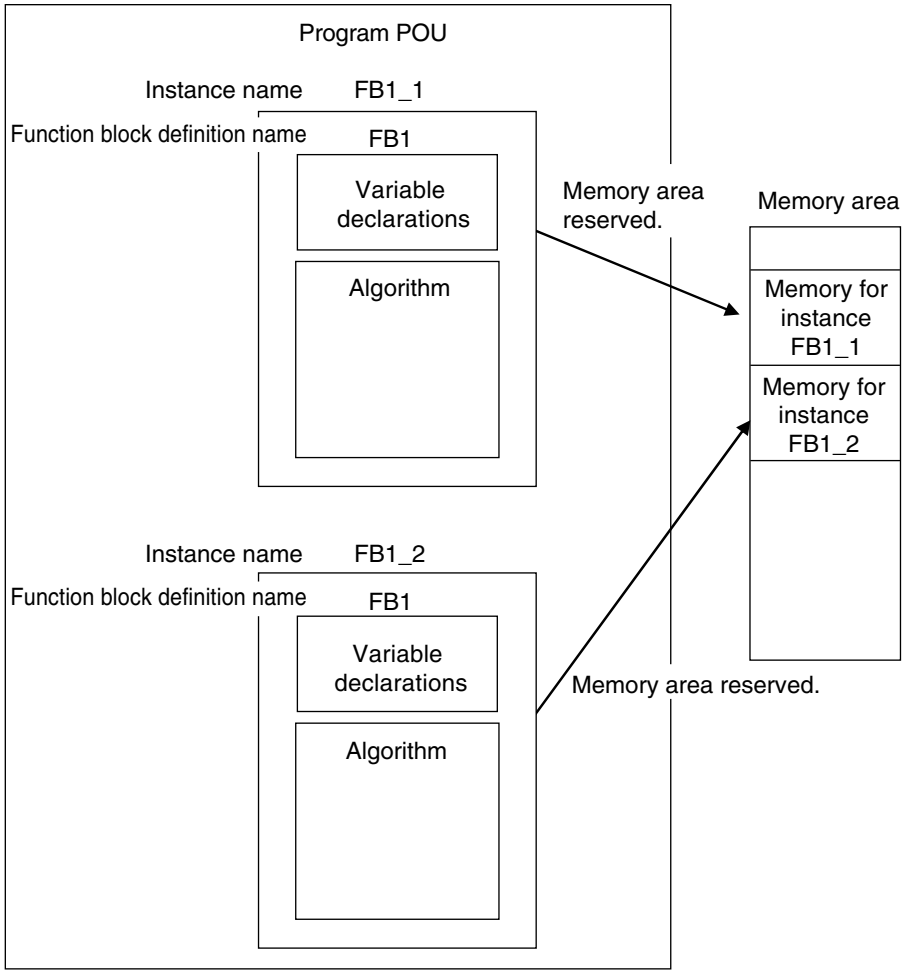
Function Block Definitions and Instances

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.



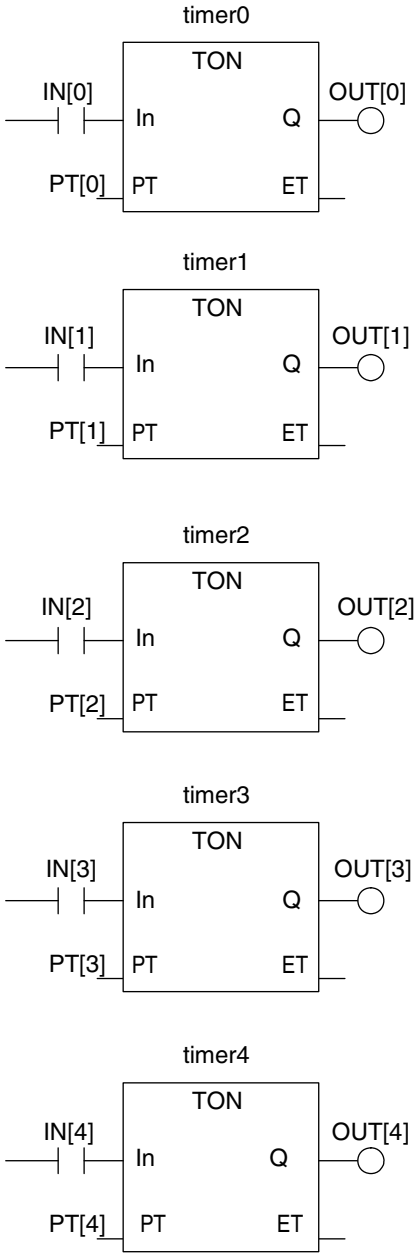
Instances cannot be read from other programs or function blocks. If an instance with the same name as another instance is placed in a different program or another function block, that instance will operate as a completely separate instance.

Array Specifications for Instances

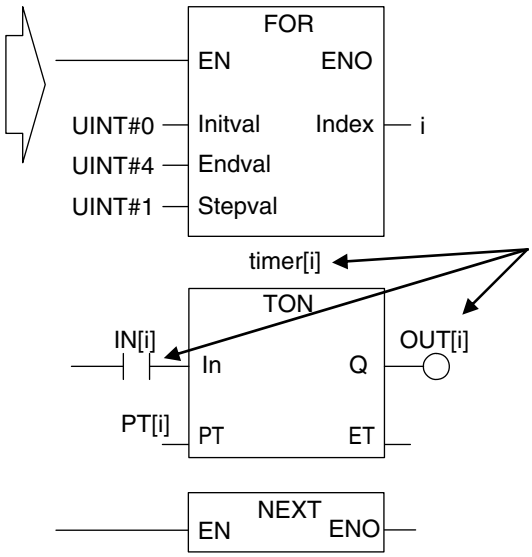
Array specifications can be made for instances. You can indirectly specify an array element number with a variable to execute multiple instances with one instance name. Furthermore, you can switch input sources and output destinations and effectively execute multiple instances with a single instance name if you use an array specification for the input parameter and output parameter and specify the element numbers with the same variable.

Example:

Not Using an Array to Specify Instances



Using an Array to Specify Instances



Here, array variables are used to specify instances of the function block definition TON and all input parameters and output parameters. All of the element numbers are incremented to execute five instances in succession.

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

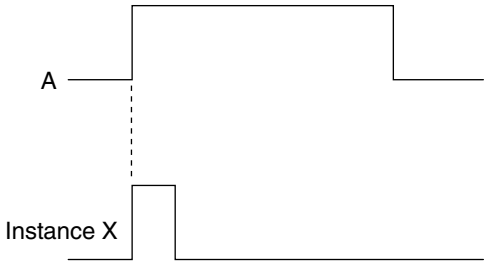
Function blocks do not have an EN input like functions. They are executed each period.

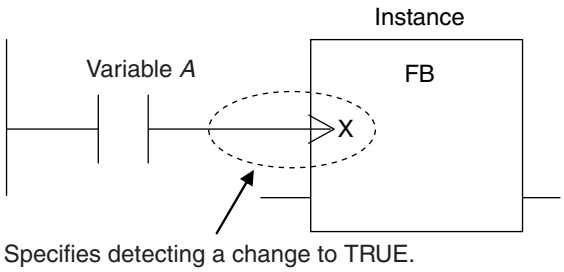
Processes That Require Constant Data Monitoring

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.

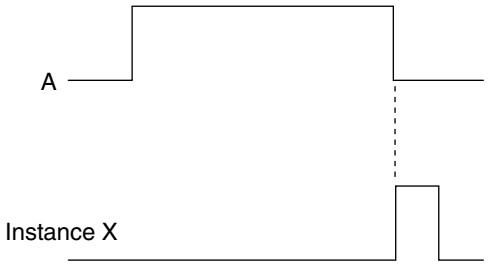
Refer to *6-5-2 Ladder Diagram Language* for details on power flow output and parameter output.

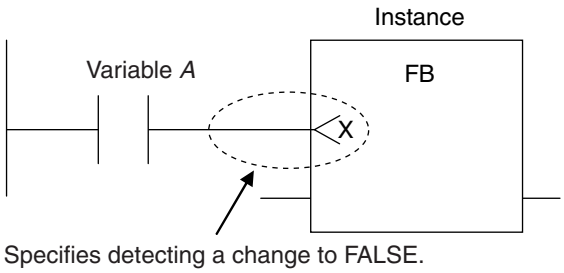
You can specify the edge for an input variable to make the variable TRUE only when the input parameter changes to TRUE.





You can specify falling edges too.





Accessing Variables in a Function Block from Outside the Function Block

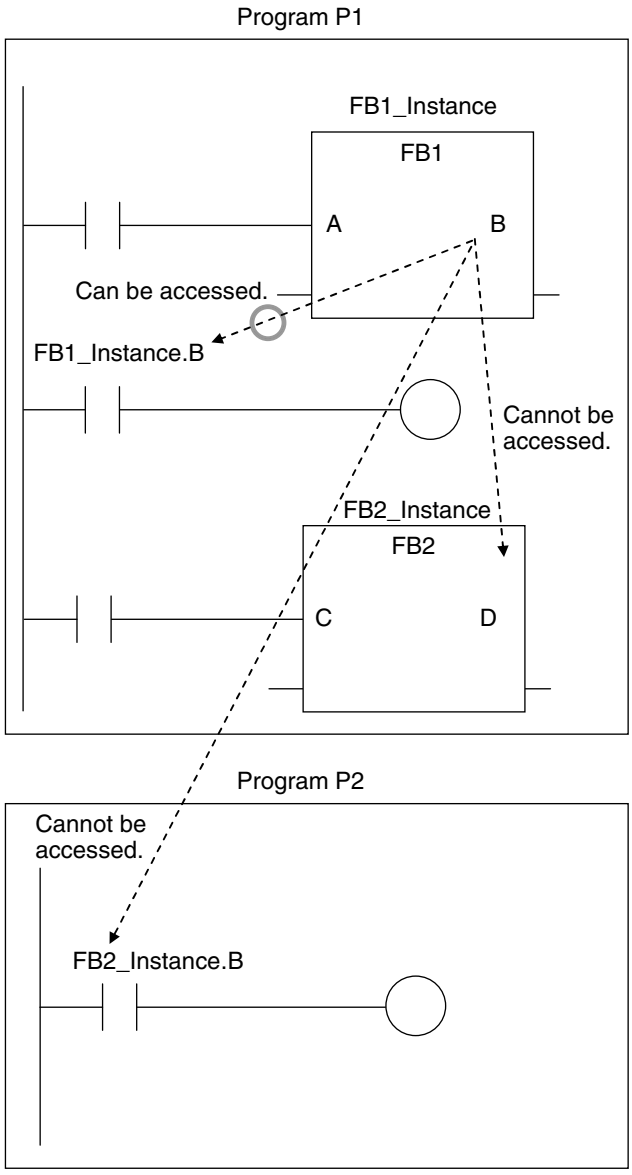
You can access the input and output variables of a function block from outside the function block. Variables are written as follows:

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.



Input and Output Variables for FB1

- Access is possible only from within program P1.
- Access is not possible from program P2.
- Access is not possible from within FB2.

The following variables cannot be accessed from external devices. If these variables are accessed, a building error will occur.

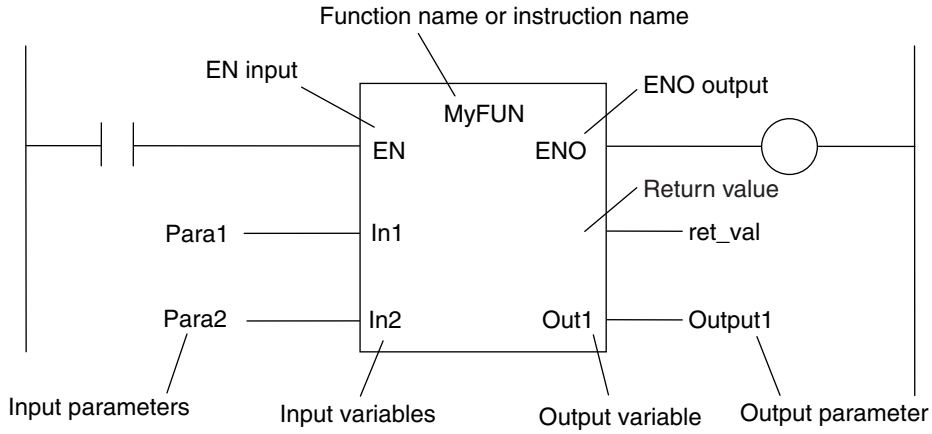
- In-out variables for function blocks
- Input variables for FB instructions for which the default value is not applied if an input parameter is omitted

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.

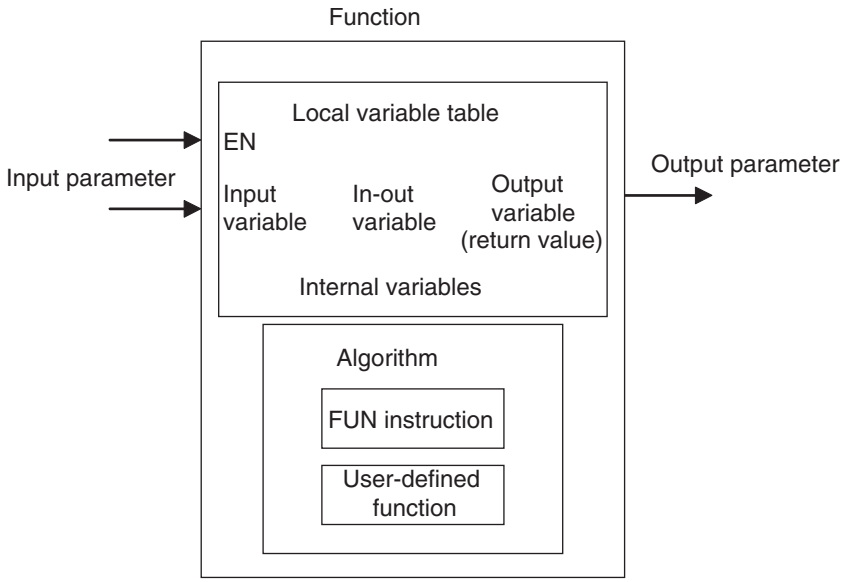
● Instance Name

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

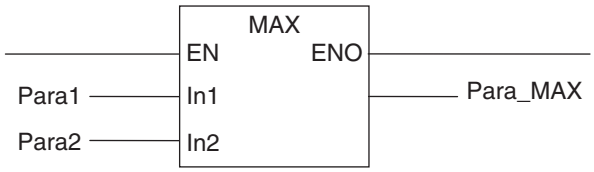
The following example shows how to call functions from ST.

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:

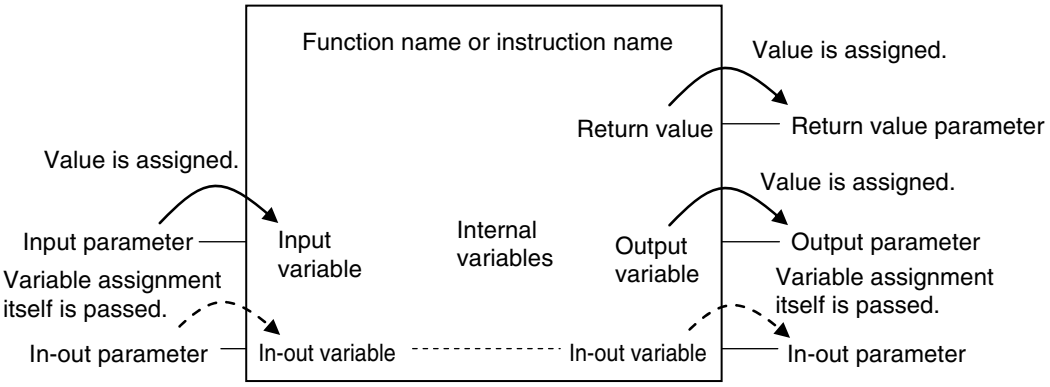


Function name

```
Para_MAX := MAX(In1:=Para1, In2:=Para2);
Para_MAX := MAX(Para1, Para2);              (*The input variables are omitted here.*)
```

Refer to *Function Calls* on page 6-120 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. • The values of the output variables of user-defined functions must always be set in the algorithms of the functions. If the output variables are not set in the algorithms of the functions, the values of the output variables are not stable.
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.

Variables	Number	Specification
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.
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. • The return values of user-defined functions must always be set in the algorithms of the functions. If return values are not set in the algorithms of the functions, the return values are not stable. • Refer to *Return Values* on page 6-21 for details.

Refer to *6-3-4 Attributes of Variables* for details on setting variable attributes.

Additional Information

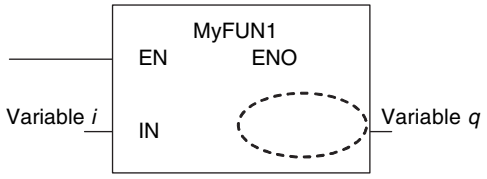
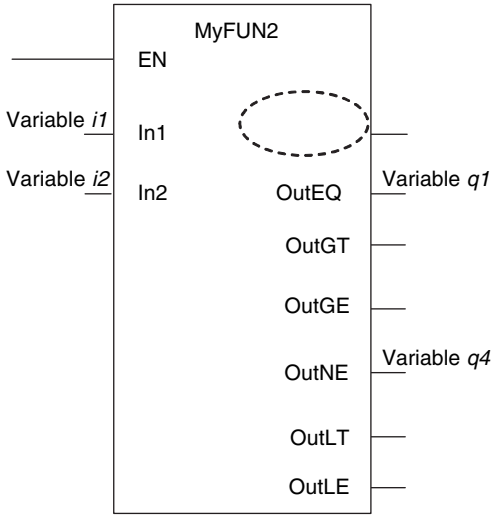
You can register global variables as external variables in a function variable table to access global variables. We recommend that you create your functions so that they produce output values uniquely based on their input parameter values. Algorithms that access global variables and use them to affect 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

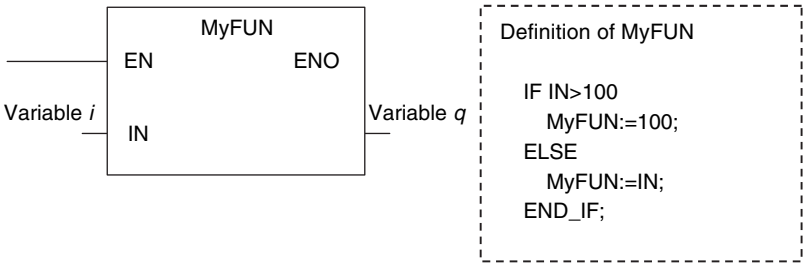
- When *ENO* is FALSE, the previous values of all other output variables are retained.

● 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:=*variable_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*.

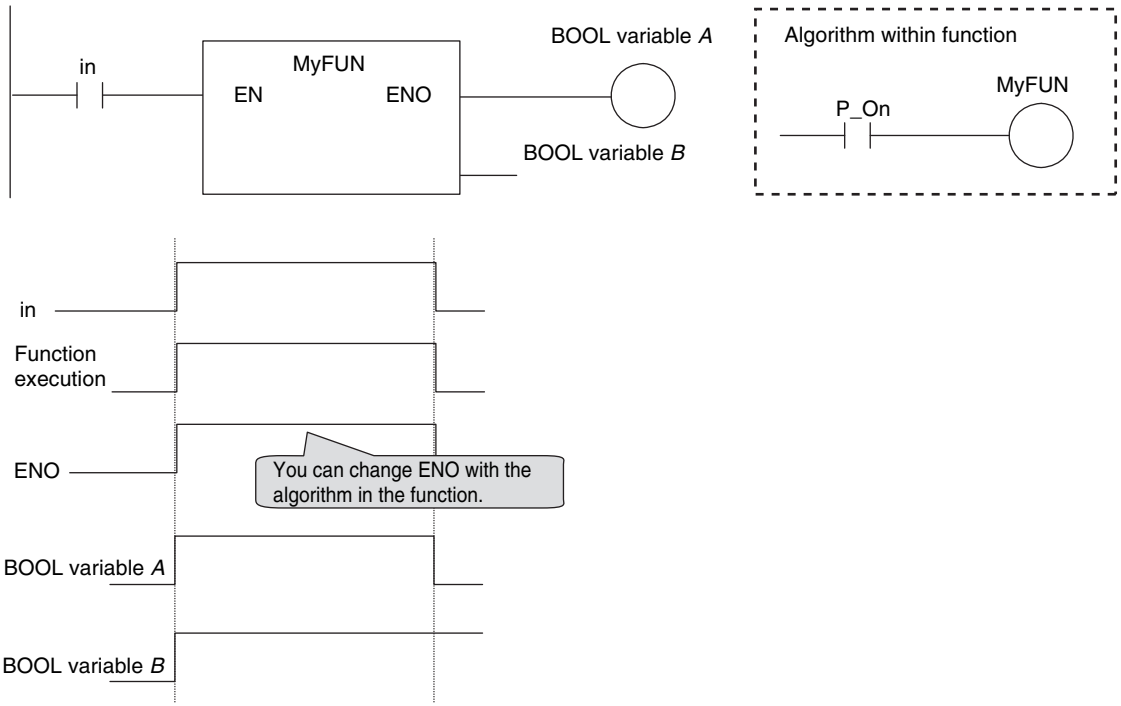


Execution Conditions for Functions

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.

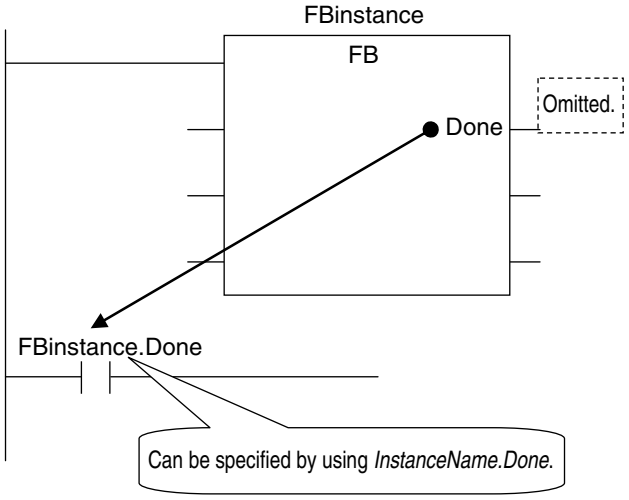
POU		EN	ENO
FB	User-defined functions	Cannot be used. A building error occurs if you try to define *EN* in the variable table from the Sysmac Studio.	Can be used or omitted. You define *ENO* as an output variable in the Sysmac Studio.
	Instruction	All FB instructions do not use *EN*.	Some instructions 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.

	Operation when omitted	
Parameters omitted in	**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).	• *EN* is operated when its value is TRUE. • For other input parameters, 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.



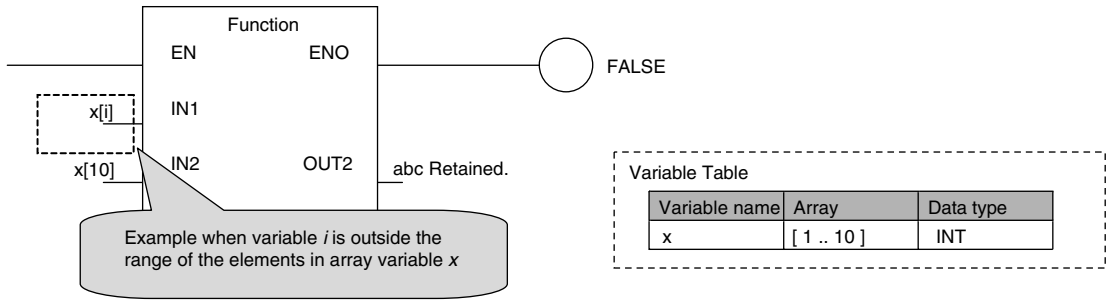
Operation for Parameter Errors

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.

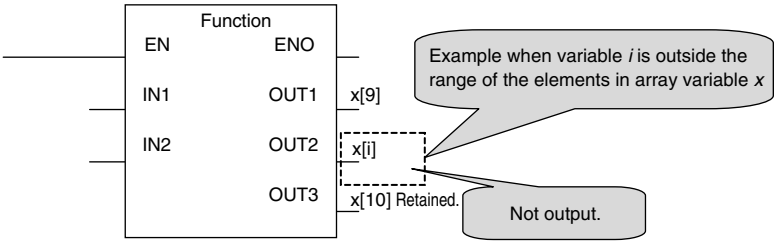
Example:



● Errors in Output Parameters

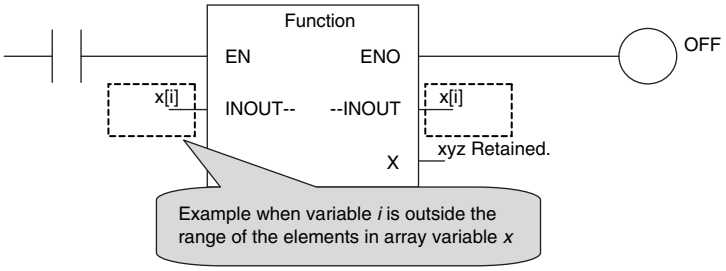
If an error is detected in an output parameter, all values after that parameter are not output but their values are retained.

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

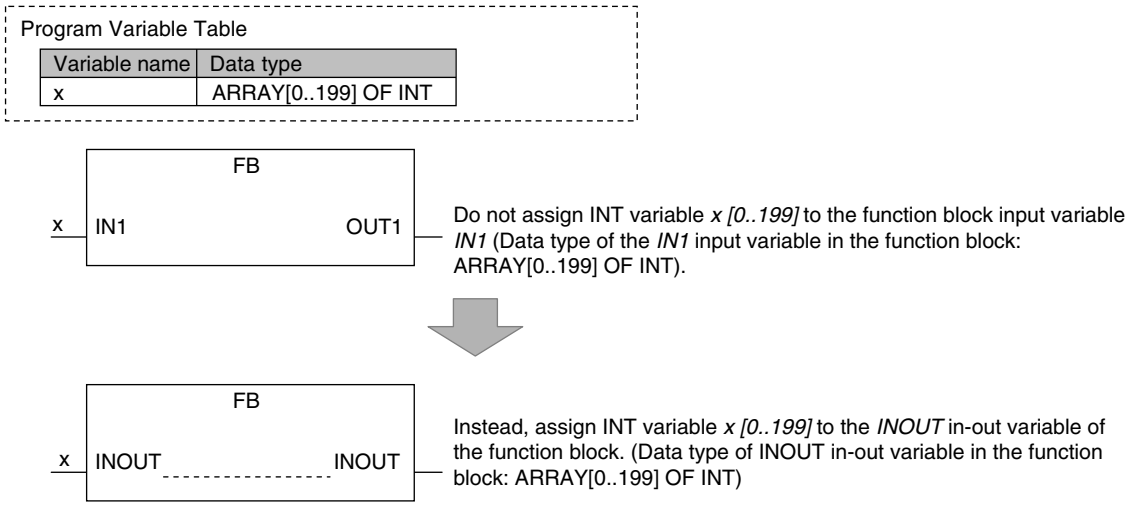
If you need to pass multiple arguments to a function or function block, use an array specification or structure to pass the required data.
This will make your program simpler. However, be aware that if you use an in-out variable, the data passed to the function block or function as a parameter is written and the original data is not retained.

Additional Information

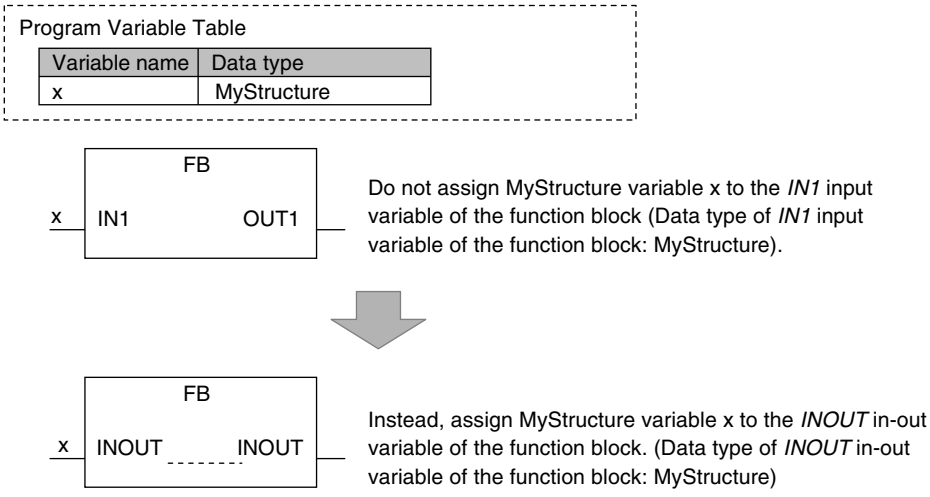
Specifying an Array Variable or Structure Variable as a Parameter

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



Example 2: Specifying a Structure Variable

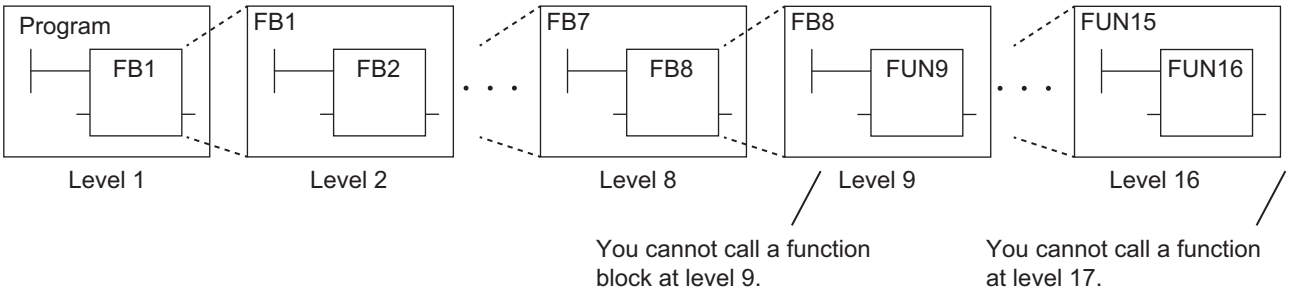


Nesting Levels

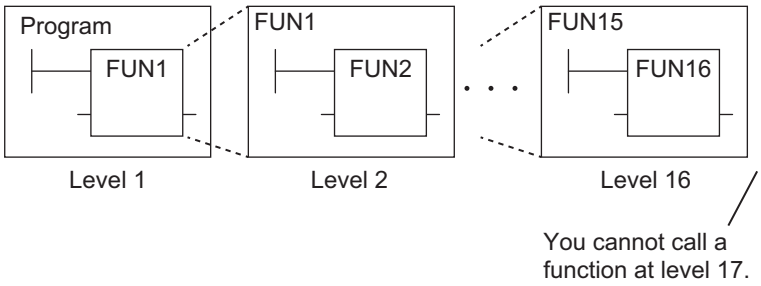
Calling another function or function block from a function or function block that was called from a program is called nesting. The limits that are given in the following table apply to the POUs that you can call from a user-defined function or function block and the number of nesting levels. A building 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.

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 NY-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. The NY-series Controller automatically allocates memory addresses in the memory area for variables.

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 NY-series Controller. 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-3 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 an I/O port for an NX Unit or an EtherCAT slave as a variable, specify the address to assign to that variable.	• Not specified. • Specify.	Not specified.

Attribute	Description	Specification	Default
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.*1 • 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 An AT specification is retained in a non-volatile memory if a UPS is connected to the Industrial PC and the Industrial PC is normally shut down.

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) × 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 *NY-series Instructions Reference Manual* (Cat. No. W560) 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 *NY-series Instructions Reference Manual* (Cat. No. W560) 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]

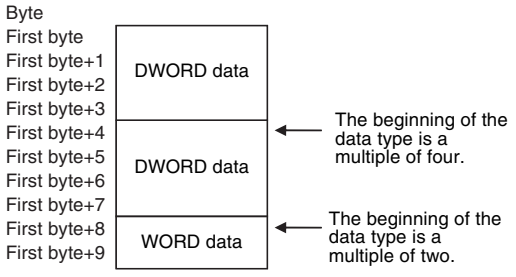
*4 If you want to insert tabs, line break codes, or other special characters, you can use a single-byte dollar sign ($) as an escape character before them. Refer to *Text Strings* on page 6-85 for a list of the escape characters.

Precautions for Correct Use

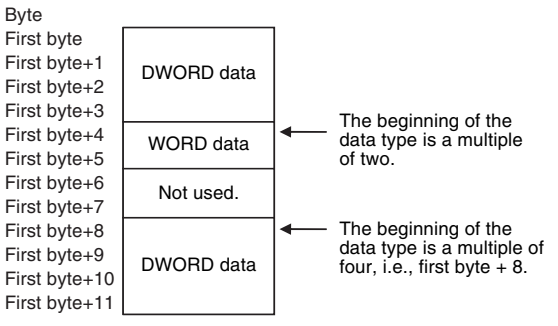
The total amount of memory required by all variables is not equal to the total of the data sizes of each of those variables. This is because the first position where data is stored in memory is automatically set to a multiple of the alignment value for that data type. This results in some empty space 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-6 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.

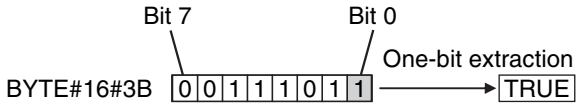
POU type	Type of variable	Unusable data types	
		Basic data types	Derivative data types
Programs	Internal variables	None	
	Global variables	None	
FUN	Input variables, output variables, and in-out variables	None	Unions
	Internal variables	None	
	Return values	None	A structure or union
FB	Input variables, 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

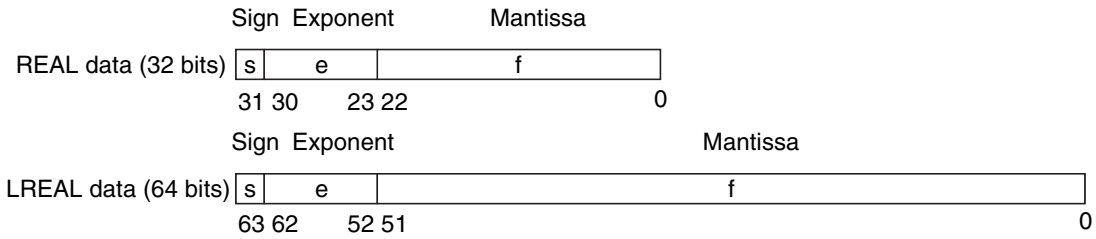
REAL data is 32 bits, while LREAL data is 64 bits.

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.

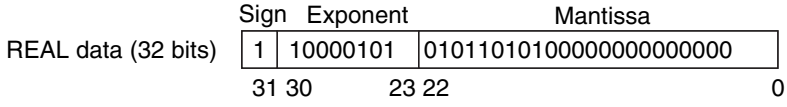
2 86.625 in binary is 1010110.101.

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.

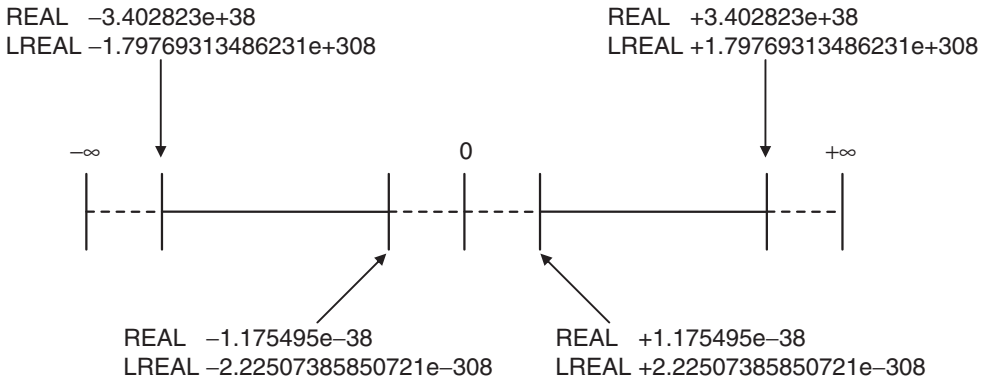
Therefore, you can express −86.625 as shown in the following figure.



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 ≠ 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

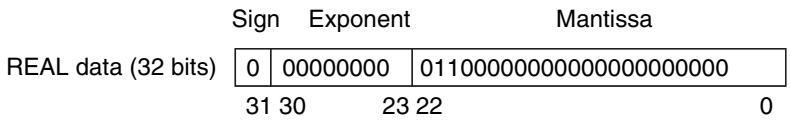
1 This is a positive number, so *s* = 0.

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.



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.

Precautions for Correct Use

Generally, calculation results for real number data may be different if the hardware such as a processor is different. Confirm the calculation results for real number data when you reuse programs and libraries with the different model number of the Controller.

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).

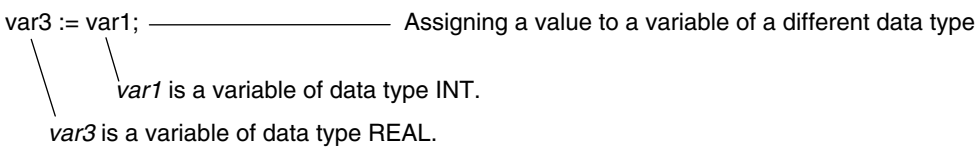
Converting Data Types

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:



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 *NY-series Instructions Reference Manual* (Cat. No. W560) 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

NY-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.)

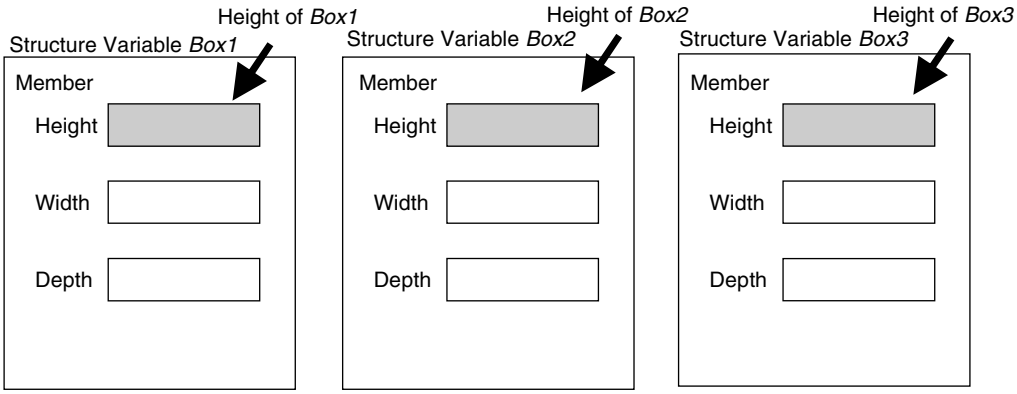
Structures

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.

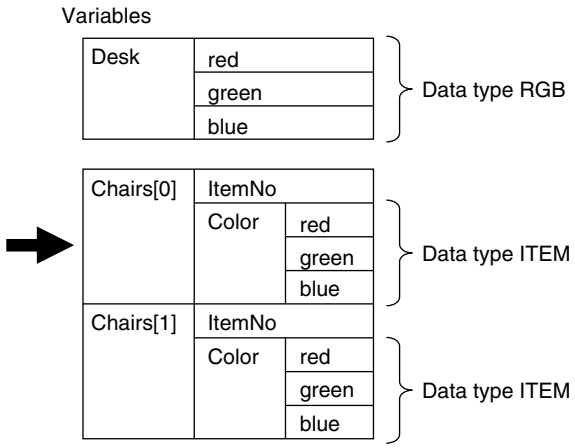
Data Type Table

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	8 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 NY-series is automatically used. Refer to *A-6 Variable Memory Allocation Methods* for details on the memory configuration of NY-series Controllers. Refer to *A-6-2 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 NY-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.

Setting Offsets

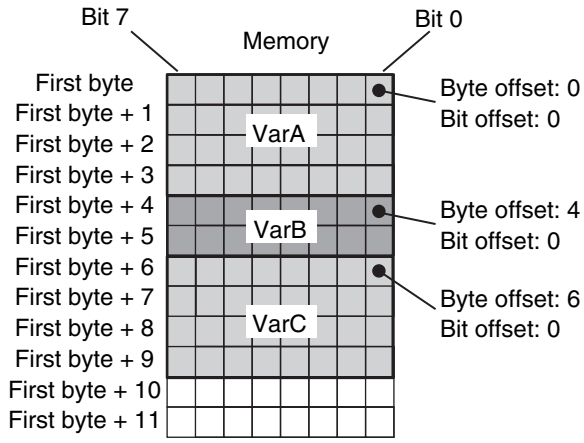
If you set the memory offset type to *User*, you can set memory offsets for each member of the structure. There are byte offsets and bit offsets. If you set the memory offset type to *NJ* or *CJ*, the memory configuration is determined automatically. You do not need to set offsets.
The meanings of the 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 1023
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

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 6-61.
- 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

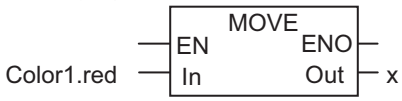
Data Type Table

Name	Member	Data type
RGB	red	UINT
	green	UINT
	blue	UINT

Variable Table

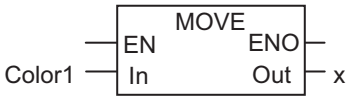
Variable name	Data type
Color1	RGB

■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

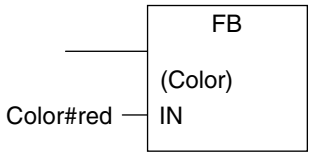
1 Create an enumeration data type in the Enumeration Table.

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

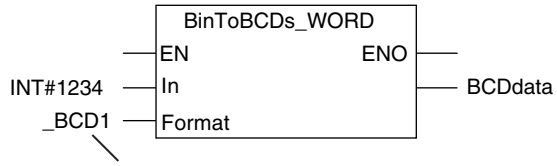
Item	Specification
Enumerator names	Enumerator names consist of single-byte alphanumeric characters. They are not case sensitive. A building error will occur if you specify the same enumerator more than once. A building error will occur if you specify an enumerator with the same name as a variable in 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

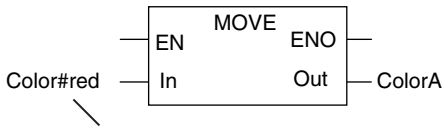
For a function or function block for which the parameter specifies an enumerator, you can just specify the enumerator.
Example: Passing an Enumerator to the BCDToBins_WORD Instruction



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).

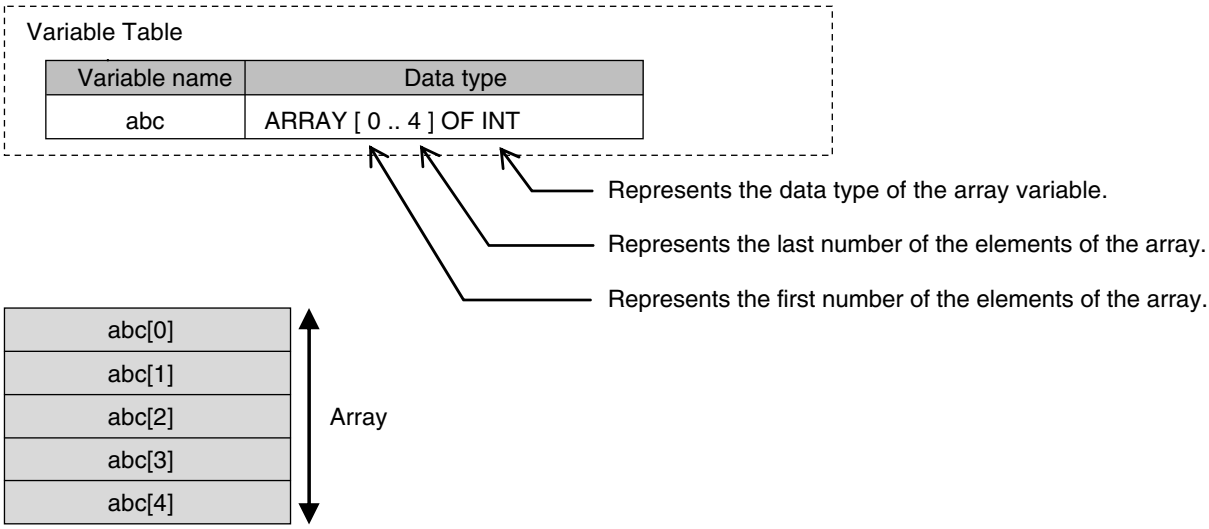
Specifying An Array (i.e., the Entire Array)

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

Item	Specification
Maximum number of elements for an array variable	65,535
Element numbers	0 to 65535 The number for the first element in an array does not have to be 0.
Subscripts	Constants: Integer value between 0 and 65535 Variables: Arithmetic expressions: Arithmetic expressions can be specified only in ST. Example: y:= x[a+b];
Maximum size of one array variable	8 MB

Variables:

Classification	Data type		Usage
Basic data type	Integer	SINT, INT, DINT, USINT, UINT, or UDINT	Supported.
		LINT or ULINT	Not supported.
	Boolean, bit string, real, duration, date, time of day, date and time, or text string data		Not supported.
Derivative data types	Structures, unions, and enumerations		Not supported.
POU instances			Not supported.

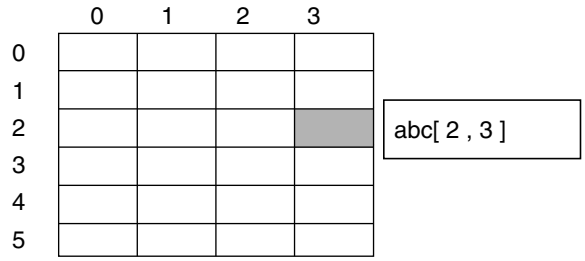
● Dimensions of Array Variables

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

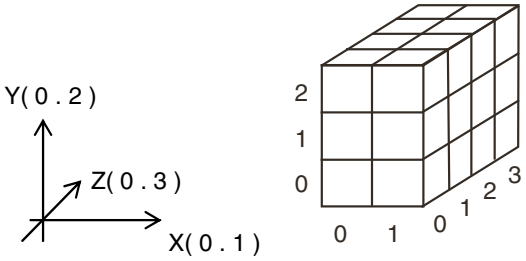
Variable name	Data type
abc	ARRAY [0..5 , 0..3] OF INT



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

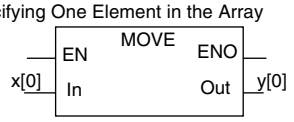
abc[1].x[0]
abc[1].x[1]
abc[1].y
:
abc[4].x[0]
abc[4].x[1]
abc[4].y

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

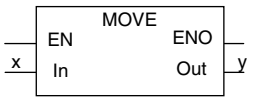


Moves INT variable *x[0]* to *y[0]*

Variable Table

Variable name	Data type
x	ARRAY [0 .. 199]OF INT
y	ARRAY [0 .. 199]OF INT

Specifying the Array

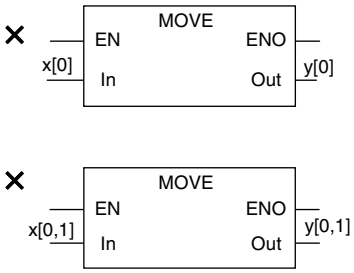


Moves INT variable *x[0..199]* to *y[0..199]*

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

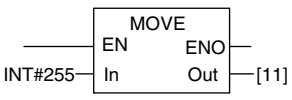
The following errors occur if you attempt to access an element that exceeds the number of elements in an array.

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

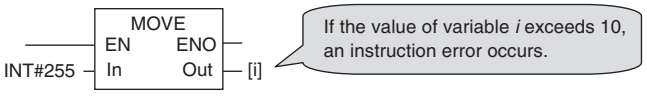


When the Subscript Is a Variable

When an output parameter is assigned to an output variable, the Controller 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

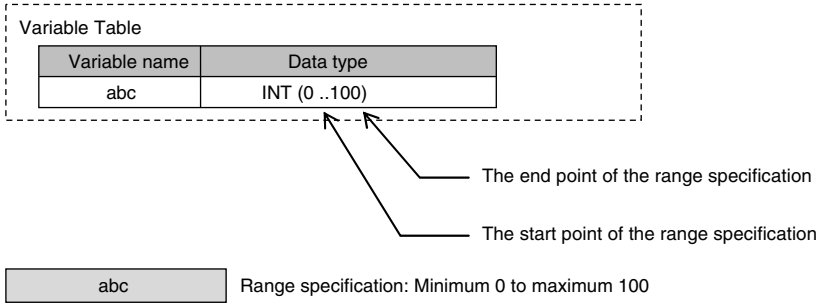
● Making a Range Specification

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

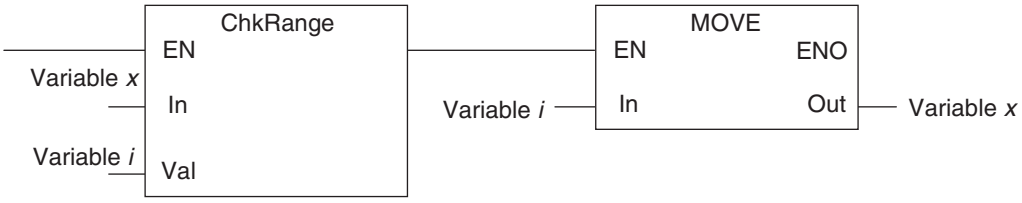
<table>
<tr><th>Item</th><th colspan="4">Specification</th></tr>
<tr><td>Data types that you can specify</td><td colspan="4">Integer type</td></tr>
<tr><td>Operation for attempts to write out-of-range value</td><td colspan="4">
<table>
<tr><th colspan="2">Case</th><th colspan="2">Operation</th></tr>
<tr><td colspan="2">User program</td><td colspan="2">An error does not occur and the value is written. The Controller 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="5">Communications</td><td rowspan="4">Write from the Sysmac Studio or a CIP message</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 structure</td><td rowspan="2">A command error does not occur and the value is written.</td></tr>
<tr><td>For an integer array</td></tr>
<tr><td>Tag data links (both via built-in EtherNet/IP ports and EtherNet/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

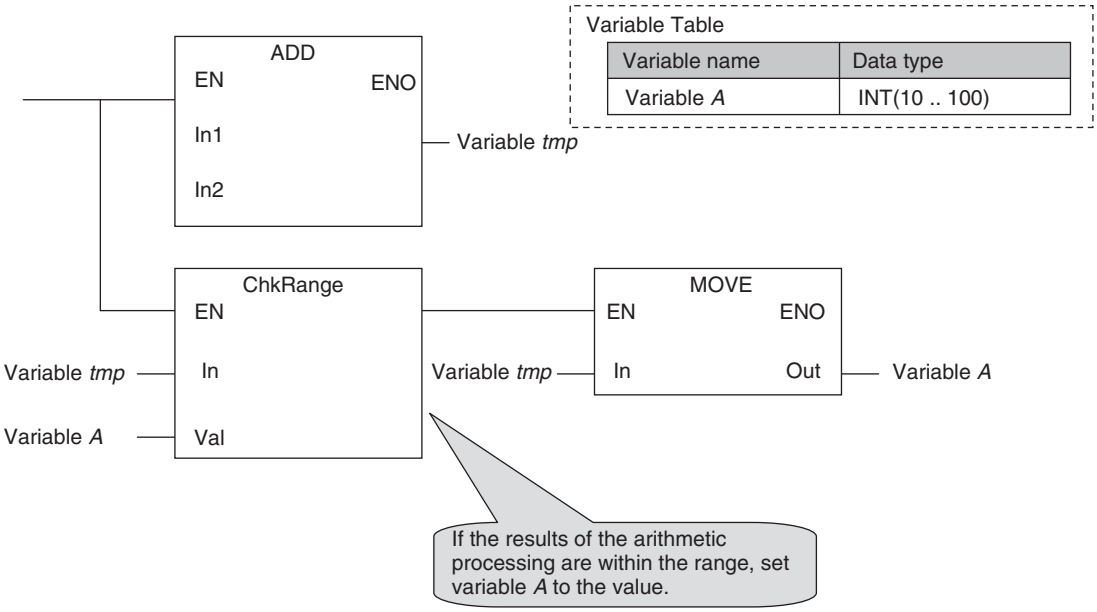
Variables with range specifications are not checked for changes in variable values that result from the execution of instructions in the user program. To check the range of values for a variable that are set from execution of the user program, use instructions that perform range checks.

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).



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 a device variable to an I/O port for the NX Unit or EtherCAT slave. When you assign a variable to an I/O port, they are automatically given an AT specification internally.

● Displaying AT Specifications

An AT specification is displayed in the Allocated Address Box of the variable table or the I/O Map.

Type of variable	Displays in the *AT* field.	Example
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://_MC_AX[1]
Axes Group Variables	MC://_MC_GRP[]	MC://_MC_GR[1]

*1 These system-defined variables are available only for the NX-series CPU Unit.

● 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.	The setting of the Retain attribute agrees with the attribute of the memory where the address is assigned. All of I/O ports for NX Units and EtherCAT slaves are not retained.
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.)

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.

Conditions Required to Enable the Retain Attribute

An AT specification is retained in a non-volatile memory if a UPS is connected to the Industrial PC and the Industrial PC is normally shut down.

Refer to *2-8 Shutdown* on page 2-30 for details on shutdown.

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	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
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

Precautions for Correct Use

If a UPS is not connected to the Industrial PC or the Industrial PC is not normally shut down, 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 Controller to the local Controller).
	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 Controller to another Controller).

● 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

This section describes the changes to the values of variables for status changes of the Controller.

Version Information

With the combination of a Controller with unit version 1.14 or later and Sysmac Studio version 1.18 or higher, you can set the operations when the operating mode changes or when downloading according to the setting of the *_DeviceOutHoldCfg* (Device Output Hold Configuration) system-defined variable.

The values of variables in the Controller will change as shown in the following table when the power is turned ON, when the operating mode changes, or when downloading.

<table>
<tr><th rowspan="2">Retain attribute of variable</th><th colspan="2" rowspan="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>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 colspan="2">Device variables for EtherCAT slaves*2</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 rowspan="2">Retain</td><td rowspan="2">User-defined variables</td><td>Retain condition is met.*3</td><td rowspan="2">No change (retains value before power interruption).</td><td rowspan="2">No change (i.e., the values in RUN mode are retained).</td><td rowspan="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>The values from before the download are retained.</td></tr>
<tr><td>Retain condition is not met.*3</td><td>• 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>
</table>

*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 6-61.

*2 Device outputs are retained even when the operating mode changes or when downloading if the device output hold configuration is set to enable (16#A5A5) in the *_DeviceOutHoldCfg* (Device Output Hold Configuration) system-defined variable. Refer to *Device Output Hold Configurations* on page 6-66 for the device output hold configuration.

*3 Refer to *Retain Condition* on page 6-67 for the retain conditions.

The values of variables in the Controller will change as shown in the following table when a major fault level Controller error occurs, or during online editing.

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	

Device Output Hold Configurations

You can set the operations when the operating mode changes or when downloading according to the setting of the *_DeviceOutHoldCfg* (Device Output Hold Configuration) system-defined variable.

If the device output hold configuration is set to disable (other than 16#A5A5), device outputs will change as given in *6-3-9 Changes to Variables for Status Changes*.

Device outputs are retained even when the operating mode changes or when downloading if the device output hold configuration is set to enable (16#A5A5).

Status	Device output hold configuration	
	Disable (other than 16#A5A5)	Enable (16#A5A5)
When the operating mode is changed	Variables are initialized.	Values of variables are retained.
When downloaded	I/O refreshing is stopped and variables are initialized after downloading.	I/O refreshing is executed and values of variables are retained after downloading.

Device Outputs for Hold Configurations

The following gives the device outputs for device output hold configuration.

- Device variables for EtherCAT slaves

Conditions To Retain Device Outputs When Downloading

There are restrictions on the changeable items as the conditions to retain device outputs when downloading. The following table gives the items for which device outputs can be retained even if items are changed.

Item	
EtherCAT	Backup Parameter Settings of EtherCAT slaves
I/O Map	Variable Name
	Variable Comment
	Variable Type
Event Settings	
Task Settings	Program Assignment Settings
Programming - POUs	Program
	Section
	Internal/ External Variable
	Function
	Function Block
Programming - Data	Data Type
	Global Variable
EtherNet/IP Connection Settings	

If items other than those given in the above table are changed and downloaded, device outputs are not retained.

Also, in the Synchronization Window of the Sysmac Studio, select the check box for the **Do not transfer the following. (All items are not transferred.) - CJ-series Special Unit parameters and EtherCAT slave backup parameters**. If you do not select the check box, device outputs are not retained.

Perform the following procedure if you want to retain device outputs, and change EtherCAT slave backup parameters.

1. Change the user program only and download it.
2. Select the slave in the EtherCAT Tab Page.
3. Click the **Edit Backup Parameter Settings** Button in the Slave Parameter Settings Area on the right of the network configuration.
4. Click the **Transfer to Slave** Button in the **Edit Backup Parameter Settings** Tab Page.

If you execute **Transfer to Slave**, the slave is restarted after the data is transferred. In this case, device outputs of the slave are not retained. However, device outputs are retained for the slave that the data is not transferred.

● 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-4 Specifications for Individual System-defined Variables* for details on system-defined variables.

Variable	Meaning	Function	Data type	R/W
_DeviceOutHoldCfg	Device Output Hold Configuration	It is 16#A5A5 if you retain the target device output when the operating mode is changed or when downloaded. In the case other than 16#A5A5, the target device output is initialized when the operating mode is changed or when downloaded.	WORD	RW
_DeviceOutHoldStatus	Device Output Hold Status	It is TRUE if the target device output is retained when the operating mode is changed or when downloaded. When the device output hold configuration is other than 16#A5A5, or when a major fault level Controller error occurs, the target device output is initialized and changes to FALSE.	BOOL	R

Retain Condition

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) is the same.
- The Retain attribute is set to retain the value of the variable.

Refer to *Values of Retain Variables After New Creations or Changes of POU Names* on page 6-67 for the value of a variable with a Retain attribute after its POU name is changed. Also refer to *Variable Values When Data Types of Retained Variables Are Changed* on page 6-71 for the value of a variable when its data type is changed.

Values of Retain Variables After New Creations or Changes of POU Names

When you download a project that you created with the Sysmac Studio to a Controller, the Controller treats it with a POU name that is different from the POU name displayed on the Sysmac Studio. Even if the POU name is the same as the one displayed on the Sysmac Studio, the Controller may treat it as a different POU.

This section describes how the Controller treats POU names and the values of local variables with a Retain attribute when POU names are changed or newly created on the Sysmac Studio and downloaded.

Before an Operation with the Sysmac Studio

A project, named Poject_A, which was created with the Sysmac Studio is downloaded to a Controller. The Poject_A contains POU_A in which a local variable VarA is contained.

Assume the POU that is displayed as POU_A on the Sysmac Studio is treated as POU_XXX on the Controller. Also, assume the present value of the local variable VarA on the Controller is changed to 20.

On the Sysmac Studio	
Project name:	Project_A
POU name:	POU_A
Variable name:	VarA
Initial value of VarA:	0

Download

On the Controller	
Project name:	Project_A
POU name:	POU_XXX
Variable name:	VarA
Present value of VarA:	20

After Changing a POU Name

The POU name and the present value of the local variable VarA do not change on the Controller even after you change the POU name from POU_A to POU_B on the Sysmac Studio and download it.

On the Sysmac Studio	
Project name:	Project_A
POU name:	POU_B
Variable name:	VarA
Initial value of VarA:	0

Download

On the Controller	
Project name:	Project_A
POU name:	POU_XXX
Variable name:	VarA
Present value of VarA:	20

After Creating a New POU

When you create a new POU, POU_C, for a local variable on the Sysmac Studio and download it, the Controller treats it as a new POU, POU_YYY, and as a new local variable VarA.

Therefore, the value of the new local variable VarA is the initial value, 0.

On the Sysmac Studio	
Project name:	Project_A
POU name:	POU_A
Variable name:	VarA
Initial value of VarA:	0
POU name:	POU_C
Variable name:	VarA
Initial value of VarA:	0

Download

On the Controller	
Project name:	Project_A
POU name:	POU_XXX
Variable name:	VarA
Present value of VarA:	20
POU name:	POU_YYY
Variable name:	VarA
Present value of VarA:	0

After Copying a POU Name

When you copy a POU, POU_A, to create a POU_A_copy on the Sysmac Studio and download it, the Controller treats it as a new POU, POU_ZZZ, and as a new local variable VarA.

Therefore, the value of the new local variable VarA is the initial value, 0.

On the Sysmac Studio	
Project name:	Project_A
POU name:	POU_A
Variable name:	VarA
Initial value of VarA:	0
POU name:	POU_A_copy
Variable name:	VarA
Initial value of VarA:	0

Download

On the Controller	
Project name:	Project_A
POU name:	POU_XXX
Variable name:	VarA
Present value of VarA:	20
POU name:	POU_ZZZ
Variable name:	VarA
Present value of VarA:	0

● When Copying an Exported Project File

The POU name and the present value of the local variable VarA do not change on the Controller even after you copy the Project_A to create the Project_A_copy on the Sysmac Studio and download it.

On the Sysmac Studio	
Project name:	Project_A_copy
POU name:	POU_A
Variable name:	VarA
Initial value of VarA:	0

Download

On the Controller	
Project name:	Project_A_copy
POU name:	POU_XXX
Variable name:	VarA
Present value of VarA:	20

The present value of the local variable VarB does not change on the Controller if you create a local variable VarB that was nonexistent in the Project_A when the Project_A was copied, add VarB to the Project_A and download it while you add a local variable VarB of the same definition to the Project_A_copy and download it.

In the same manner, the present value of the local variable VarB does not change on the Controller if you copy the local VarB from the Project_A or merge the variable table.

On the Sysmac Studio	
Project name:	Project_A_copy
POU name:	POU_A
Variable name:	VarA
Initial value of VarA:	0
Variable name:	VarB
Initial value of VarB:	0

Download

On the Controller	
Project name:	Project_A_copy
POU name:	POU_XXX
Variable name:	VarA
Present value of VarA:	20
Variable name:	VarB
Present value of VarB:	50

● When Adding a POU of the Same Name to Another Project

When you create Project_B which has a POU, POU_A, that is the same name as the POU in the Project_A on the Sysmac Studio and download it, the Controller treats it as a new POU, POU_YYY, and as a new local variable VarA. Therefore, the value of the new local variable VarA is the initial value, 0.

On the Sysmac Studio	
Project name:	Project_B
POU name:	POU_A
Variable name:	VarA
Initial value of VarA:	0

Download

On the Controller	
Project name:	Project_B
POU name:	POU_YYY
Variable name:	VarA
Present value of VarA:	0

When Copying a POU from Another Project to Original Project

When you create Project_B on the Sysmac Studio, copy and past POU_A from the Project_A to Project_B, and download it, the Controller treats it as a new POU, POU_YYY, and as a new local variable VarA. Therefore, the value of the new local variable VarA is the initial value, 0.

On the Sysmac Studio	
Project name:	Project_B
POU name:	POU_A
Variable name:	VarA
Initial value of VarA:	0

Download

On the Controller	
Project name:	Project_B
POU name:	POU_YYY
Variable name:	VarA
Present value of VarA:	0

Variable Values When Data Types of Retained Variables Are Changed

This section describes how the Controller treats the variable values when the data types of variables with a Retain attribute are changed on the Sysmac Studio and the present values of variables with a Retain attribute are restored, transferred, or downloaded.

● Operation and Variable Values

There are two patterns in the way how the variables with a Retain attribute are treated. The table below shows the relationship between the operations and treatment patterns.

Operation	Treatment pattern of variable value
• Restoring with Sysmac Studio Controller backups • Restoring with SD Memory Card backups	Pattern 1
• Restoring with variable and memory backup functions on the Sysmac Studio	Pattern 2
• Downloading from the Sysmac Studio	Pattern 2

The following section gives further description on the treatments of pattern 1 and 2 for each data type.

Changes in Basic Data Types

Assume the variables before changes on the Sysmac Studio are defined as in below.

Variable name	Data type	AT specification
ABC	INT	---
DEF	INT	%D100

Assume the Retain attribute of each variable is set to retain, and the following present values are given.

Variable/memory	Present value
ABC	10
DEF	20

Assume the following present values are given for the memory used for CJ-series Units, DM100 and DM101.

Variable/memory	Present value
D100	20
D101	50

The relationship between changes in the data types and how the controller treats the variables are given below.

Change description	Pattern 1	Pattern 2
No change	The variable value is retained. ABC = 10 DEF = 20	
Changing the variable name from ABC to GHI	The variable value will be the initial value. GHI = Initial value	
Changing the data type of the variable ABC from INT to DINT	The variable value will be the initial value. ABC = Initial value	
Changing the Retain attribute of the variable ABC to Non-retain	The variable value will be the initial value.*1 ABC = Initial value	
Changing the AT specification of the variable DEF from D100 to D101 *2	The variable value will be the value of the AT specification. DEF = 50	
Deleting the AT specification of the variable DEF	The variable value will be the initial value. DEF = Initial value	

*1 The variable value will be the initial value when the Retain attribute of the variable is changed from Non-retain to Retain.

*2 This is an example that the memory used for CJ-series Units is specified for AT specification. You cannot use the memory used for CJ-series Units in the NY-series Controller. Therefore, only use this example as a reference if you change the basic data types.

Changes in a Structure

Assume the structure before changes on the Sysmac Studio is defined as in below.

Data type name	Member	Data type of the member
STR_1	x	INT
	y	INT

Assume the data type of the variable ABC is STR_1, the Retain attribute is set to retain, and the members have the following present values.

Member	Present value
ABC.x	100
ABC.y	-200

The relationship between changes in the data types and how the controller treats the variables are given below.

If the condition applies both to retain and to change to the initial value, the variable value changes to the initial value.

Change description	Pattern 1	Pattern 2
No change	The member value is retained. ABC.x = 100 ABC.y = -200	
Changing the variable name from ABC to DEF	The member value will be the initial value. DEF.x = Initial value DEF.y = Initial value	
Changing the member x to x1	The member value is retained. ABC.x1 = 100 ABC.y = -200	The value of the changed member will be the initial value. ABC.x1 = Initial value ABC.y = -200
Changing the data type of the member x from INT to WORD*1 (Same data size)	The member value is retained. ABC.x = WORD#16#0064 (= 100) ABC.y = -200	
Changing the data type of the member y from INT to UINT*1 (Same data size. Changing from signed to unsigned)	The member value is retained. The absolute value and sign will change because the changed member is directly assigned. ABC.x = 100 ABC.y = 65336	
Changing the data type of the member y from INT to DINT*1 (Expanding data size)	The member value will be the initial value. ABC.x = Initial value ABC.y = Initial value	The member value is retained. The sign of the changed member is expanded while the absolute value and sign do not change. ABC.x = 100 ABC.y = -200
Changing the data type of the member y from INT to UDINT*1 (Expanding data size, and changing from signed to unsigned)	The member value will be the initial value. ABC.x = Initial value ABC.y = Initial value	The member value is retained. The sign of the changed member is expanded and the absolute value and sign change. ABC.x = 100 ABC.y = 4294967096

Change description	Pattern 1	Pattern 2
Changing the data type of the member y from INT to SINT*1 (Decreasing data size)	The member value will be the initial value. ABC.x = Initial value ABC.y = Initial value	The value of the member whose data size is decreased will be the initial value. ABC.x = 100 ABC.y = Initial value
Changing the member x to y and the member y to x	The member value is retained incorrectly. ABC.y = 100 ABC.x = -200	The member value is retained. ABC.y = -200 ABC.x = 100
Adding the member z of INT after the member y	The member value will be the initial value.* 2 ABC.x = Initial value ABC.y = Initial value ABC.z = Initial value	The value of the existing member is retained. The value of the added member will be the initial value. ABC.x = 100 ABC.y = -200 ABC.z = Initial value
Adding the member z of INT between the member x and the member y	The member value will be the initial value.* 2 ABC.x = Initial value ABC.z = Initial value ABC.y = Initial value	The value of the existing member is retained. The value of the added member will be the initial value. ABC.x = 100 ABC.z = Initial value ABC.y = -200
Changing the number of array elements for the member x from 1 to 3	The member value will be the initial value.* 2 ABC.x[0] = Initial value ABC.x[1] = Initial value ABC.x[2] = Initial value ABC.y = Initial value	The value of the existing member is retained. The value of the member whose number of array elements was changed will be retained from the top, and the value of the expanded member will be the initial value. ABC.x[0] = 100 ABC.x[1] = Initial value ABC.x[2] = Initial value ABC.y = -200

*1 Implicit casting applies for changes in data types. Refer to *Implicit Casts* on page 6-123 for details on implicit casting.

*2 You can retain the value of the existing member if you upload the variable value before downloading and then download the variable value after downloading.

Changes in a Union

Assume the union before changes on the Sysmac Studio is defined as in below.

Data type name	Member	Data type of the member
UNI_1	x	WORD
	y	DWORD

Assume the data type of the variable ABC is UNI_1, the Retain attribute is set to retain, and the members have the following present values.

Member	Present value
ABC	WORD#16#0100

The relationship between changes in the data types and how the controller treats the variables are given below.

If the condition applies both to retain and to change to the initial value, the variable value changes to the initial value.

Change description	Pattern 1	Pattern 2
No change	The member value is retained. ABC = WORD#16#0100	
Changing the variable name from ABC to DEF	The member value will be the initial value. DEF = Initial value	
Changing the data type of the member x from WORD to BYTE*1 (Same data size as union)	The member value is retained. ABC = WORD#16#0100	
Changing the data type of the member x from WORD to LWORD*1 (Expanding data size as union)	The member value will be the initial value. ABC = Initial value	The member value is retained. ABC = WORD#16#0100
Changing the data type of the member y from DWORD to BYTE*1 (Decreasing data size as union)	The member value will be the initial value. ABC = Initial value	
Changing the member x to y and the member y to x	The member value is retained. ABC = WORD#16#0100	
Adding the member z of BYTE after the member y (Same data size as union)	The member value is retained. ABC = WORD#16#0100	
Adding the member z of LWORD after the member y (Expanding data size as union)	The member value will be the initial value. ABC = Initial value	The member value is retained. ABC = WORD#16#0100
Adding the member z of BYTE after the member y, and deleting the member y. (Decreasing data size as union)	The member value will be the initial value. ABC = Initial value	

*1 Implicit casting applies for changes in data types. Refer to *Implicit Casts* on page 6-123 for details on implicit casting.

Changes in Array Variables

Assume the array variables before changes on the Sysmac Studio are defined as in below.

Variable name	Data type
ABC	ARRAY[1..3] OF INT

Assume the Retain attribute of the variable ABC is set to retain, and the elements have the following present values.

Member	Present value
ABC[1]	100
ABC[2]	200
ABC[3]	300

The relationship between changes in the data types and how the controller treats the variables are given below.

If the condition applies both to retain and to change to the initial value, the variable value changes to the initial value.

Change description	Pattern 1	Pattern 2
No change	The element value is retained. ABC[1] = 100 ABC[2] = 200 ABC[3] = 300	
Changing the variable name from ABC to DEF	The element value will be the initial value. DEF[1] = Initial value DEF[2] = Initial value DEF[3] = Initial value	
Changing the data type of the element from INT to UINT*1 (Same data size)	The element value will be the initial value. ABC[1] = Initial value ABC[2] = Initial value ABC[3] = Initial value	
Changing the data type of the element from INT to DINT*1 (Expanding data size)	The element value will be the initial value. ABC[1] = Initial value ABC[2] = Initial value ABC[3] = Initial value	
Changing the data type of the element from INT to SINT*1 (Decreasing data size)	The element value will be the initial value. ABC[1] = Initial value ABC[2] = Initial value ABC[3] = Initial value	
Changing the last number of the array element from 3 to 4*2 (Expanding the number of elements from 3 to 4)	The element value will be the initial value. ABC[1] = Initial value ABC[2] = Initial value ABC[3] = Initial value ABC[4] = Initial value	The values of the existing element numbers are retained. The values of the new element numbers will be the initial values. ABC[1] = 100 ABC[2] = 200 ABC[3] = 300 ABC[4] = Initial value

Change description	Pattern 1	Pattern 2
Changing the first number of the array element from 1 to 0*2 (Expanding the number of elements from 3 to 4)	The element value will be the initial value. ABC[0] = Initial value ABC[1] = Initial value ABC[2] = Initial value ABC[3] = Initial value	The values of the existing element numbers are retained. The values of the new element numbers will be the initial values. ABC[0] = Initial value ABC[1] = 100 ABC[2] = 200 ABC[3] = 300
Changing the last number of the array element from 3 to 2*2 (Decreasing the number of elements from 3 to 2)	The element value will be the initial value. ABC[1] = Initial value ABC[2] = Initial value	The values of the existing element numbers are retained. ABC[1] = 100 ABC[2] = 200
Changing the first number of the array element from 1 to 2*2 (Decreasing the number of elements from 3 to 2)	The element value will be the initial value. ABC[2] = Initial value ABC[3] = Initial value	The values of the existing element numbers are retained. ABC[2] = 200 ABC[3] = 300
Changing the first number of the array element from 1 to 3 and the last number from 3 to 5*2 (Same number of elements)	If the number of elements is the same, the values of the elements are retained from the top.*3 ABC[3] = 100 ABC[4] = 200 ABC[5] = 300	The values of the existing element numbers are retained. The values of the new element numbers will be the initial values. ABC[3] = 300 ABC[4] = Initial value ABC[5] = Initial value

*1 Implicit casting applies for changes in data types. Refer to *Implicit Casts* on page 6-123 for details on implicit casting.

*2 The result will be the same for basic and derivative data types.

*3 If the number of elements is expanded or decreased, all values of the elements will be the initial values.

Changes in an Enumeration

Assume the enumeration before changes on the Sysmac Studio are defined as in below.

Data type name	Enumerator	Value
ENU_1	x	1
	y	2

Assume the data type of the variable ABC is ENU_1, the Retain attribute is set to retain, and the variable has the following present value.

Member	Present value
ABC	1 (= x)

The relationship between changes in the data types and how the controller treats the variables are given below.

Change description	Pattern 1	Pattern 2
No change	The variable value is retained. ABC = 1 (= x)	
Changing the variable name from ABC to DEF	The variable value will be the initial value. ABC = Initial value	
Adding the enumerator z whose value is 3 after the enumerator y	The variable value is retained. ABC = 1 (= x)	
Deleting the enumerator x	The variable value is retained even if the enumerator does not exist. ABC = 1	
Changing the value of the enumerator x from 1 to 5	The variable value is retained even after the enumerator value is changed. ABC = 1	

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 compatibility</th><th>Case sensitivity</th><th>Maximum size (not including NULL)</th><th>Character encoding</th></tr>
<tr><td>Variable name (including POU instance names)</td><td rowspan="14">● Usable characters
• 0 to 9, A to Z, and a to z
• Single-byte kana
• _ (underlines)
• Multibyte characters (e.g., Japanese)
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 Unicode space characters)</td><td rowspan="14">Refer to Reserved Words below.</td><td rowspan="14">Supported.</td><td rowspan="12">Not case sensitive.</td><td rowspan="2">127 bytes</td><td rowspan="14">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 variable names</td><td>Network variable: 255 bytes
Other: 511 bytes</td></tr>
<tr><td>Device names</td><td rowspan="3">127 bytes</td></tr>
<tr><td>Section names</td><td>Case sensitive.</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 *NY-series Instructions Reference Manual* (Cat. No. W560)
- A name that is the same as any of the instructions that are described in *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561)
- 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 Controller
- 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 NY-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	

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}	16#0064
*{numeric_value}**1	100

*1 A base of 10 (i.e., a decimal number) is assumed.

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}	10#–1
*{data_type_name}#{numeric_value}**1	INT#–1
*{numeric_value}**1	–1

*1 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}	10#–3.14
*{data_type_name}#{numeric_value}**1	LREAL#–3.14
*{numeric_value}**1	–3.14

*1 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}*'	STRING#'This is a 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 NY-series Controller: ladder diagram language (LD) and ST (structured text) language.

6-5-2 Ladder Diagram Language

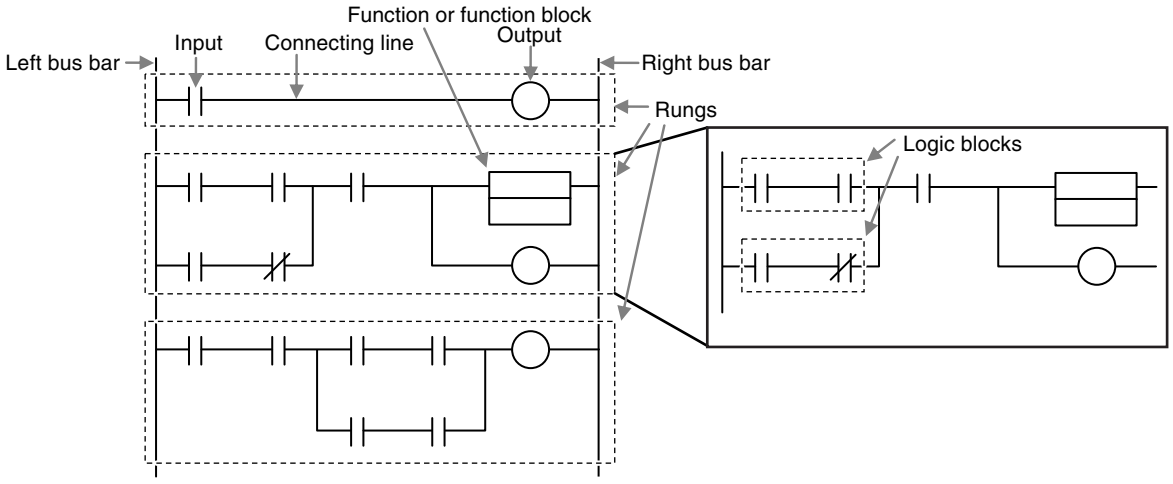
The ladder diagram language (LD) is a graphical programming language that is written in a form that appears similar to electrical circuits. Each object for processing, including functions and function blocks, is represented as a diagram. Those objects are connected together with lines to build the algorithm. 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 *NY-series Instructions Reference Manual* (Cat. No. W560).

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 *NY-series Instructions Reference Manual* (Cat. No. W560).

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 *NY-series Instructions Reference Manual* (Cat. No. W560).

Order of Execution for Ladder Diagrams

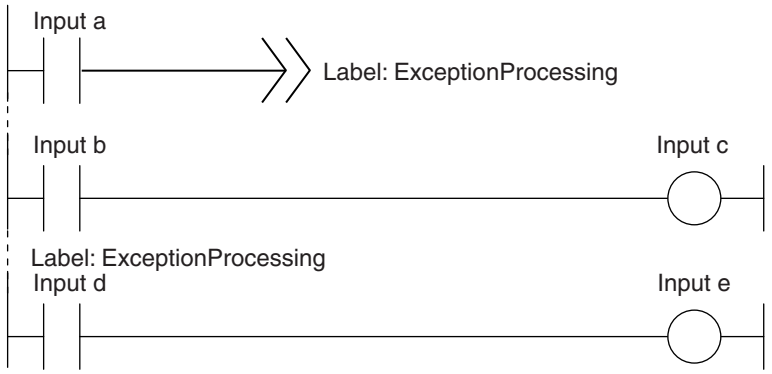
Inputs, outputs, functions, and function blocks are executed when they receive the power flow. The order of execution for a ladder diagram is from top to bottom. Elements at the same level are executed from left to right.

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

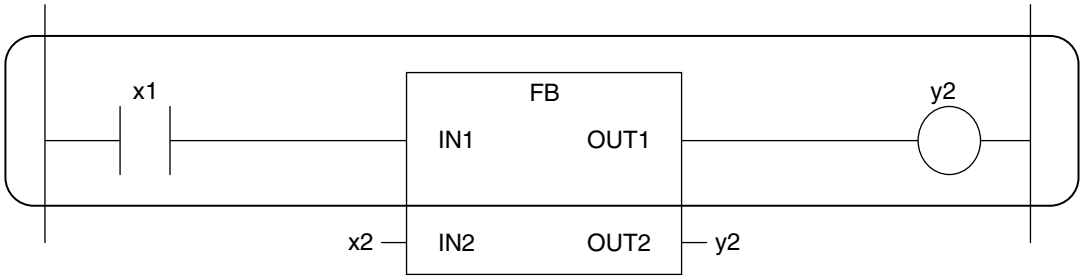
You use the following two types of connections for functions or function blocks.

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.

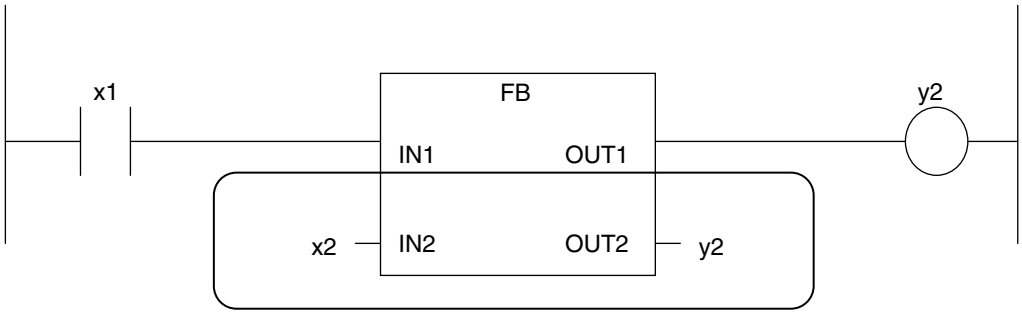
Example:

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.



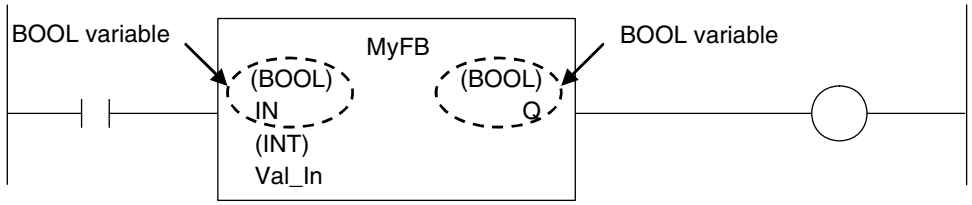
As shown below, you can specify either variables or constants for input and output parameters.

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

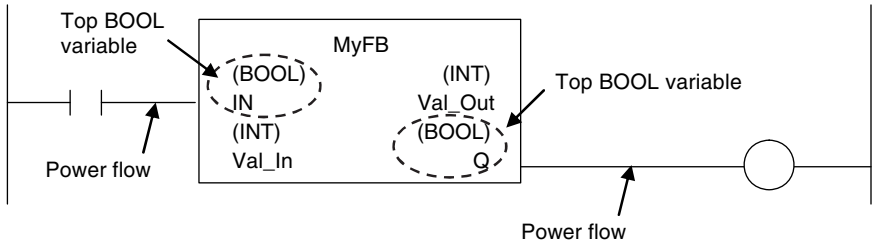
At least one BOOL variable each is required for the input and the output (such as EN and ENO) of a function or function block.

Example:

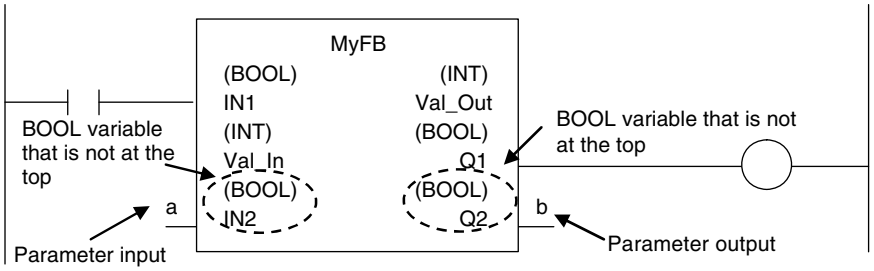


Connections Based on the BOOL Variable Positions

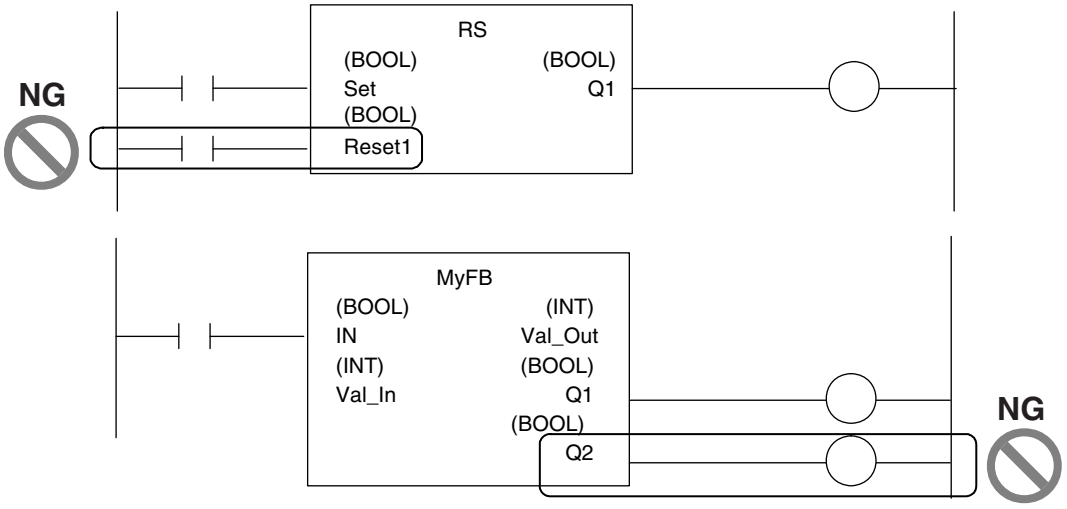
The top BOOL variables are connected to the left and right bus bars. In other words, they become the power flow input and power flow output.



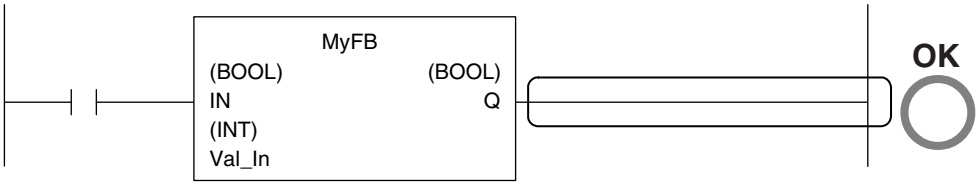
There is only one power flow input and one power flow output for each function or function block. All other BOOL variables that are not at the top are for parameter inputs and parameter outputs.



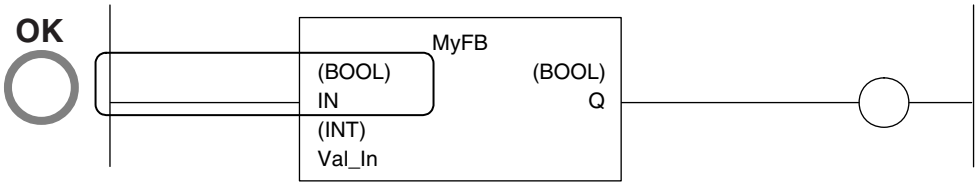
You cannot connect multiple BOOL variables to the left bus bar or the right bus bar as shown below.



You do not have to connect an OUT instruction to the right bus bar. You can connect the function or function block directly.



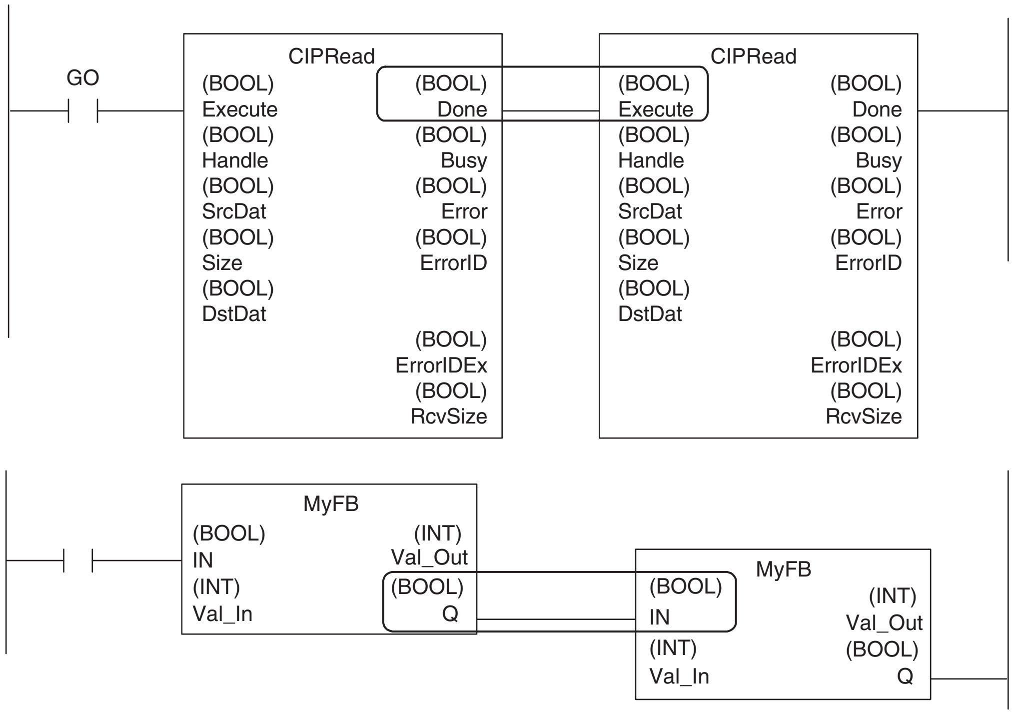
A LD instruction is not necessarily required. You can also connect directly to the left bus bar.



● Cascade Connections

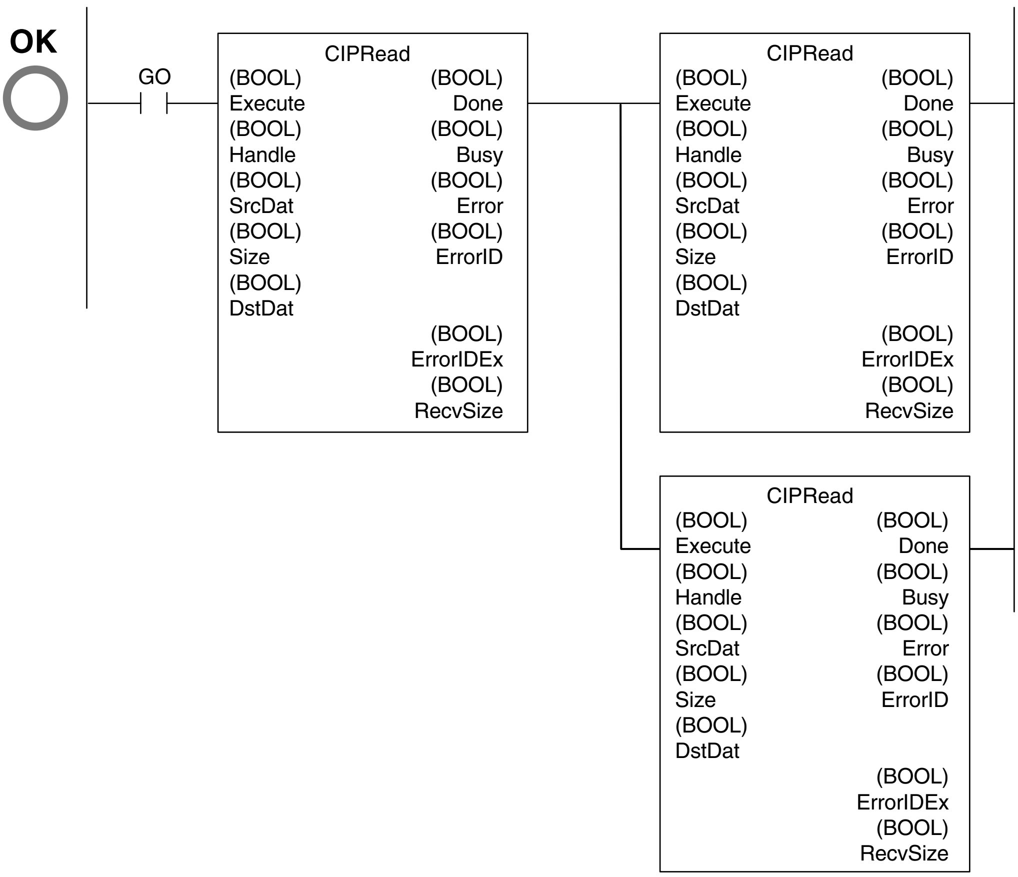
Cascade connections in which the output of a function or function block is connected to the input of another function or function block are allowed only for power flow outputs and inputs.

Example:



You can branch the power flow output.

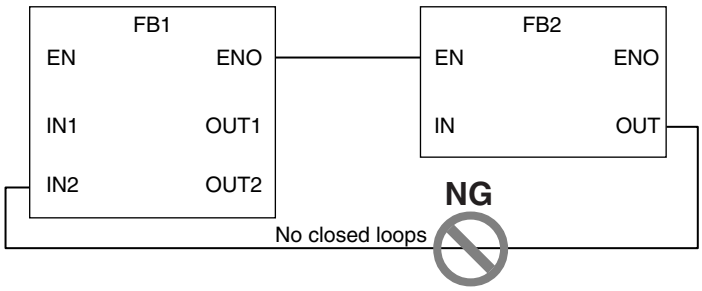
Example:



Restriction

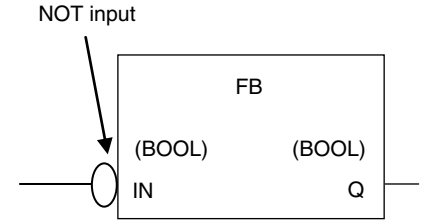
- You cannot create closed loops or intersect connecting lines.

Example:



● Reversing Inputs

You can reverse the value of a BOOL input variable when you input it to an instruction.



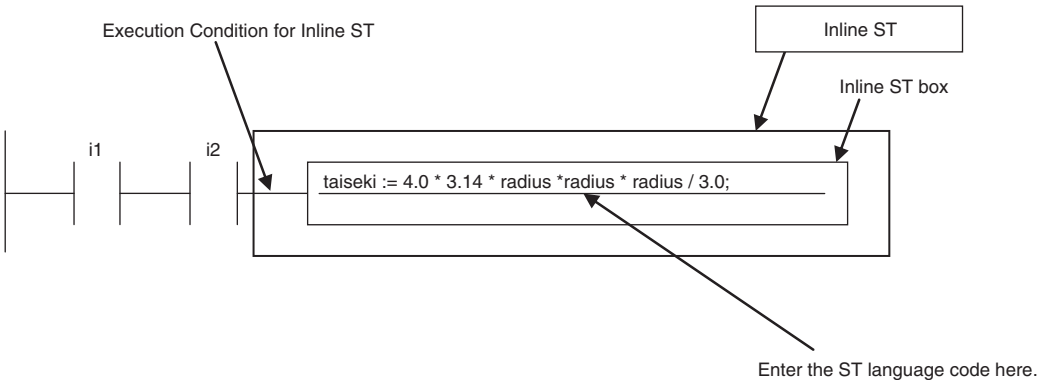
TRUE→FALSE

FALSE→TRUE

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.

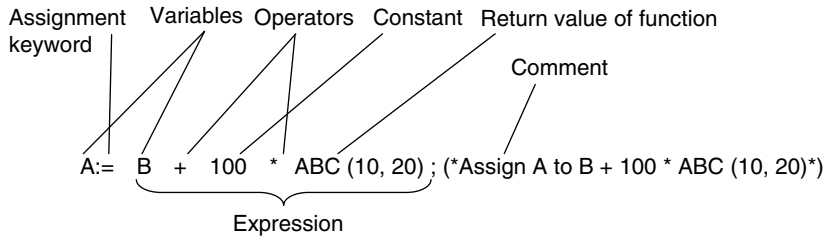
- Loop constructs and control constructs such as IF THEN ELSE are provided.
- You can write programs like high-level languages such as C, and you can include comments to make the program easy to read.

```
1
2       // Determine TableNo
3       FOR i:=0 TO ItemNum DO
4
5           IF (MinNo[i] <= ItemBox[i]) AND (ItemBox[i] <= MaxNo[i]) THEN   // Normal
6               TableNo[i] := ItemBox[i];
7               RangeOK[i] := TRUE;
8
9           ELSIF (ItemBox[i] > MaxNo[i]) THEN                              // Upper
10              TableNo[i] := MaxNo[i];
11              RangeOK[i] := FALSE;
12
13          ELSE                                                            // Lower
14              TableNo[i] := MinNo[i];
15              RangeOK[i] := FALSE;
16
17          END_IF;
18
19      END_FOR;
```

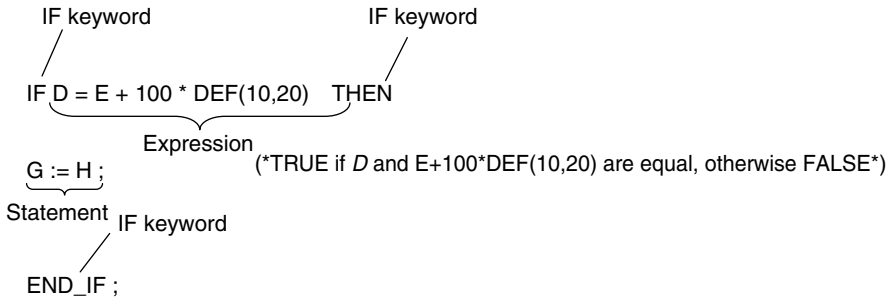
Structure of ST

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 (;).

```
IF A=B THEN
   C := D;   } Assignment statement
END_IF;      } IF construct
```

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 6-117.
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;

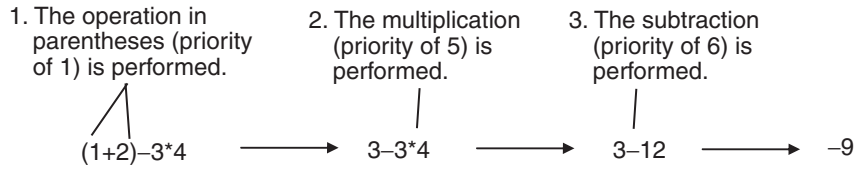
EXIT	Exit the loop.	FOR i := 1 TO 100 DO
		  FOR j := 1 TO 10 DO
		  IF Val[i, j]>100 THEN EXIT;
		  END_IF;
		  END_FOR;
		END_FOR;

;	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.



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 6-123 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*1	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	---	---	---	---*2	---*2	---	---
Enumeration	OK	OK	---	---	---	---	OK	---	---
Structure parent	OK	OK	---	---	---	---	---	---	---
Array parent	OK	OK	---	---	---	---	---	---	---

OK: Possible

---: A building error will occur.

*1 Integer variables are calculated as real number variables even if they set as operands. If a rounding error is included in the result of calculations, the result may not be an intended value because all values after the decimal point are truncated. Use the EXPT and TO_** (Integer Conversion Group) instructions together to round values after the decimal point.
Example: TO_INT(EXPT(X,Y))

*2 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:

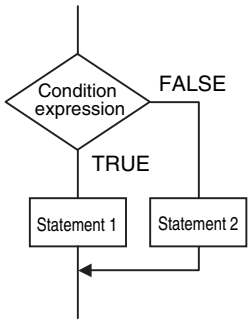
IF, THEN, (ELSE), END_IF

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.

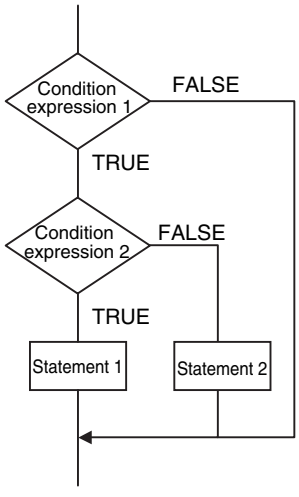
Precautions:

- 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:

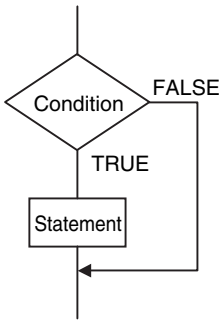


ELSE corresponds to the previous THEN statement, as shown above.

- 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:

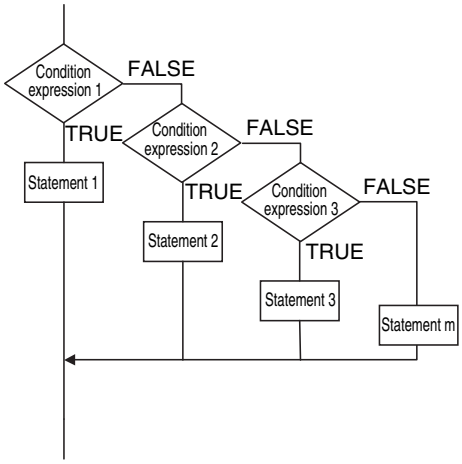
IF, THEN, ELSIF, (ELSE), END_IF

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.

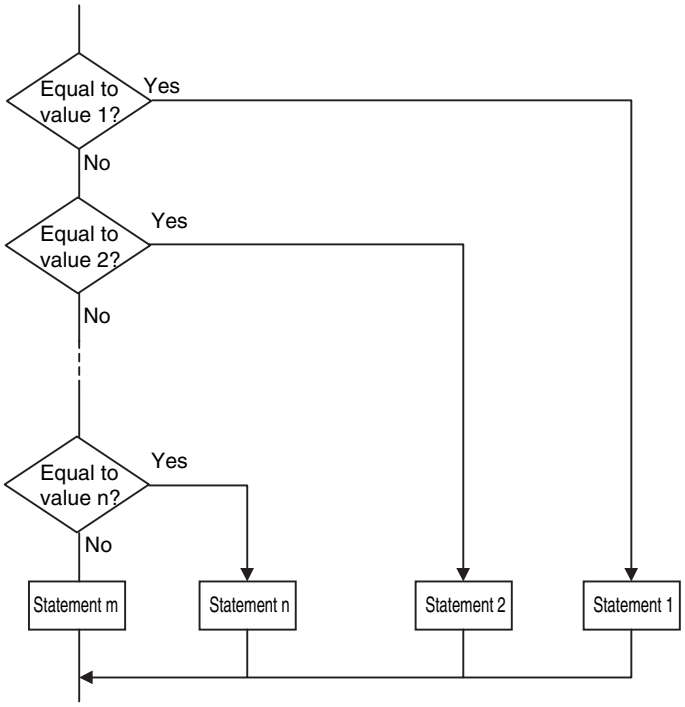
Reserved Words:

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;
```

A value of 1 is assigned to variable *X* when variable *A* is 1.
A value of 2 is assigned to variable *X* when variable *A* is 2.
A value of 3 is assigned to variable *X* when variable *A* is 3.
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;
```

A value of 1 is assigned to variable *X* when variable *A* is 1.
A value of 2 is assigned to variable *X* when variable *A* is 2 or 5.
A value of 3 is assigned to variable *X* when variable *A* is between 6 and 10.
A value of 4 is assigned to variable *X* when variable *A* is 11, 12, or between 15 and 20.
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:

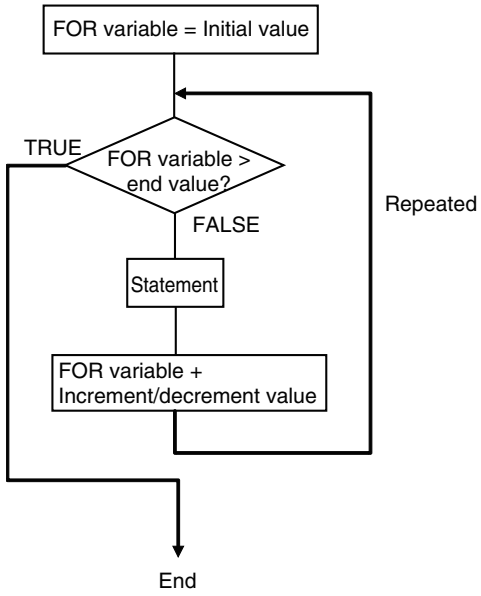
FOR, TO, (BY), DO, END_FOR

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;
```

Additional Information

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;
```

Data Types That You Can Use in FOR Constructs

Classification	Data type		<*FOR_variable*>, <*initial_value*>, <*end_value*>, and <*increment /decrement*>*1
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.*2
Derivative data types	**Structures**	**Structures**	Not supported.
		Members	Supported for integers and enumerations only.*2
	Unions	**Unions**	Not supported.
		Members	Not supported.
	Enumerations		Supported.*2

*1 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.

*2 You cannot use enumerations for <*FOR_variable*>, <*end_value*> and <*increment/decrement*>.

● WHILE

Overview:

This construct repeatedly executes the specified statements as long as a condition expression is TRUE.

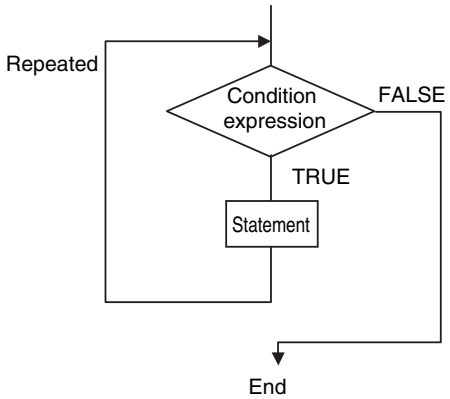
Reserved Words:

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.

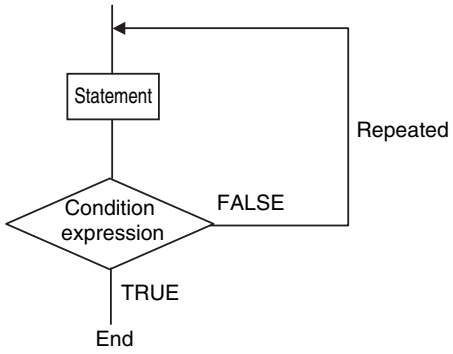
Reserved Words:

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:

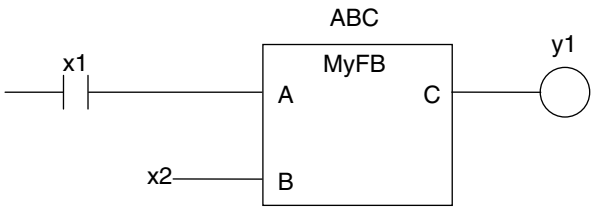
Give the argument specifications (to pass the values of the specified variables to the input variables of the called function block) and the return value specification (to specify the variable that will receive the value of the output variable of the called function block) in parenthesis after the instance 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



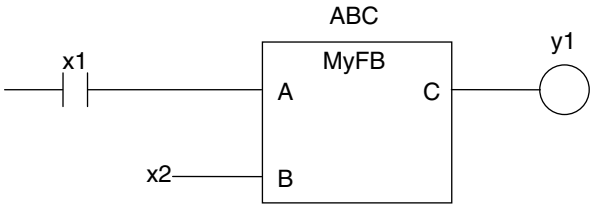
- You can give the arguments and return values in any order.
- You can omit the input variable names and input parameter names. If you omit these names, the values assigned to the input variables for the previous call are assigned to the input variables again. If this is the first time that the function 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.

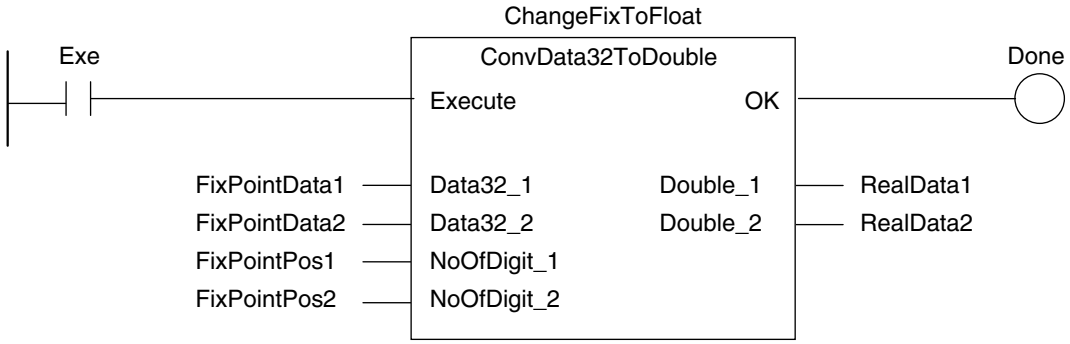
Example

- 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

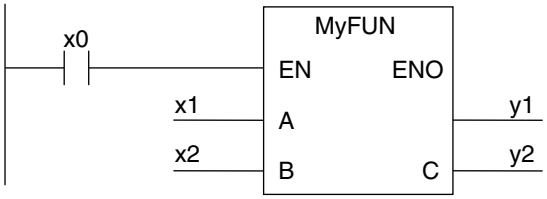
Statement Structure:

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

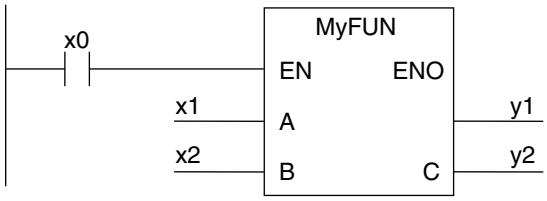
- You can give the arguments in any order.
- You can omit the input variable names and input parameter names. If they are omitted, the input variables are assigned their initial values.
- You can omit *EN* as well. If it is omitted, *EN* is assigned a value of TRUE.

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:

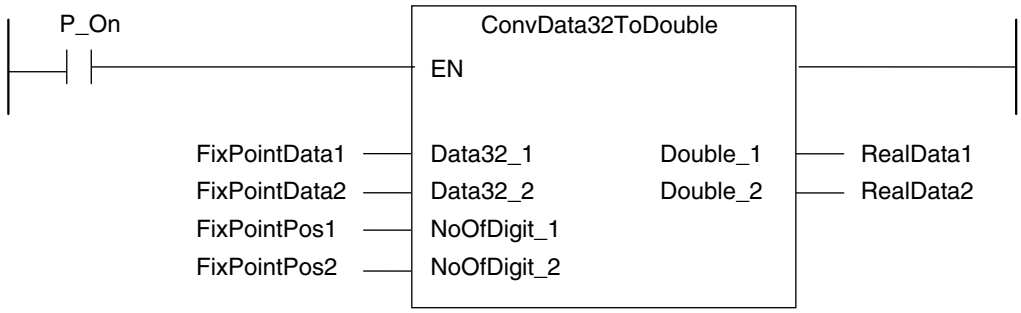
- Programming
- Notation 1

  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:

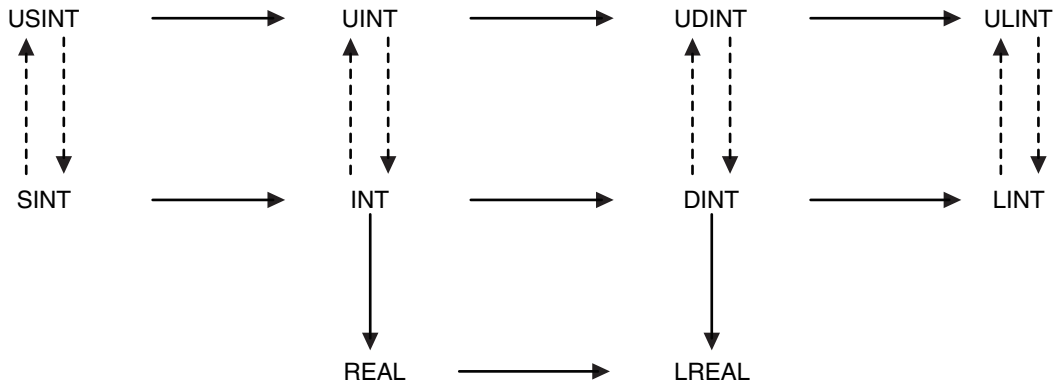
The higher the rank, the larger the range of numerical values that the data type can express.

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

Rank	Data type
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 NY-series Controller, 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

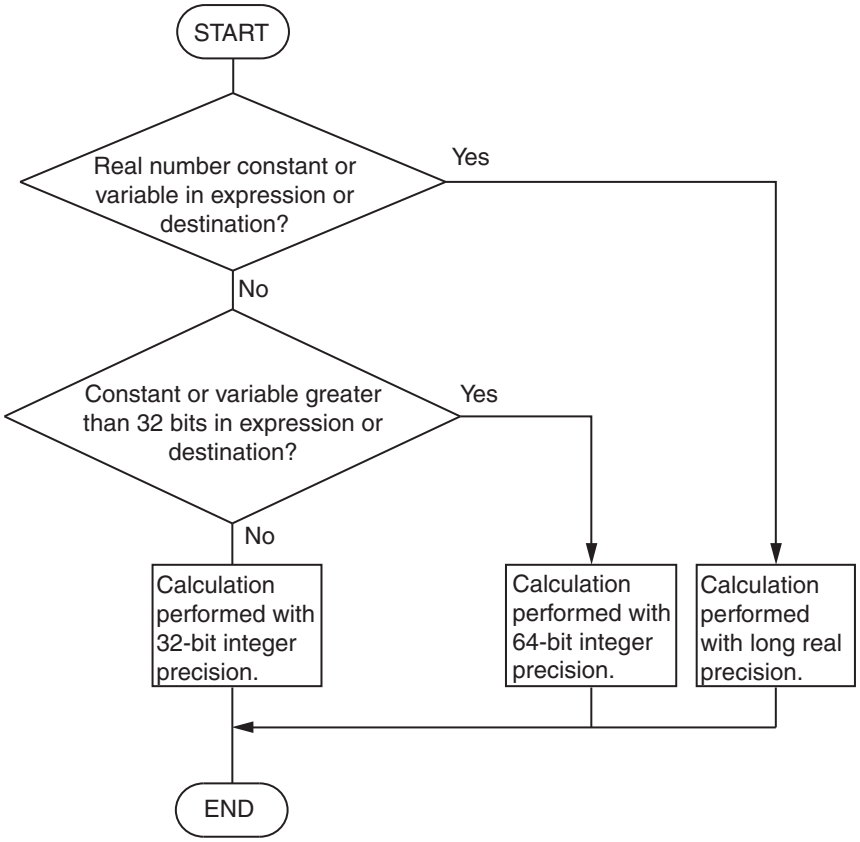
The calculation precision of an expression that contains a constant without a data type specification is automatically determined by the data types of the variables and constants that are given in the expression and destination.

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.

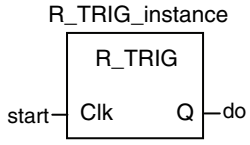
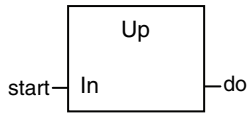
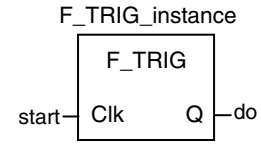
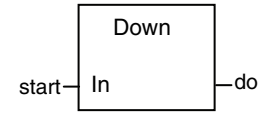
Example: realv := 2 / 3 * 4; // The data type of the *realv* variable is REAL.
Dividing 2 by 3 does not include an integer that exceeds 32 bits for the divisor or dividend, so the calculation is performed with 32-bit integer precision. The calculation result is 0.
In the next step, the *realv* variable is a real number, so the calculation of 0 * 4 is performed with long real precision. The calculation result is 0.

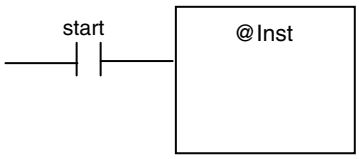
Precautions for Correct Use

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.

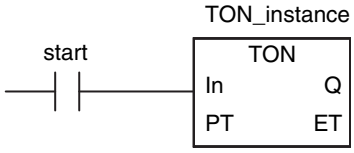
Differences between ST and Ladder Diagrams

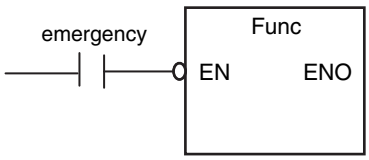
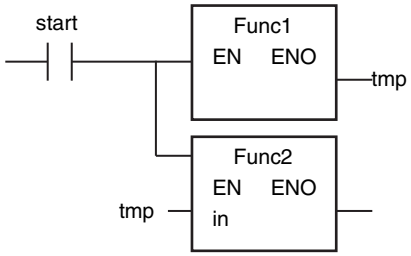
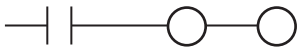
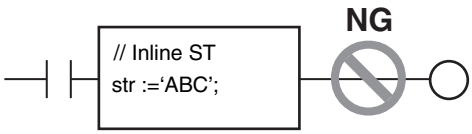
The differences between ST and ladder diagrams are described below.

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.

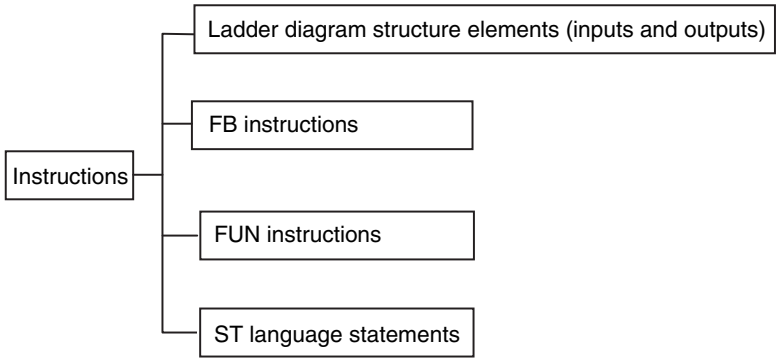
6-6 Instructions

This section describes the instructions that are pre-defined by the NY-series Controller.

For details on these instructions, refer to the *NY-series Instructions Reference Manual* (Cat. No. W560) and *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561).

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

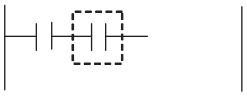
The fundamental specifications of the instructions follow the specifications of functions and function blocks.

This section describes specifications that are unique to instructions.

Ladder Diagram Structure Elements (Inputs and Outputs)

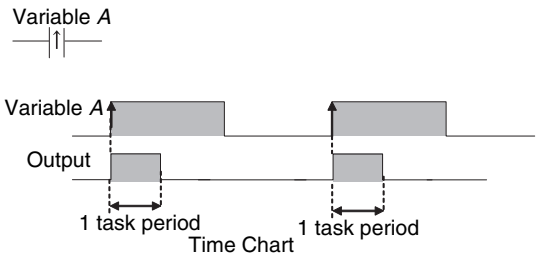
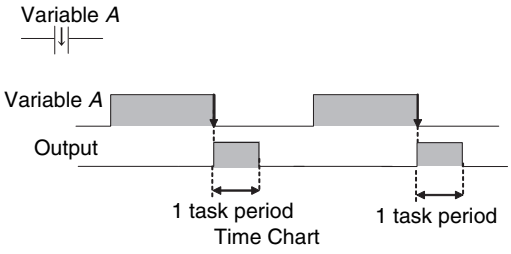
● Locations

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.

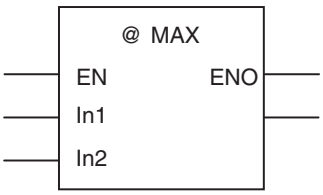
Instruction Options

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 NY-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 *NY-series Instructions Reference Manual* (Cat. No. W560).

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 *NY-series Instructions Reference Manual* (Cat. No. W560).

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 6-136.

● Events

When an instruction error occurs, an event is created for it. For details on events, refer to *Events for Instruction Errors* on page 6-137.

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 constructs	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 condition expression	No statements between REPEAT and END_REPEAT are executed.

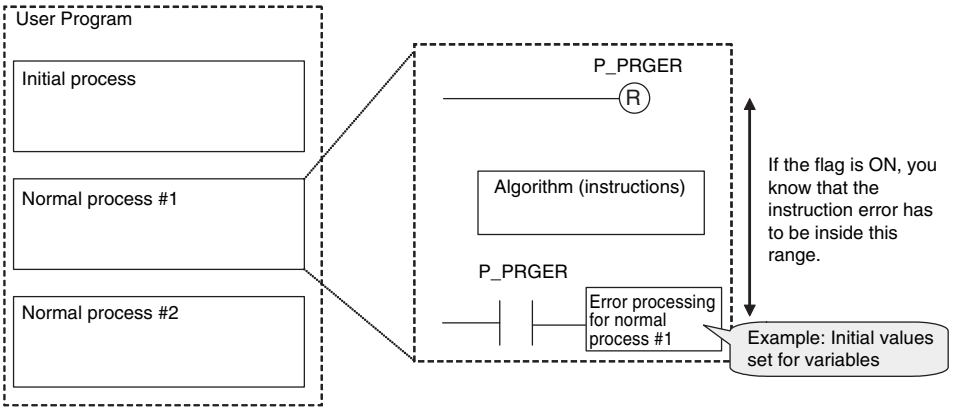
Instruction Error Flag

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

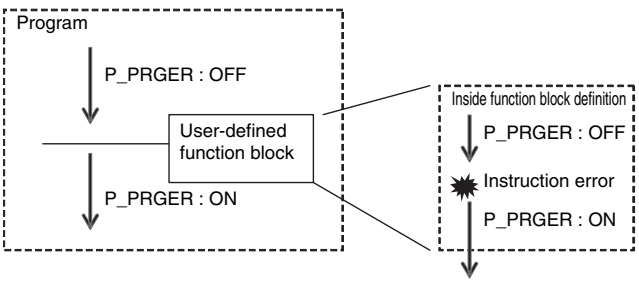
The user can write the P_PRGER Flag. You can temporarily set the value of this flag to FALSE through a user operation to determine if the error occurs within a specific range, for example. After this flag changes to TRUE, it remains TRUE until the operating mode is changed or the flag is overwritten 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 *8-5 Event Logs* for the procedure to check events. For information on the events that are created, refer to the *NY-series Instructions Reference Manual* (Cat. No. W560).

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.

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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561).
- 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-E1-03 or later) for the procedures to manipulate them.

6-7-1 Namespaces

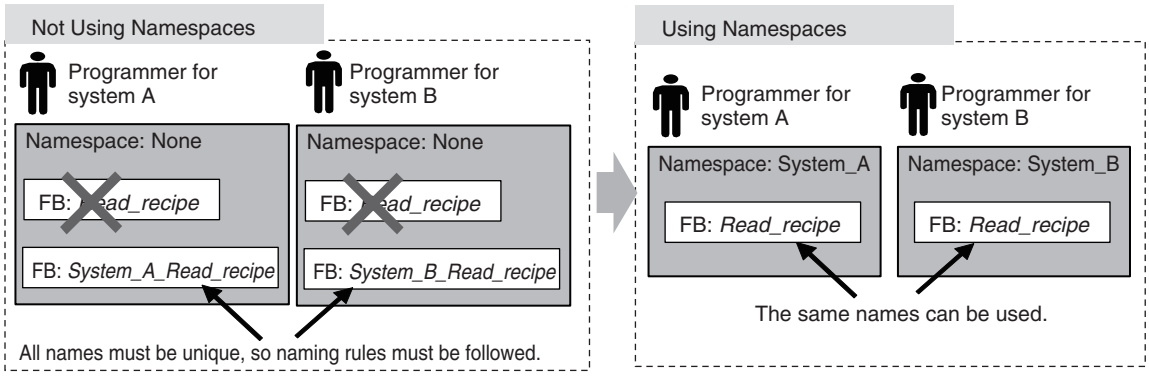
Namespaces are a system for grouping function block definitions and other entities to manage them in nested structures. They are similar to grouping files in folders to manage them in a directory structure. If you do not use namespaces, the name of each function block definition or other entity must 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:

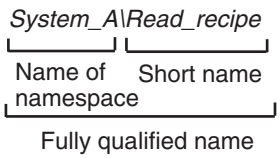
Location of namespace	Expression
Outside of an algorithm	*System_A\Read_recipe*
Inside of an algorithm	*\\System_A\Read_recipe*

● 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:



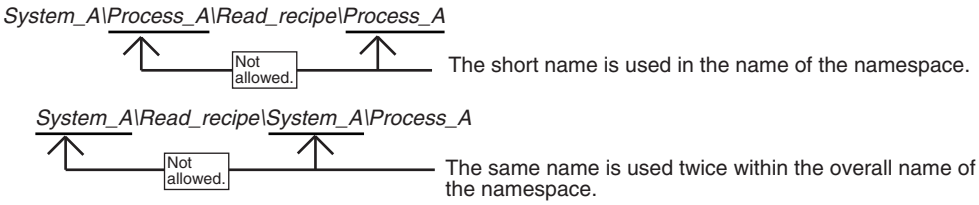
● Restrictions on Namespace Notation

- 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.



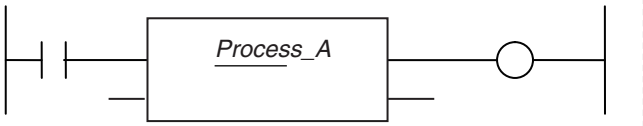
- You can use the short name of a POU definition in the algorithm of a POU definition if it is in the same namespace. However, an error will occur when you build the program if there is a POU definition or data type with the same short name at a higher level in that namespace.
  For example, assume that 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.

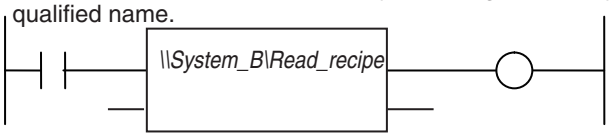
Namespace	Entity	Short name
System_A	Function block definition	*Lifter*
System_B	Function block definition	*Read_recipe*
System_C	Function block definition	*Calculate_upper_limit*

The following notation is used in the namespace declaration for the *Lifter* function block definition.

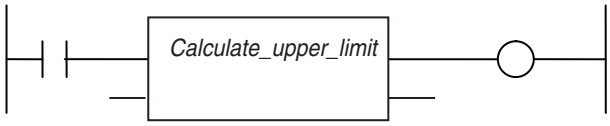
System_C

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

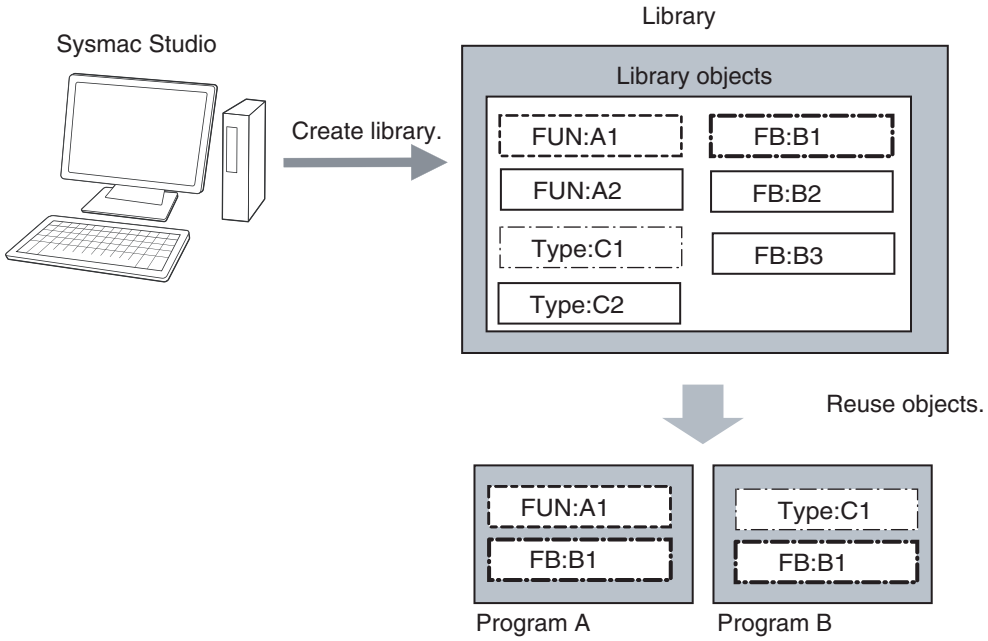
This section describes the specifications of libraries. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504-E1-03 or later) for specific procedures.

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 NY-series Controller 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.

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.

● Selecting Library Objects

You can select the objects to include in a library.

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.

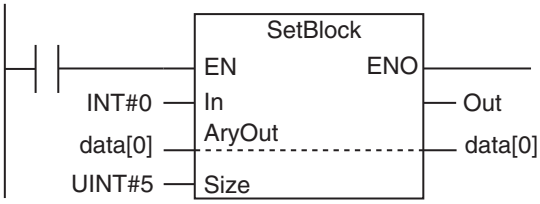
6-9-1 Array Specifications for Input Variables, Output Variables, In-Out Variables

Some instructions handle array variables.

Example:

Data Type Table

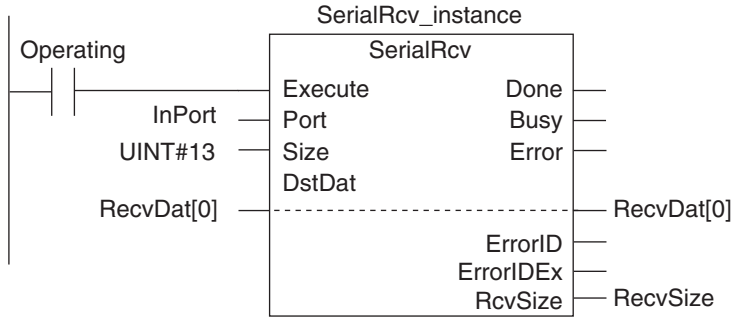
Variable name	Data type
data	ARRAY[0..4] OF INT



6-9-2 Structure Variables for Input Variables, Output Variables, In-Out Variables

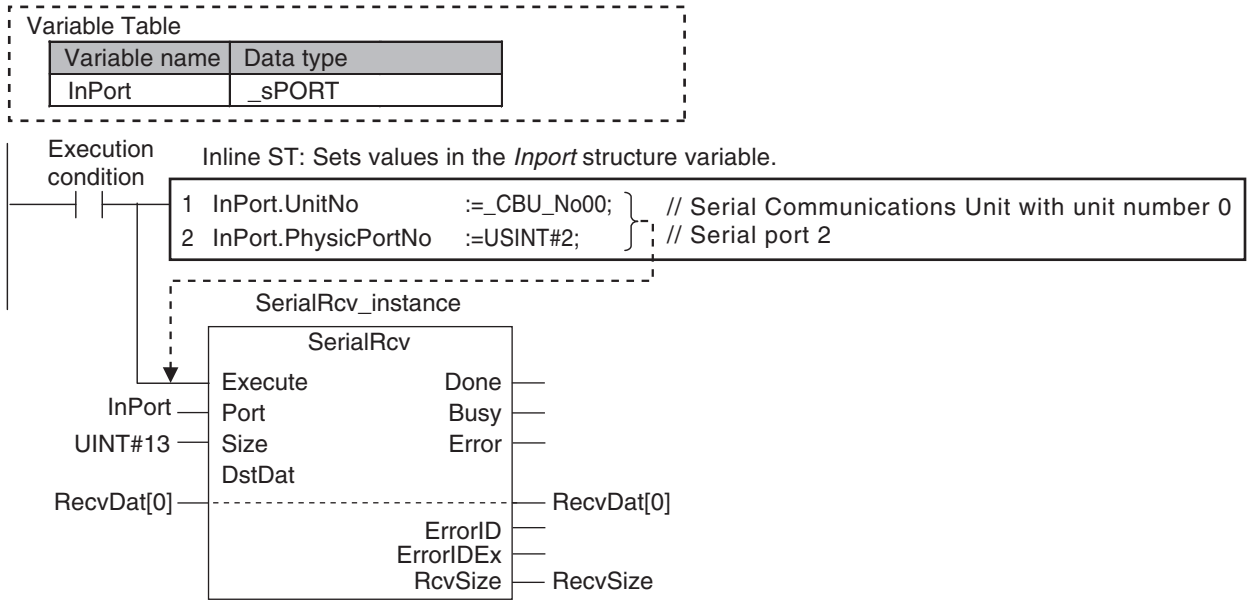
Some instructions have structure variables for input, output, or in-out variables.

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 *NY-series Instructions Reference Manual* (Cat. No. W560) for details.

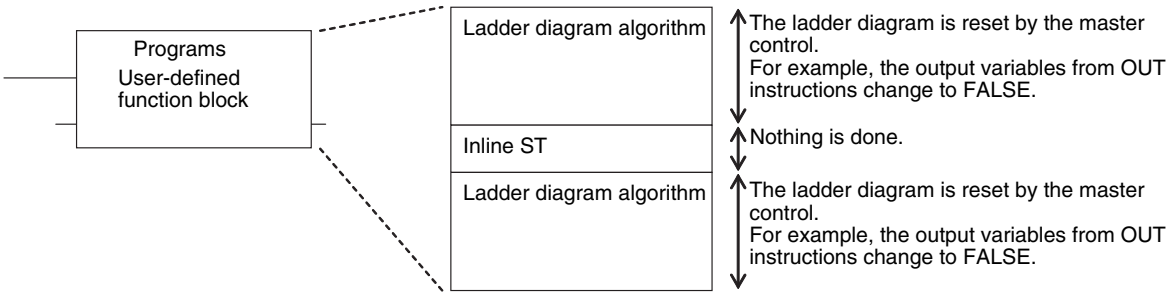
Master Control Programming Languages

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 *NY-series Instructions Reference Manual* (Cat. No. W560) 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 *1-4 Overall Operating Procedure for the NY-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 NY-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 debugging programs to manipulate inputs from outside the Controller.

7-2-2 Simulation Execution

You can do the following for simulations.

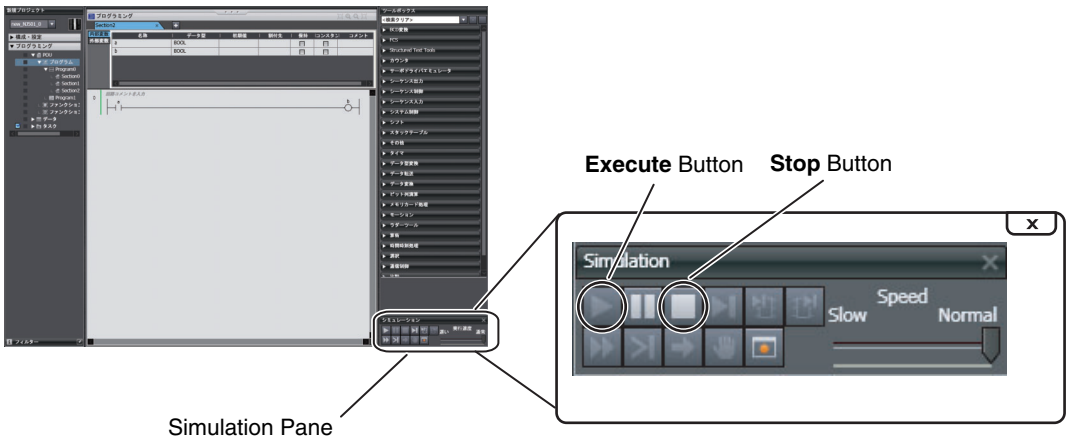
- Start and stop the Simulator.
- Check the logic of programs.
- Use online debugging functions.

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.

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.	Not 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.

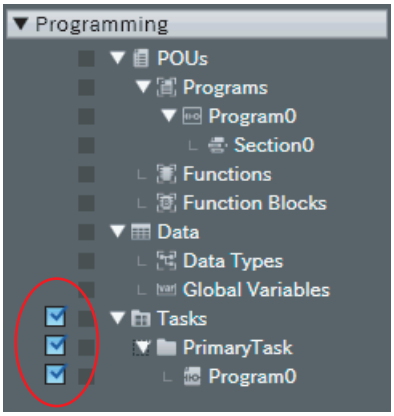
- Setting simulation programs
- Setting debug programs
- Setting the simulation speed

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 **Settings For Debugging** – **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* *NY-series Industrial Panel PC / Industrial Box PC Motion Control User's Manual* (Cat. No. W559) *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 *8-4-1 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 *NY-series Industrial Panel PC / Industrial Box PC Motion Control User's Manual* (Cat. No. W559).

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 Controller 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 *8-4-1 Forced Refreshing* and *8-4-2 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 Controller. Refer to *8-4-3 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 *8-4-4 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

Controller Functions

This section describes the functionality provided by the NY-series Controller.

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, variables, and absolute encoder home offsets in the NY-series Controller 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 NY-series Controller during the Clear All Memory operation.

After you clear the memory, the NY-series Controller operates in the same way as immediately after you create the system configuration with the NY-series Controller in the factory default condition.

● Operations from the Sysmac Studio

Connect the Sysmac Studio to the NY-series Controller online, and select **Clear All Memory** from the Controller Menu.

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for specific procedures.

8-1-2 Clock

Of the built-in clocks in the NY-series Industrial PCs, this section describes a clock in the Controller.

Introduction

In the NY-series Industrial PC, there is a clock and time zone in either Windows or the Controller that is embedded in the Real-Time OS.

The clock data of the clock in Windows is the master clock and the clock data of the clock in the Controller is synchronized with the clock in Windows. Therefore, you can change the clock data of the clock in Windows, but you cannot change the clock data of the clock in the Controller.

If you cannot get the clock data from Windows due to a shutdown on Windows only or other reasons, the clock data of the clock in the Controller is not synchronized with the clock in Windows. The clock in the Controller continues to operate even while it is not synchronized with the clock in Windows.

When you restart Windows to get the clock data from Windows, the clock data of the clock in the Controller will be synchronized with the clock in Windows again.

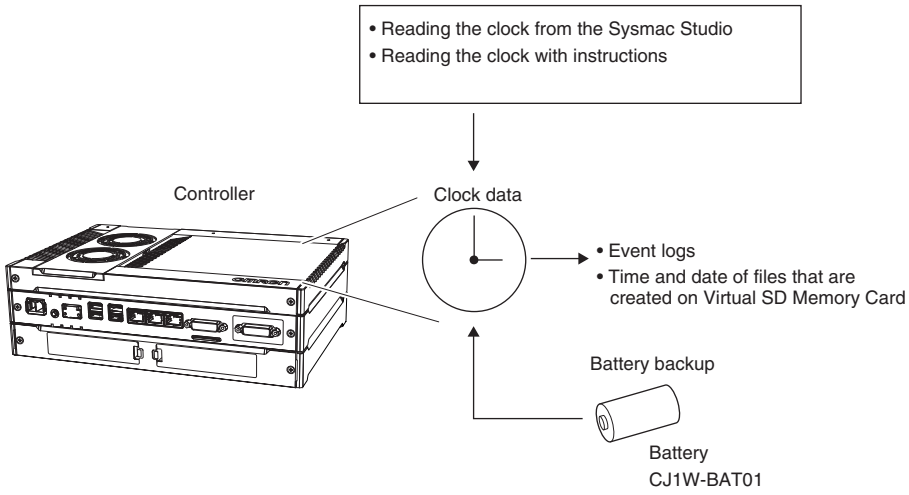
The clock data of the clock in the Controller is used for event logs and for the time and date of files that are created on the Virtual SD Memory Card.

The time zone in the Controller depends on the time zone in Windows.

You cannot separately set the time zone in Windows and the time zone in the Controller.

The following functions are supported for the clock in the Controller.

- Reading the clock from the Sysmac Studio (Writing is not possible.)
- Reading the clock from instructions (Writing is not possible.)
- Reading the clock from system-defined variables (Writing is not possible.)



Clock Data Range for Controller

- 2000-01-01 to 2099-12-31 (January 1, 2000 to December 31, 2099).

Setting the Time Zone and the Local Time

The time zone and local time in the clock data in the Controller depend on the settings in Windows. Therefor, before you use the Controller for the first time, set the time zone and local time in the clock data in Windows.

The clock data that is read by the EtherCAT slaves from the Controller 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.

Reading the Clock Data

If the clock data is incorrect, the incorrect value is read.

● Reading the Clock Data from Instructions (Writing Is Not Possible)

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 (Writing Is Not Possible)

You can select **Controller Clock** from the Controller Menu of the Sysmac Studio to display the clock data.

Related System-defined Variables

Variable name	Meaning	Description	Data type	R/W
_CurrentTime	System Time	This variable contains the Controller's internal clock data.	DATE_AND_TIME	R

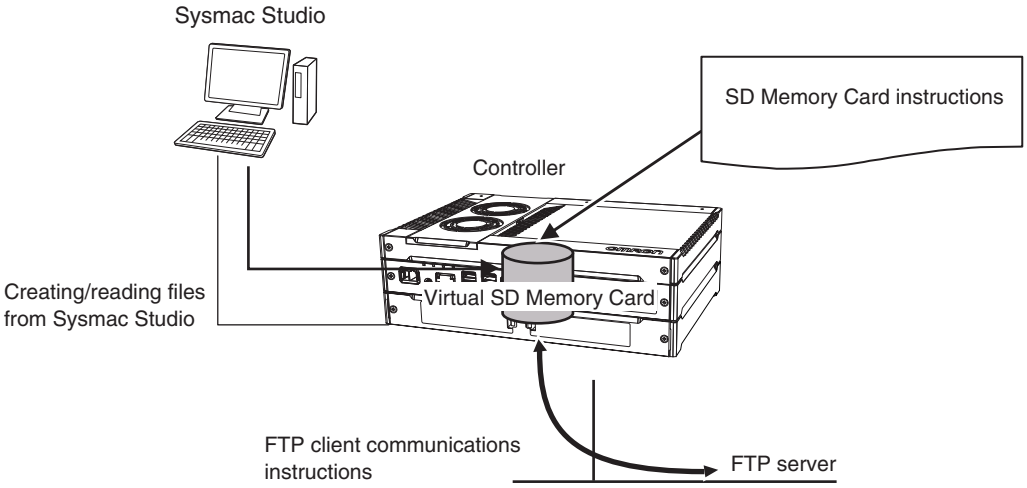
8-2 SD Memory Card Operations

This section describes the functions that you can use for SD Memory Cards.

In the NY-series Controller, a shared folder with Windows is handled as a Virtual SD Memory Card. In this manual, only the SD Memory Card is sometimes given without "virtual".

8-2-1 SD Memory Card Operations

The NY-series Controller supports the following functions for SD Memory Cards.



Function	Introduction
SD Memory Card instructions	You can access Virtual SD Memory Cards from instructions in the user program.
FTP client communications instructions	You can use these instructions to transfer files via FTP from the Controller 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 Virtual SD Memory Card through EtherNet/IP.
File operations from the Sysmac Studio*1	You can perform file operations from the Sysmac Studio for the Virtual SD Memory Card in the Controller. You can perform file operations for Controller files in the Virtual SD Memory Card and save standard document files on the computer.
SD Memory Card backup	You use the Virtual SD Memory Card in the Controller to backup, restore, and verify user programs and data in the Controller. Refer to *9-2 SD Memory Card Backups* for details.

*1 For an NY-series Controller, you cannot initialize the SD Memory Card or change the folder and file attributes on the SD Memory Card.
The volume level of the SD Memory Card is not displayed on the Status Bar in the SD Memory Card Window of the Sysmac Studio because the Virtual SD Memory Card is using a shared folder.

8-2-2 Setting Shared Folders and Virtual SD Memory Cards

This section describes the settings for using the SD Memory Card operations in the NY-series Controller.

Setting Shared Folders

You need to set the shared folder in Windows before you make the settings for the Virtual SD Memory Card.

In the default setting of Windows in an NY-series Industrial PC, D:\OMRON-NY\VirtualSDCard is set as the shared folder for a Virtual SD Memory Card.

Refer to the *NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat. No. W568) for details on setting the shared folder.

Setting Virtual SD Memory Cards

Make the settings for the Virtual SD Memory Card in the Controller Setup of the Sysmac Studio.

Refer to *Virtual SD Memory Card Settings* on page 4-5 for details on the Virtual SD Memory Card Setting.

Additional Information

You can also make the settings for the Virtual SD Memory Card with the Industrial PC Support Utility instead of the Sysmac Studio.

Refer to the *NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat. No. W568) for the procedure to set the Virtual SD Memory Card with the Industrial PC Support Utility.

8-2-3 Recognizing and Canceling Recognitions of Virtual SD Memory Cards

This section describes how to recognize and cancel the recognition of the Virtual SD Memory Card.

Recognizing Virtual SD Memory Cards

The Virtual SD Memory Card is recognized if the following operations are performed.

When the Virtual SD Memory Card is recognized, the *_Card1Ready* (SD Memory Card Ready Flag) system-defined variable changes to TRUE.

- Starting the NY-series Controller
- Downloading
- Restoring data
- Changing the settings of the Virtual SD Memory Card with the Industrial PC Support Utility
- Checking the recognition of the Virtual SD Memory Card with the Industrial PC Support Utility

The Virtual SD Memory Card cannot be recognized in the following cases.

- Windows is stopped.
- The file sharing service of Windows is stopped.
- The settings of the Virtual SD Memory Card are invalid.

Canceling Recognition of Virtual SD Memory Cards

The recognition of the Virtual SD Memory Card is canceled due to the reasons given in the following table.

Cause
Windows was shut down or restarted.
The file sharing service of Windows was stopped or disabled.
The computer name on Windows was changed.
The IP address of the internal port for Windows was changed.
Sharing the shared folder was canceled.
The shared folder was deleted.
The access right of the shared folder was changed.
The IP address of the internal communications port for Controller was changed to a different IP address of the internal communications port for Windows.

When the recognition of the Virtual SD Memory Card is canceled, the *_Card1Ready* (SD Memory Card Ready Flag) system-defined variable changes to FALSE and a Shared Folder Recognition Cancel Completed event will occur.

The file access that is in progress is failed if the recognition is canceled while accessing the Virtual SD Memory Card.

The operations after canceling are different depend on the causes that are canceled.

● If the Cause Is *Sharing the Shared Folder Was Canceled* or *the Shared Folder Was Deleted*

If the cause that the recognition of the Virtual SD Memory Card is canceled is *sharing the shared folder was canceled* or *the shared folder was deleted*, the Virtual SD Memory Card is not recognized again only by re-sharing the shared folder in Windows.

The Virtual SD Memory Card is recognized again by either of the following methods.

- Re-share the shared folder and execute **Check recognition** from the Industrial PC Support Utility. However, if the file is open with an instruction in the SD Memory Card instructions before the sharing is canceled, you need to close the opened file.
- Re-share the shared folder and the idle session time (15 minutes for the default) for Windows is elapsed from the sharing was canceled.

● If the Cause Is One Other Than *Sharing the Shared Folder Was Canceled* or *the Shared Folder Was Deleted*

The Virtual SD Memory Card is automatically recognized when the cause is removed and the normal status is restored.

If the Virtual SD Memory Card is recognized, when downloading, performing restore operation, clearing all memory, or transferring the settings from the Industrial Support Utility, the followings will happen.

- The recognition of the Virtual SD Memory Card is canceled.
- The *_Card1Ready* (SD Memory Card Ready Flag) system-defined variable changes to FALSE.
- A Shared Folder Recognition Cancel Completed event occurs.

However, while accessing the Virtual SD Memory Card, the recognition of the Virtual SD Memory Card is not canceled and a Shared Folder Recognition Cancel Failed event occurs.

Precautions for Correct Use

Do not turn OFF the power supply to the Industrial PC while accessing the Virtual SD Memory Card.

The files may be corrupted if the power supply is turned OFF.

The *_Card1PowerFail* (SD Memory Card Power Interruption Flag) system-defined variable changes to TRUE at next time you start operation.

8-2-4 Specifications of Shared Folders and Files for Windows

This section describes the specifications of the shared folder that is used as a Virtual SD Memory Card and the file specifications.

Specifications of Shared Folders

The following describes the specifications of the shared folder that is set on Windows.

● Settable Storage

You can use the shared folder that is set in the HDD or SSD built in the industrial PC as a Virtual SD Memory Card.

The shared folder that is not set in the HDD or SSD built in the industrial PC cannot be guaranteed as a Virtual SD Memory Card.

● Restrictions on Capacity of Shared Folder

Use the shared folder as a Virtual SD Memory Card within the capacity of 4 GB.

If the file capacity in the shared folder exceeds 4 GB, correct operation may not be possible.

● File System

You can only use the NTFS file system as a Virtual SD Memory Card.

Folder and File Specifications

● Character Restrictions

Object named by user	Usable characters	Reserved words	Multibyte character compatibility	Case sensitivity	Maximum size (without NULL)
Directory name	0 to 9, A to Z, and a to z, as well as $ % ’ - _ @ ! ‘ ( ) ~= # & + ^ [ ] { } , . ;	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.*1	Case insensitive	65 bytes
File name					65 bytes

*1 Even if the computer supports multibyte characters (e.g., for Japanese), you cannot use them in the Controller.

Precautions for Correct Use

If unusable characters are included in the directory name or file name, the Controller cannot be correctly recognized. Files may not be identified in instructions, in the SD Memory Card Window of the Sysmac Studio or in the FTP server.

Subdirectory Levels

You can create up to 5 levels (example: f1/f2/f3/f4/f5/abc.txt)

You can create up to 5 levels.

Example) If VirtualSDCard is used as a name of the shared folder

VirtualSDCard\f1\f2\f3\f4\f5\abc.txt

Restrictions on Number of Stored Files

Use the number of stored files on a Virtual SD Memory Card which is less than the value given in the following table. If the number of stored files exceeds the value given in the following table, correct operation may not be possible.

Directory level	Number of stored files
Root directory	65,535
Subdirectory	65,534

Maximum Size of One File

The maximum size of any one file is 2,147,483,647 bytes (2 GB −1 byte).

8-2-5 SD Memory Card Instructions

You can perform various operations on the Virtual 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 Virtual 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 Virtual 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 Virtual 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 Virtual SD Memory Card.
Delete Directory	**DirRemove**	The DirRemove instruction deletes a directory from the Virtual SD Memory Card.

8-2-6 FTP Client Communications Instructions

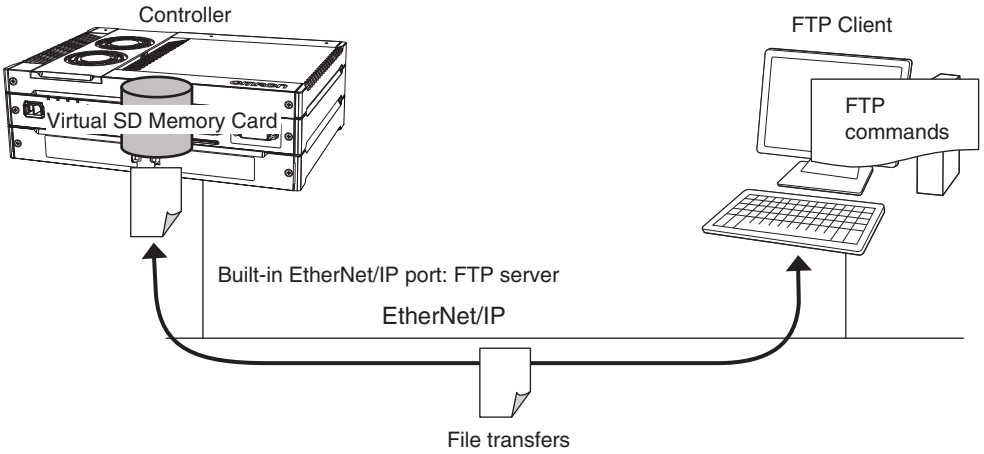
FTP client communications instructions are used to transfer files via FTP from an NY-series Controller to computers or Controllers at Ethernet nodes.

The files on the Virtual 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 Virtual 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 Virtual SD Memory Card.

8-2-7 FTP Server

You can read and write files on the Virtual SD Memory Card via EtherNet/IP by sending FTP commands to the built-in EtherNet/IP port from an FTP client.
Refer to the *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherNet/IP Port User's Manual* (Cat. No. W563) for details.



8-2-8 File Operations from the Sysmac Studio

You can perform file operations from the Sysmac Studio for the Virtual SD Memory Card in the Controller. In addition to Controller files, you can also store document files or other files on the Virtual SD Memory Card.

8-2-9 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 Virtual SD Memory Card is recognized in an NY-series Controller. It is FALSE when a Virtual 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	The NY-series Controller does not use this variable. The value is always FALSE.	BOOL
_Card1Err	SD Memory Card Error Flag	The NY-series Controller does not use this variable. The value is always FALSE.	BOOL
_Card1Access	SD Memory Card Access Flag	The NY-series Controller does not use this variable. The value is always FALSE.	BOOL
_Card1Deterio-rated	SD Memory Card Life Warning Flag	The NY-series Controller does not use this variable. The value is always FALSE.	BOOL
_Card1PowerFail	SD Memory Card Power Interruption Flag*1	TRUE when the power supply to the Controller was interrupted during access to the Virtual SD Memory Card. TRUE: Power was interrupted during Virtual SD Memory Card access. FALSE: Operation is normal.	BOOL

*1 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 Virtual SD Memory Card and to see if the Virtual SD Memory Card operates properly. If the correct file is missing or the Virtual SD Memory Card does not operate properly, download the correct file to the Virtual SD Memory Card again. Cycle the power supply to the Controller or reset the Controller, and then see if the Virtual SD Memory Card operates properly. When you are finished, change the 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.

8-2-10 Exclusive Control of File Access in Virtual SD Memory Cards

Access to files on the Virtual SD Memory Card is possible with the following methods.

(1) FTP server

(2) SD Memory Card instructions

(3) FTP client communications instructions

(4) File operations from the Sysmac Studio

However, if the same file on the Virtual 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><td rowspan="5">Later access</td><td rowspan="2">Instructions</td><td>Reading</td><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><td>Writing</td><td>Perform exclusive control.</td><td>Perform exclusive control.</td></tr>
<tr><td rowspan="2">File operations from the Sysmac Studio</td><td>Reading</td><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><td>Writing</td><td colspan="2">Perform exclusive control.</td><td>---</td><td>---</td><td>Perform exclusive control.</td></tr>
<tr><td rowspan="2">FTP server</td><td>Reading</td><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><td>Writing</td><td colspan="4">Perform exclusive control.</td><td>---</td><td>---</td></tr>
</table>

*1 The instructions include the SD Memory Card instructions and the FTP client communications instructions.

For an NY-series Controller, you can access files in the shared folder from Windows.

In such case, perform exclusive controls according to the specifications for sharing files of Windows.

8-3 Security

This section describes the security functions that are supported by the NY-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.

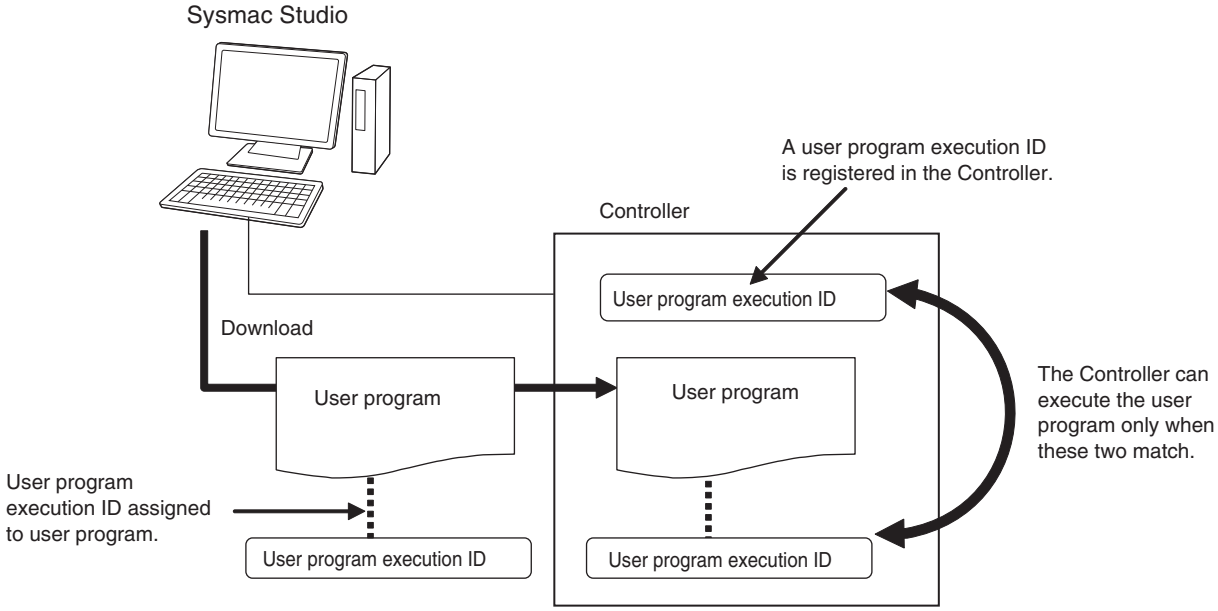
The NY-series Controller supports the following security functions.

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 Controller even if the user program is copied.	8-17
	User program transfer with no restoration information	You can transfer the user program to the Controller without the source code. This prevents anyone from displaying the user program on another computer even if they upload it.	8-20
	Overall project file protection	You can place a password on a project file to protect your assets.	8-21
	Data protection	You can place protection on part of the data in a project file to protect your assets.	8-22
Prevention of incorrect operation	Operation authority verification	You can set operation authorities to restrict the operations that can be performed on the Controller from the Sysmac Studio.	8-24
	CPU Unit write protection	You can prevent rewriting data in the Controller from the Sysmac Studio.	8-26
Prevention of incorrect connections	CPU Unit names	You can check to see if the Controller name and serial ID on the computer and in the Controller are the same to prevent going online with the wrong Controller.	8-27

8-3-1 Authentication of User Program Execution IDs

Introduction

You can set a specific ID (called a user program execution ID) in the Controller in advance. If you do, you can execute only a user program with the same ID.

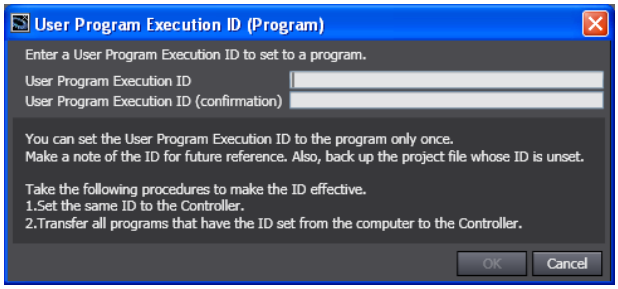


You can prevent a user program from use on a different Controller (hardware).

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

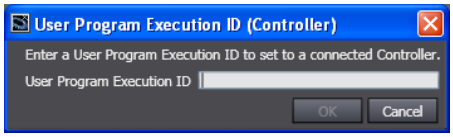
1. Always backup the project files before you assign a user program execution ID.
2. Assign the user program execution ID to the user program offline from the Sysmac Studio.



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 Controller.



The registration of the user program execution ID in the Controller is recorded in the event log. At this time, the user program execution ID in the Controller is overwritten even if it is already registered.

4 Transfer the user program with the same user program execution ID to the Controller.

If the user program execution ID in the user program does not match the user program execution ID in the Controller 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 Controller will not operate.

Precautions for Correct Use

After you assign a user program execution ID to the Controller, you cannot read or delete the ID. To delete the ID from the Controller, perform the Clear All Memory operation on the Controller.

● Operation When an ID Verification Error Occurs

When the User Program Execution ID in the Controller Is Incorrect or Not Registered:

Connect online to the Controller from the Sysmac Studio and perform the following steps.

1 Overwrite or register the correct user program execution ID in the Controller.

2 Cycle the power supply to the Controller, or reset the Controller 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 Controller 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 Controller:

Connect the Sysmac Studio to the Controller 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 Controller:

For security, the user program execution ID that is set in the Controller 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 Controller compares the user program execution ID that is registered in the Controller 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 Controller	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 Controller 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

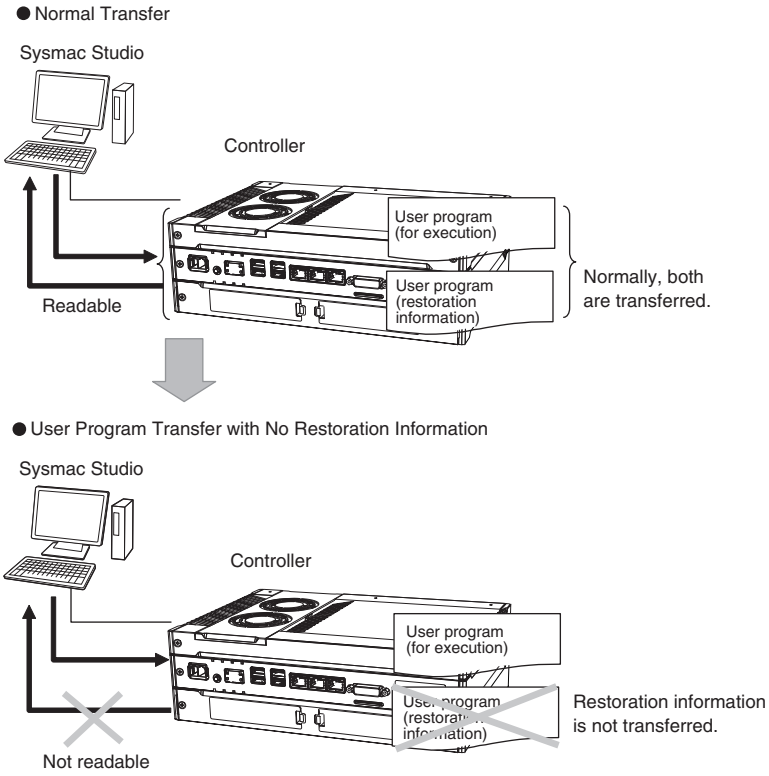
Usable characters	Case sensitivity	Maximum size (without NULL)
0 to 9, A to Z, and a to z	Case sensitive	8 to 32 characters

8-3-2 User Program Transfer with No Restoration Information

You can transfer the user program to the Controller without the source code. This prevents anyone from displaying the user program on another computer even if they upload it.

Introduction

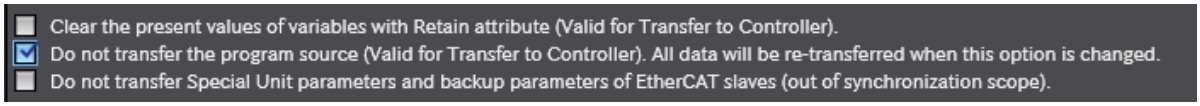
Normally, when you transfer the user program from the Sysmac Studio to the Controller, information is transferred to restore it.
This function does not transfer information for restoration. That makes it impossible to read the user program.



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 Controller, 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.

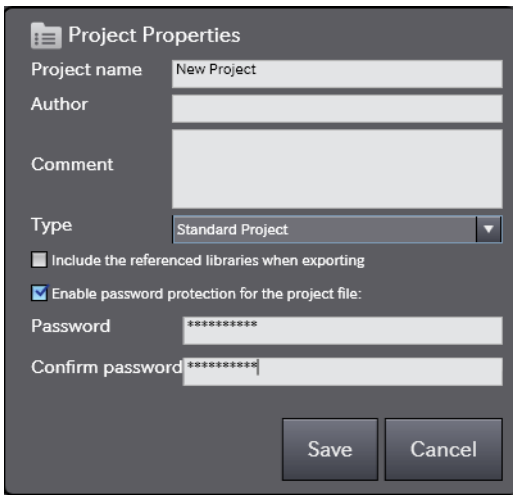


8-3-3 Overall Project File Protection

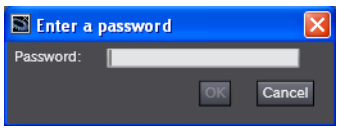
You can place a password on a project file to protect your assets.

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.

8-3-4 Data Protection

You can place protection on part of the data in a project file to protect your assets.

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

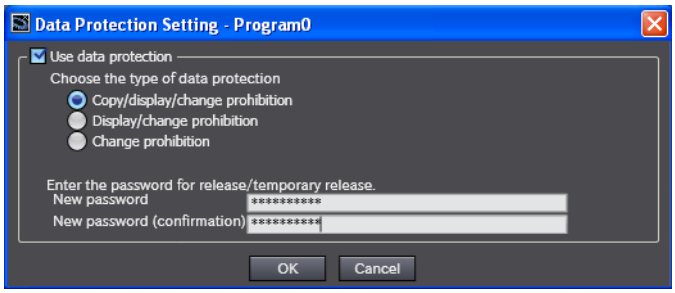
The following table shows the data access methods, restrictions for each restriction level, and the length of time that protection is cleared.

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.

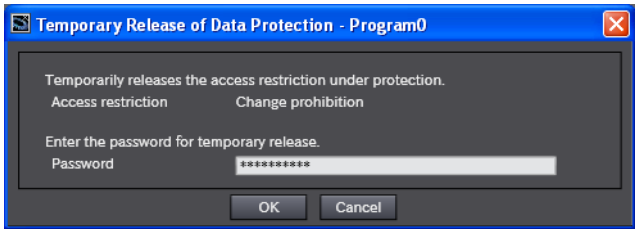
Operating Procedure

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.

8-3-5 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 Controller 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	Optional*1
Maintainer	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.
Operator	
Observer	Not required.

*1 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)	Administrator	Designer	Maintainer	Operator	Observer
Monitoring errors for troubleshooting	OK	OK	OK	OK	OK

I/O monitor operations (examples)	Administrator	Designer	Maintainer	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)	Administrator	Designer	Maintainer	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

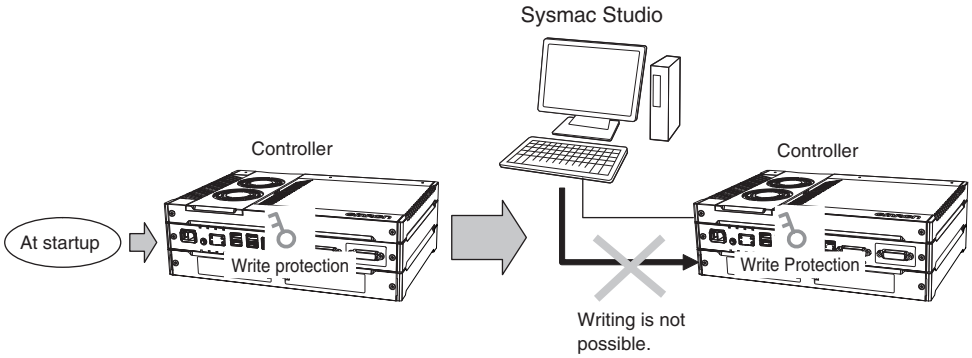
Item	Description
Valid number of characters	8 to 32
Applicable characters	Single-byte alphanumeric characters (case sensitive)

8-3-6 CPU Unit Write Protection

This function disables the ability to write data to Controller 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.

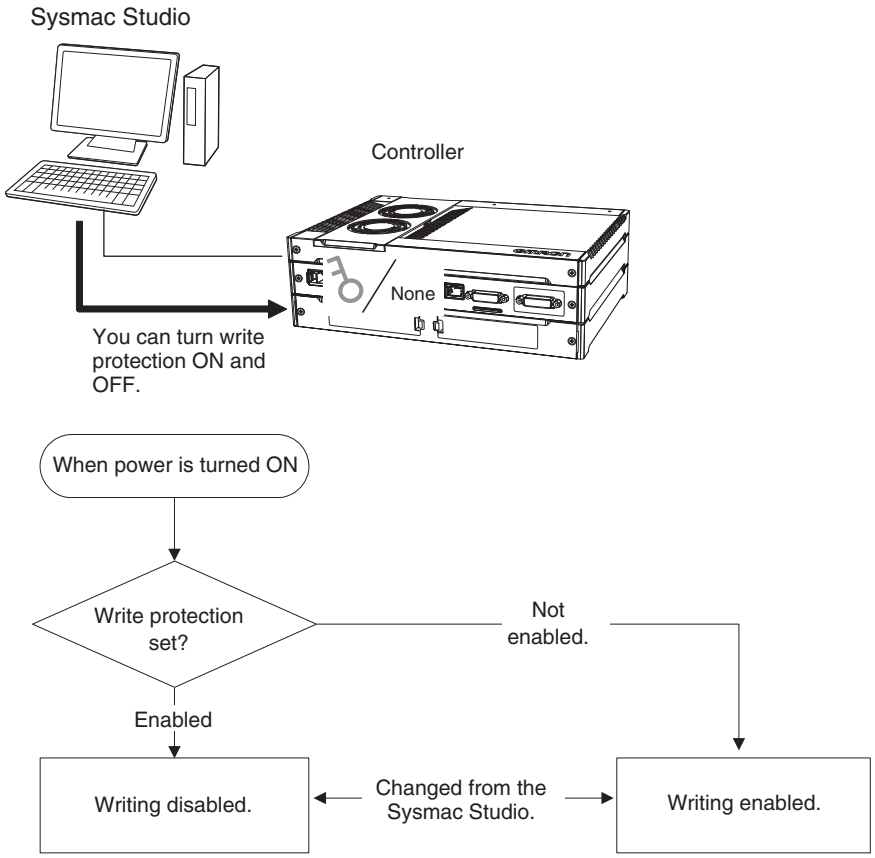


Set whether to automatically enable write protection when the power supply is turned ON in the **Operation Settings** under **Configurations and Setup** – **Controller Setup** of the Sysmac Studio.

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

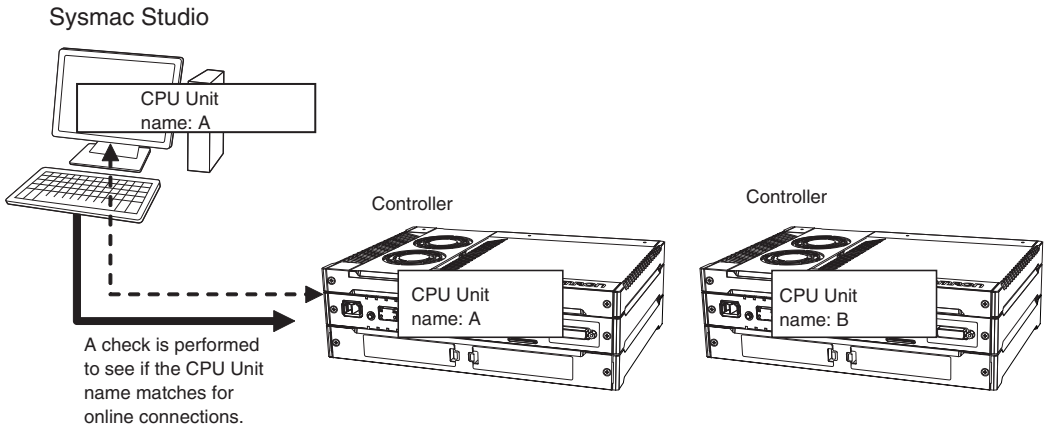
In the Sysmac Studio, go online and select **Security** – **CPU Unit Write Protection** from the Controller Menu to toggle write protection.



8-3-7 CPU Unit Names and Serial IDs

Introduction

Register a CPU Unit name in the Controller. When going online to a Controller from the Sysmac Studio, the CPU Unit name in the project is compared to the name of the Controller being connected to. This helps prevent incorrect connections to the Controller from the Sysmac Studio. It is particularly effective for operations performed over an EtherNet/IP network.

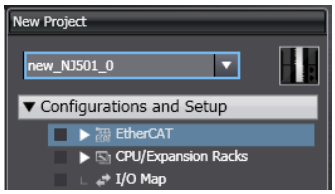


In addition to the CPU Unit name, it is also possible to use serial ID identification based on the Controller production information (optional).

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 Controller online, the Sysmac Studio prompts you to store the CPU Unit name in the Controller.

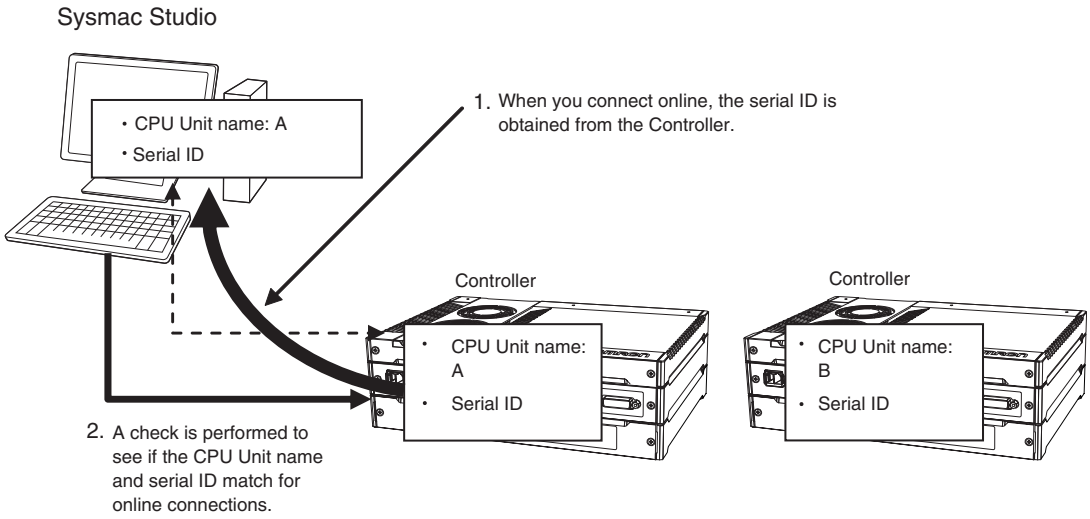
3 After that, when you connect to the Controller online, the Sysmac Studio refers to the CPU Unit name in the project and the CPU Unit name of the Controller you connect to. A warning dialog box is shown if they do not match, and you are asked whether to continue to connect.

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 Controller'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 Controller.



8-4 Debugging

This section describes debugging.

The NY-series Controller provides the following debugging operations.

- Forced refreshing
- Changing present values
- Online editing
- Data tracing
- Differential monitoring

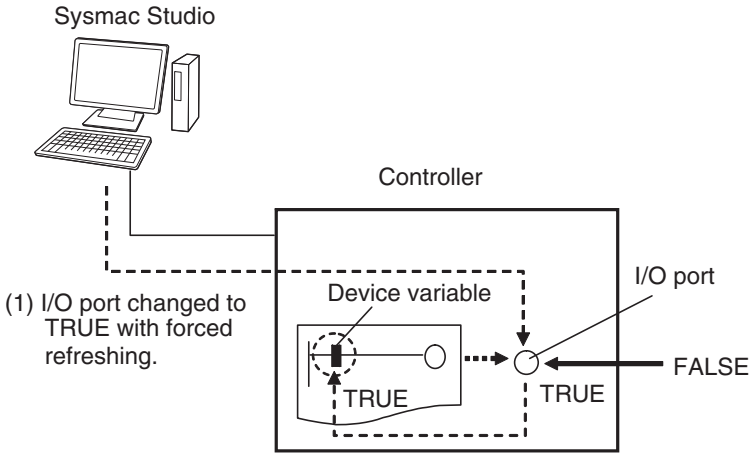
8-4-1 Forced Refreshing

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 8-32 for information how forced refreshing is retained or cleared according to changes in Controller 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

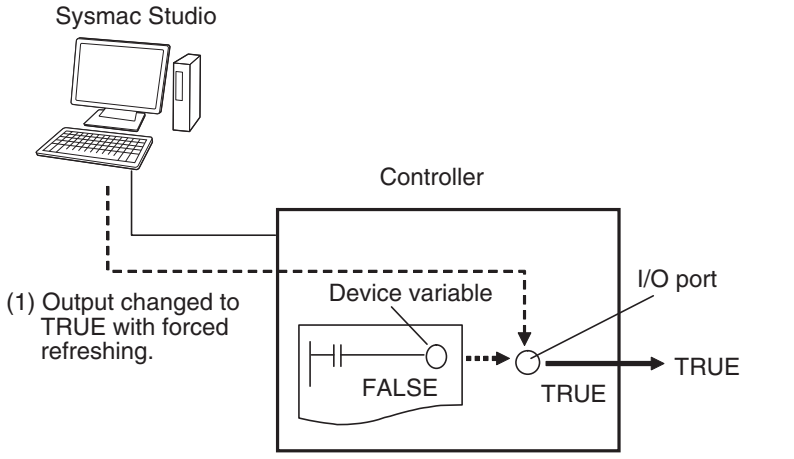
The I/O port and device variable change to the status that is specified with forced refreshing regardless of the status of the external input.



(2) I/O port and device variable change to TRUE.

● 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.



(2) The I/O port and the output to the external device change to TRUE. The device variable changes with the result of user program execution.

Applicable Areas

You can execute forced refreshing for the following I/O ports.

- I/O ports for EtherCAT slaves

Number of Simultaneous I/O for Forced Refreshing

The number of variables that you can refresh with forced refreshing is listed below.

- 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.

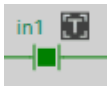
- Program Panes (Ladder diagram language)
- I/O Map
- Watch Tab Page

● 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

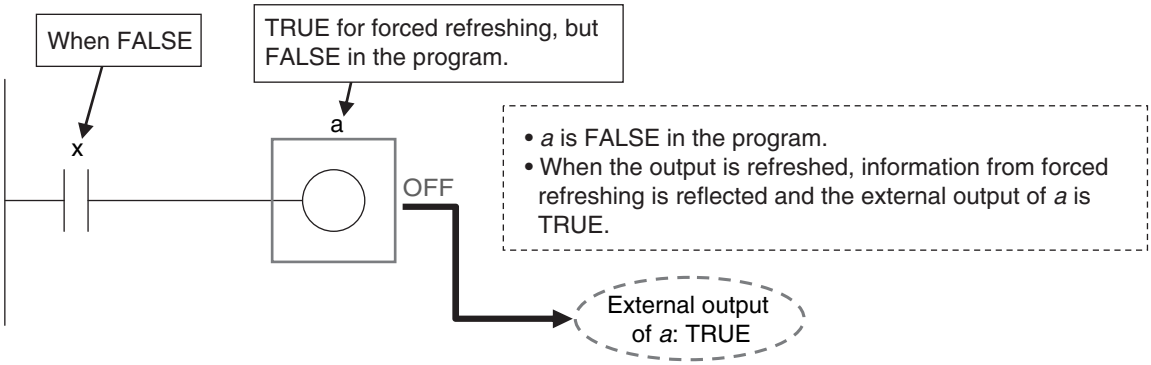
Forced refreshing is retained and cleared according to changes in the status of the Controller as shown below.

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

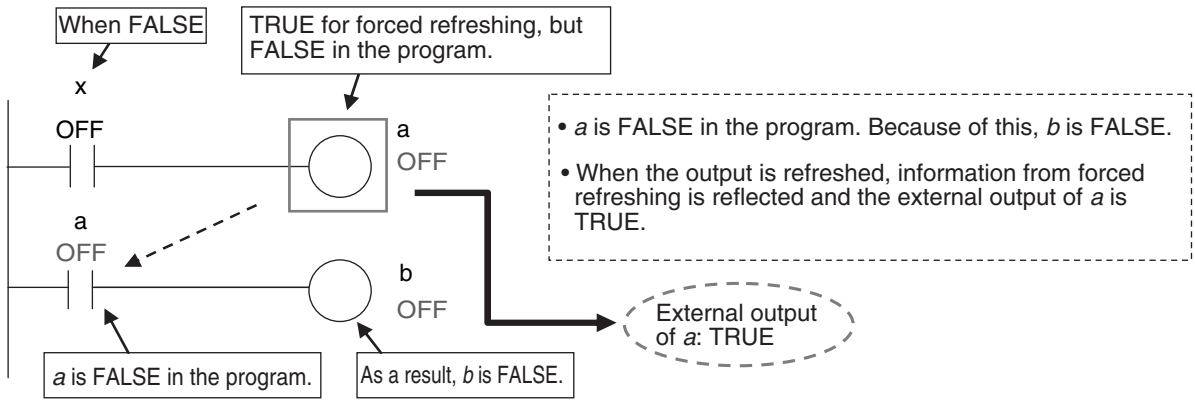
Programming Precautions for Forced Refreshing

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 Correct Use

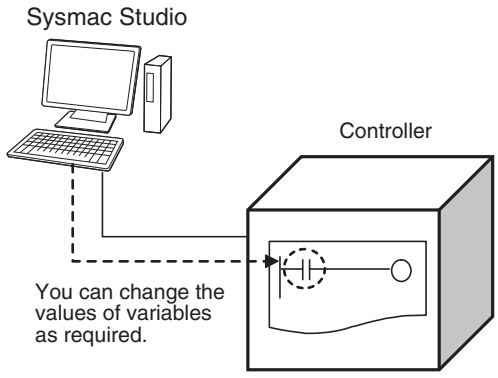
- Confirm that no adverse effect will occur in the system before you use forced refreshing.
- Forced refreshing ignores the results of user program execution and refreshes I/O with the specified values. If forced refreshing is used for inputs for which I/O refreshing is not supported, the inputs will 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.

8-4-2 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 Correct 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 Correct 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 an 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 an EtherCAT output slave.

8-4-3 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 Controller 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 Controller.

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 dialog box that indicates saving data to built-in non-volatile memory (which is displayed after the confirmation dialog box) closes.

8-4-4 Data Tracing

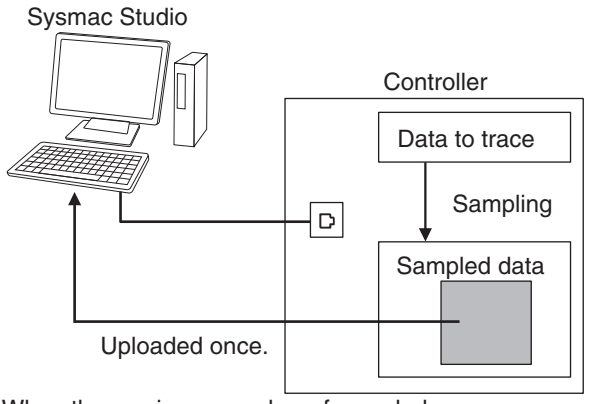
You can use data tracing to sample variables without any additional programming. You can read and check the data from the Sysmac Studio, and save the data to a file. This is used to start up, operate, and maintain devices.
This section introduces data tracing. Refer to the *Sysmac Studio Version 1 Operation 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.

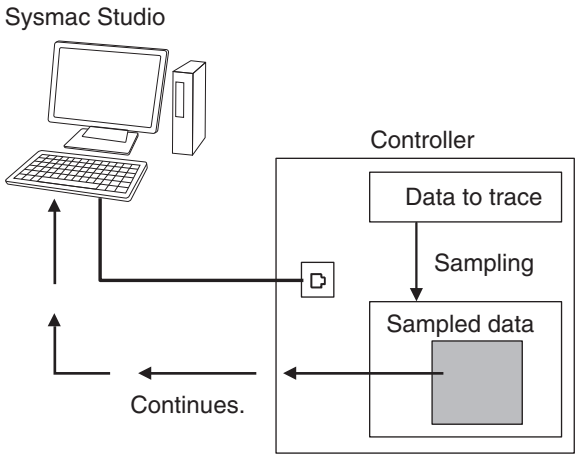
- You can check the flow of the program based on the status of changes in the present values of variables.
- You can use the data to investigate the cause of unexpected changes in the values of variables.



When the maximum number of sampled variables is reached, the trace stops and the trace data is sent to the Sysmac Studio and displayed.

● 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**	192 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		4 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.

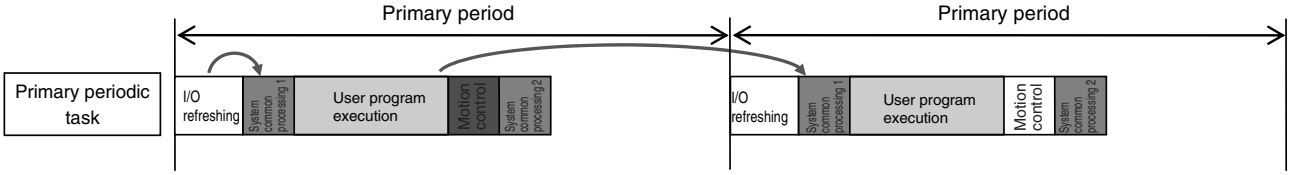
*3 Combinations of multiple condition expressions are not permitted. Also, the valid range for comparison constants is determined by the valid range of the literal expressions for the variable type on the left side of the condition expression.

*4 Trace numbers 0 to 3 are set for the NY-series Controllers. 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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561) are based on the task periods, so the display are not the same as those for data tracing.

Example 1:

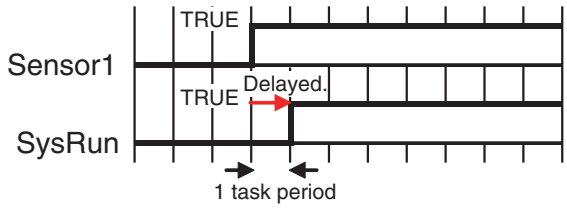
In this example, the *SysRun* variable is changed to TRUE in the user program when the *Sensor1* variable (assigned to the sensor input signal) changes to TRUE.

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

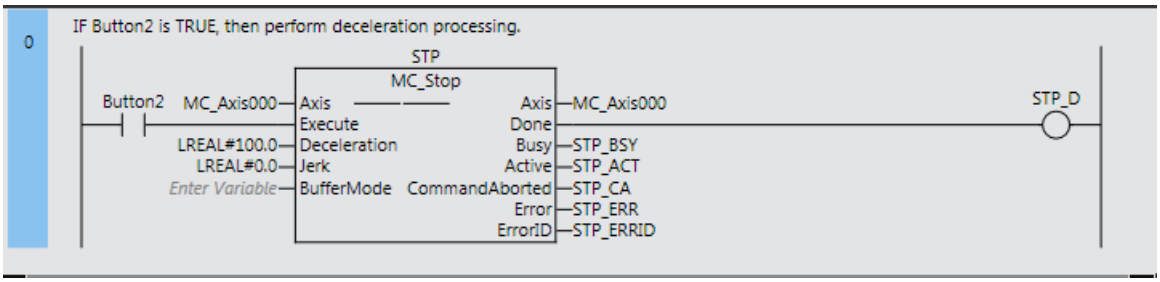


Additional Information

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:

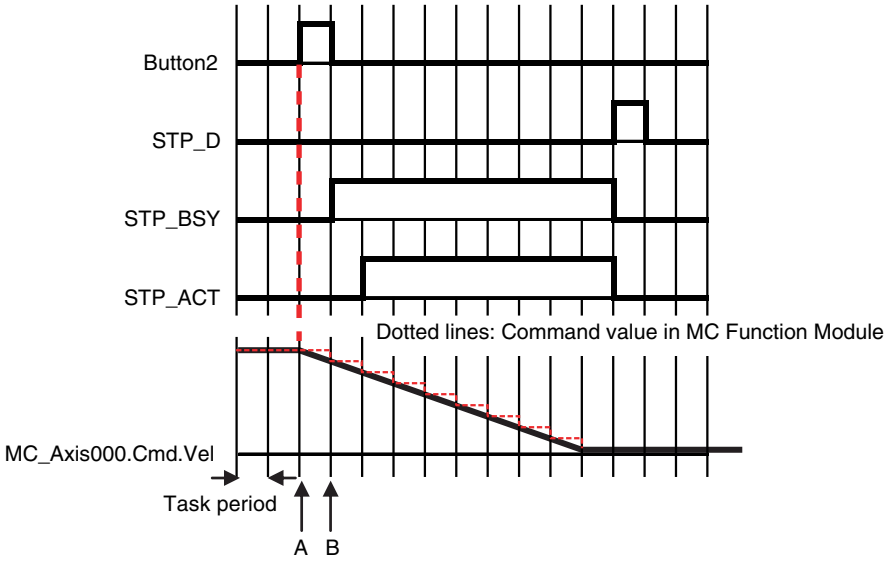
When the *Button2* variable (assigned to an input signal from a pushbutton) changes to TRUE during velocity control, the user program in this example decelerates axis 0 (*MC_Axis000*) to a stop.



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

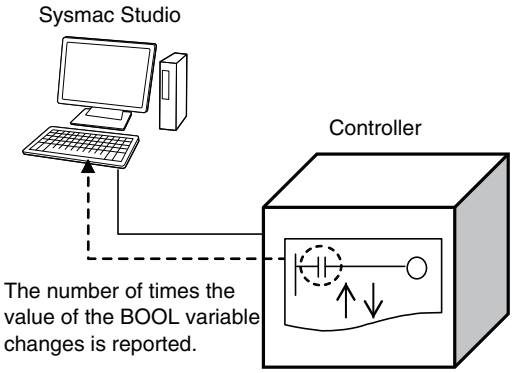
Variable name		Meaning	Description	Data type	R/W
	Member				
_PLC_TraceSta[0..3] *				_sTRACE_STA	R
	.IsStart	Trace Busy Flag	TRUE when a trace starts.	BOOL	R
	.IsComplete	Trace Completed Flag	TRUE when a trace is completed. Changes to FALSE when the next trace starts.	BOOL	R
	.IsTrigger	Trace Trigger 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.

8-4-5 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.



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

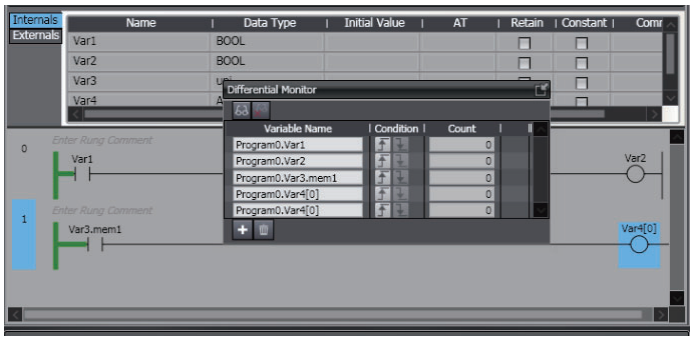
Procedure

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.

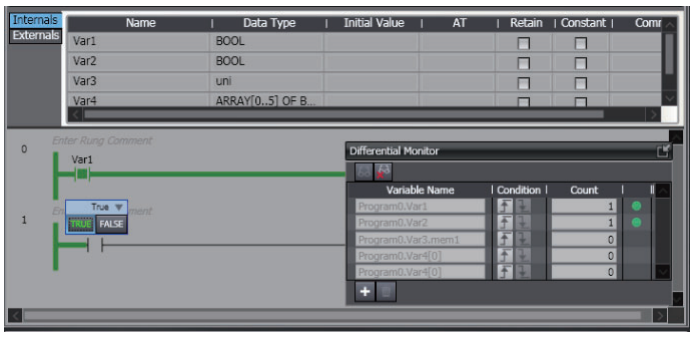
2 Right-click a variable that can be specified for differential monitoring and select **Add Differential Monitor.**

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 Controller 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

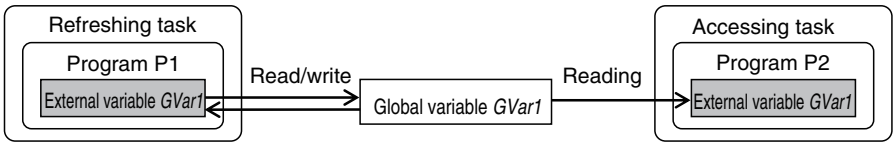
You cannot run differential monitoring from more than one copy of the Sysmac Studio running on the same computer or from the Sysmac Studio running on different computers.

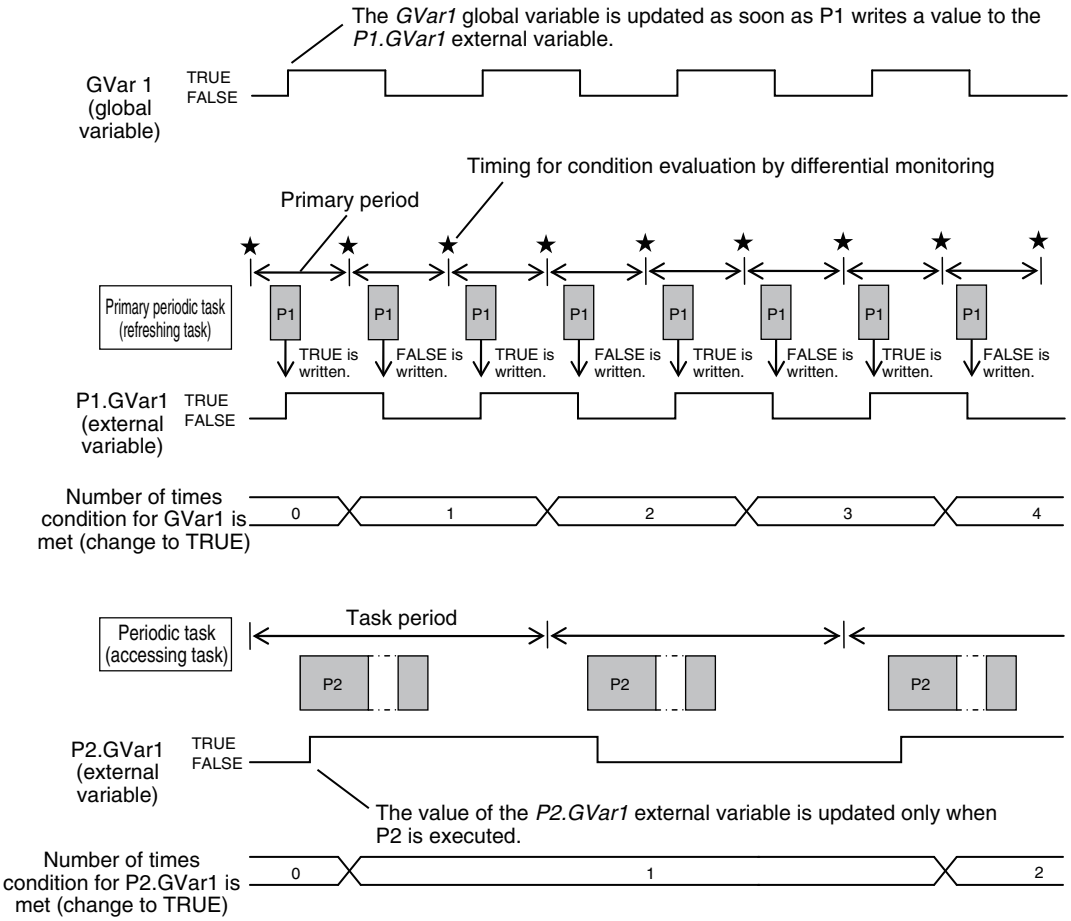
● Specifying Global Variables and External Variables

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 Controller 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 Controller 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

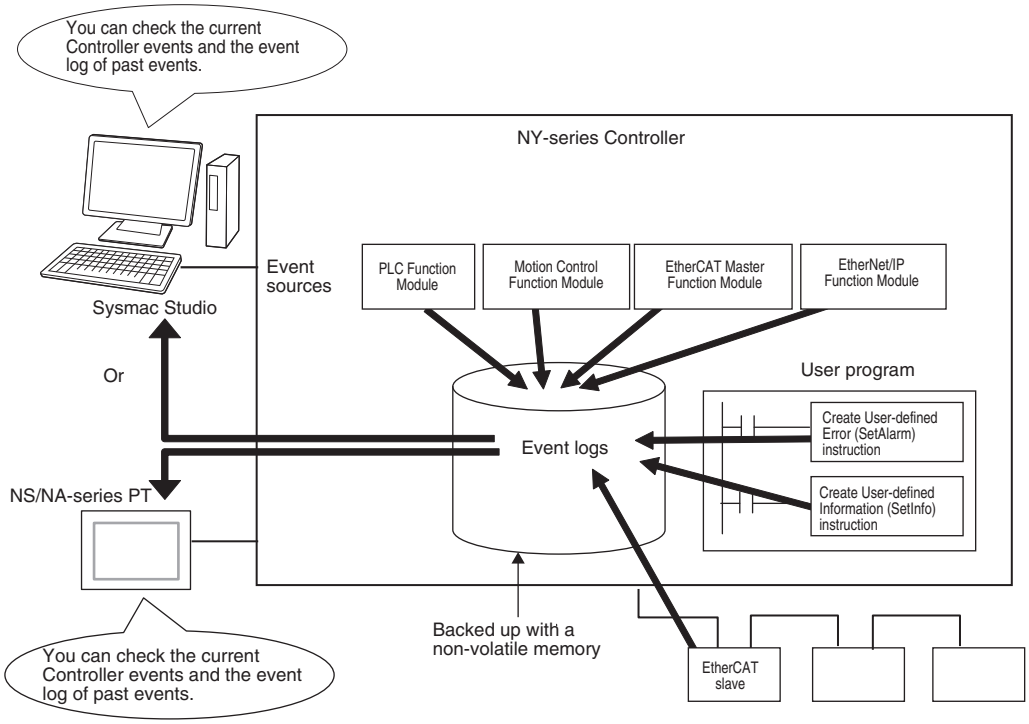

The GVar1 global variable is updated as soon as P1 writes a value to the P1.GVar1 external variable.
GVar 1 (global variable)
TRUE
FALSE
Timing for condition evaluation by differential monitoring
Primary period
Primary periodic task (refreshing task)
P1
TRUE is written.
FALSE is written.
P1.GVar1 (external 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)

8-5 Event Logs

This section describes the event logs.

8-5-1 Introduction

The event logs contain records of events,* such as errors, status changes, and user-defined events, that occurred in the NY-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/NA-series PT to check events, connect the PT to the built-in EtherNet/IP port on the Controller.

Precautions for Correct Use

- Refer to the *NY-series Troubleshooting Manual* (Cat. No. W564) 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 NY-series Controller 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.

8-5-2 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 Controller.

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	Instructions or Windows
Motion Control Function Module	Common, axis, or axes group
EtherCAT Master Function Module	Communications port, EtherCAT master, EtherCAT Coupler Unit, NX Unit, or EtherCAT slave
EtherNet/IP Function Module	Communications port, communications port 1, internal port 1, CIP, FTP, NTP, or SNMP

● 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.
	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 Controller overwrites the oldest events.

Event type	Event log category	Maximum number of records
Controller events	System log	2,048 events
	Access log	
User-defined events	User event log	

Retaining Events during Power Interruptions

The NY-series Controller uses a non-volatile memory to retain the event logs when the power is interrupted.

Precautions for Correct Use

The event logs are retained with a non-volatile memory. They are not retained when a UPS is not connected to the Industrial PC or when the Industrial PC was not normally shut down.

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 NY-series System.

The event levels are defined for user-defined events.

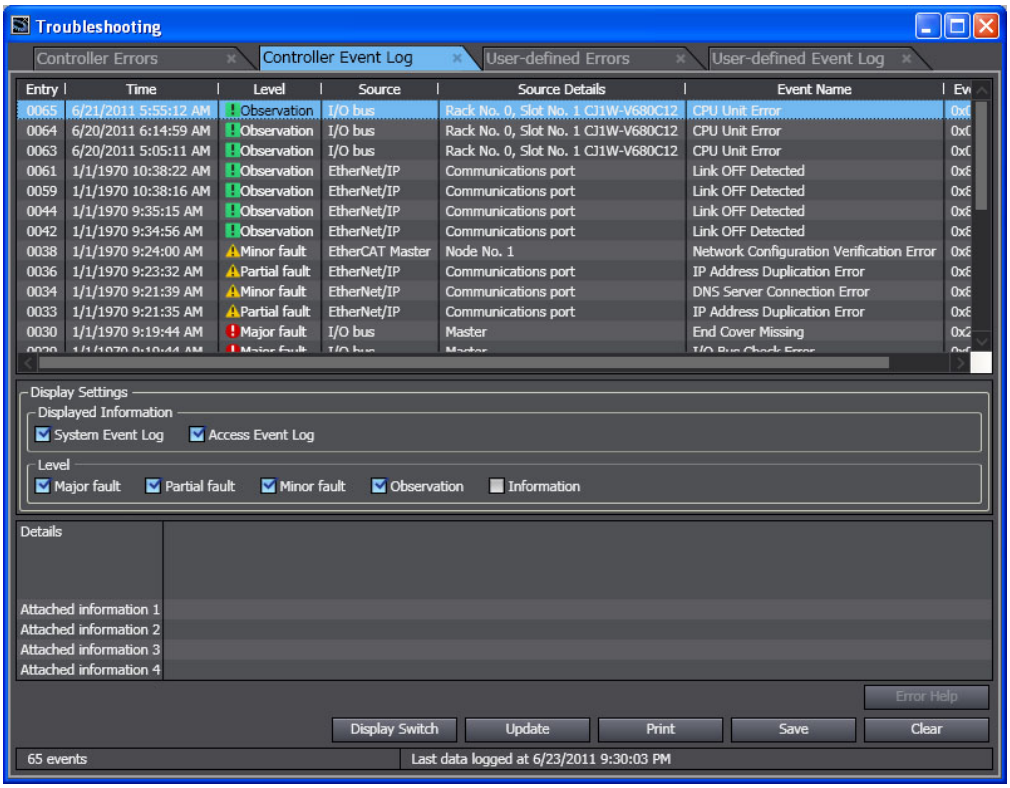
No.	Level	Type	Meaning
1	High	User fault Level 1	These event levels indicate a user-defined error in an application. The user executes the SetAlarm (Create User-defined Error) instruction to create the event.
2	▲	User fault Level 2	
3	|	User fault Level 3	
4	|	User fault Level 4	
5	|	User fault Level 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 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 Controller 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 *NY-series Troubleshooting Manual* (Cat. No. W564) 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 NY-series Controller 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.

8-5-3 Controller Events (Controller Errors and Information)

Introduction

Controller errors and information are defined by the NY-series System.

These events occur when the NY-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 NY-series Controller and EtherCAT slaves are detected. When one of these events occurs, a Controller error is recorded in the event log.

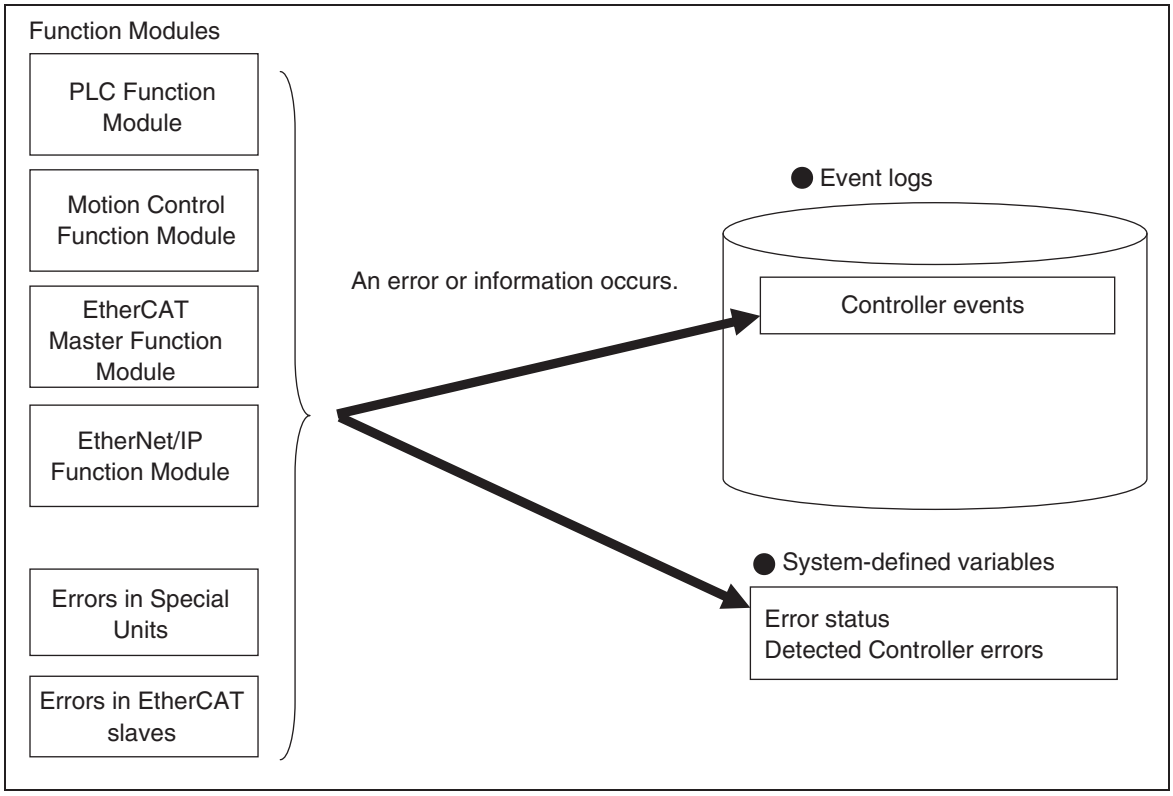
To check the status of a Controller error on the user program, you execute the Get Error Status instruction to access the status of the Error Status variable, which is a system-defined variable.

Controller errors are not reset when the operating mode changes.

Refer to the *NY-series Troubleshooting Manual* (Cat. No. W564) 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.



8-5-4 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.

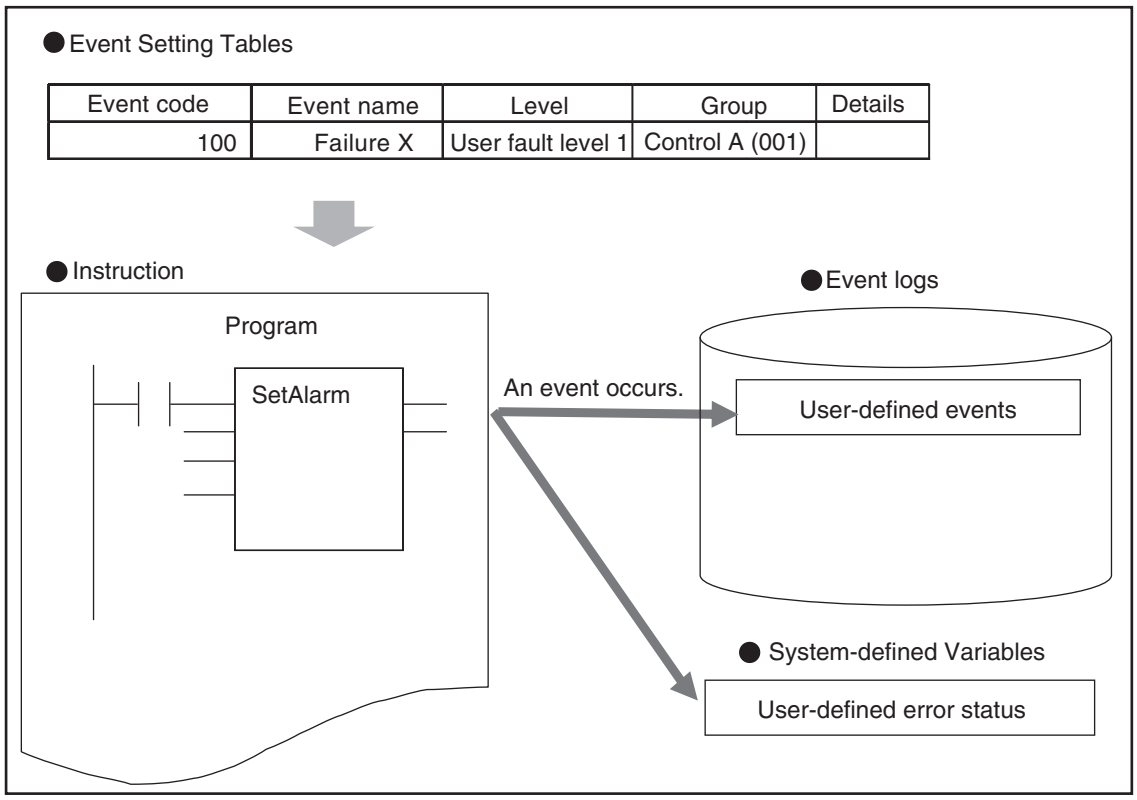
● User-defined Errors

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.



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.

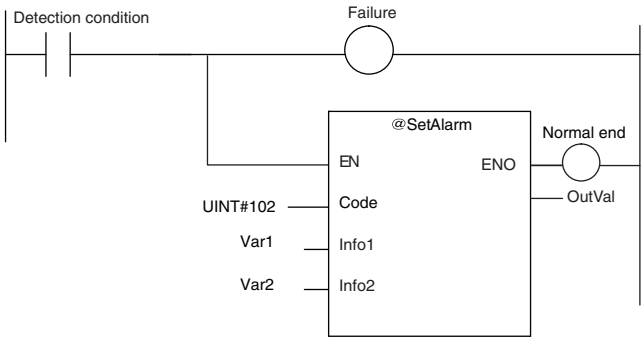
Event Setting Table

Event Setting Table

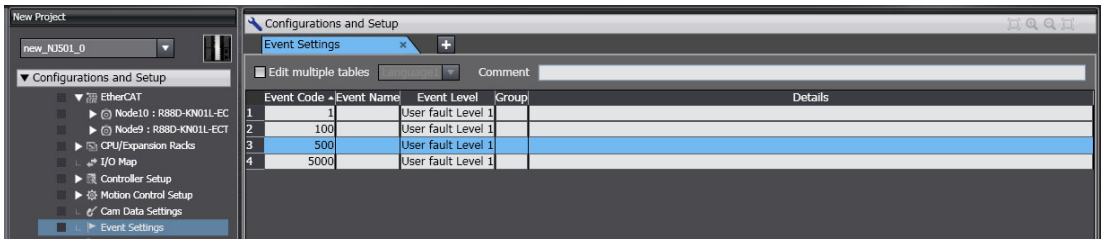
Event code	Event name	Level	Group	Details
10001	Failure X	User fault Level3	Control A (001)	
:	:	:	:	:

Details
■ Description Failure X occurred. ■ Correction Perform safety checks and handle the problem according to the cause code.

Programming Example



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 NY-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 Controller, 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.

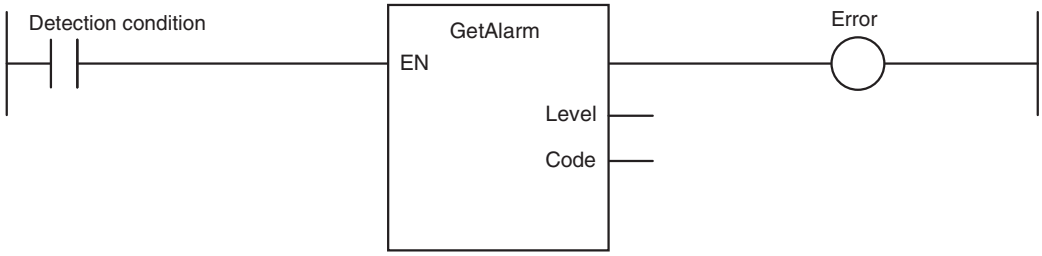
Example:

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 NY-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.

8-6 Changing Event Levels

Errors, status changes, and user-defined events that occur in the NY-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 Industrial PC.

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 8-51 for details on event levels.

You can change the event levels that are assigned to some of the Controller events.

8-6-1 Applications of Changing Event Levels

The lighting pattern for the indicators on the Industrial PC 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 *NY-series Troubleshooting Manual* (Cat. No. W564) for details on how the Controller operates for different event levels.

8-6-2 Events for Which the Event Level Can Be Changed

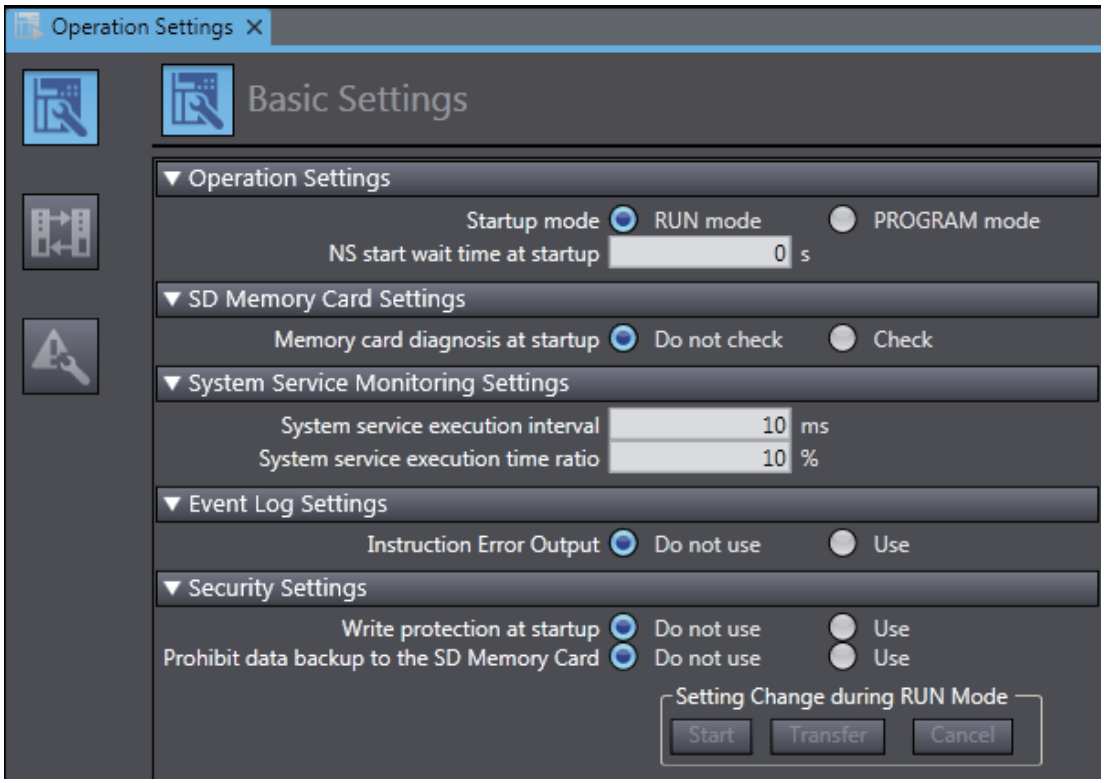
Whether an event level can be changed depends on the specific event.

Refer to the *NY-series Troubleshooting Manual* (Cat. No. W564) for details on the types and levels of the Controller events, and whether the event levels can be changed.

8-6-3 Procedure to Change an Event Level

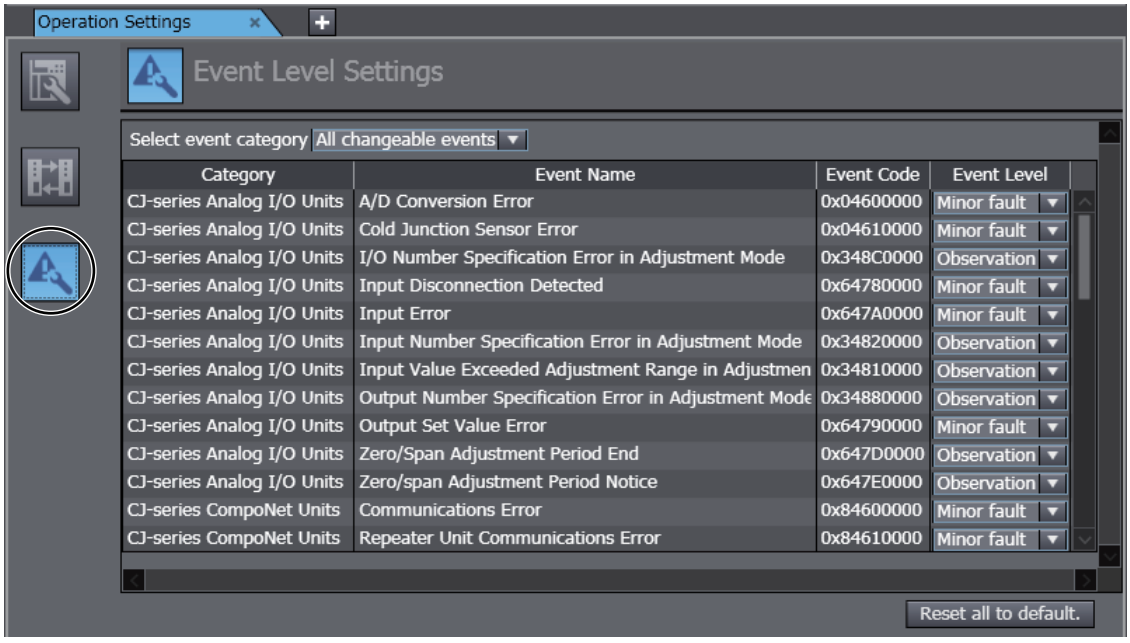
1 Double-click **Operation Settings** under **Configurations and Setup** – **Controller Setup** in the Sysmac Studio. Or right-click **Operation Settings** and select **Edit** from the menu.

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 NY-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 NY-series Controller.

Function	Description
Backing up data	You can back up all of the data in the Controller to a Virtual 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 Virtual 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 Virtual 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 Industrial PCs, 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 a 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 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 User program execution ID in user program POUs Data (data types and global variables)
IP address of built-in EtherNet/IP port*1	Of the TCP/IP Settings in the Built-in EtherNet/IP Port Settings, setting type, IP address, subnet mask, and default gateway
Present values of variables	Values of variables with a Retain attribute*2
Units and slaves settings	Backup parameters for EtherCAT slaves*3
Absolute encoder home offset	The set value to restore the actual position of a Servo Drive with an absolute encoder

*1. With a combination of the Controller with unit version 1.14 or later and Sysmac Studio version 1.18 or higher, IP address of the Built-in EtherNet/IP Port Settings can be used as a data group. IP address is included in the user program and settings other than the above combination.

*2. Of the system-defined variables with a Retain attribute, some variables are not applicable for the data backup function. Refer to *A-4 Specifications for Individual System-defined Variables* for details on the specifications for individual system-defined variables.

*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 *9-7-4 EtherCAT Slaves for Which You Can Back Up Data*.

Precautions for Safe Use

Precautions on the Absolute Encoder Home Offset

The absolute encoder home offsets are backed up with a non-volatile memory 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.

9-1-3 Types of Backup Functions

There are backup functions for the NY-series Controllers that save data to Virtual SD Memory Cards and others that save data to a computer. Also, there are three methods used to execute the backup functions: the system-defined variables, the Sysmac Studio, and the Industrial PC Support Utility.

Functions That Save Data to Virtual SD Memory Cards

The SD Memory Card backup functions are used to back up and compare data on Virtual SD Memory Cards. The related function includes disabling backups to SD Memory Cards.

Function name		Description	Operating method		Reference
			System-defined variables	Sysmac Studio	
SD Memory Card backups	Backing up data	The Controller data is saved in a backup file on the Virtual SD Memory Card.	✓	✓	*9-2-1 Backup (Controller to Virtual SD Memory Card)*
	Verifying data	The Controller data and the data in a backup file on the Virtual SD Memory Card are compared.	✓	✓	*9-2-2 Verify (between Controller and Virtual SD Memory Card)*
Disabling backups to SD Memory Cards		You can disable backing up data to Virtual SD Memory Cards.		✓	*9-3 Disabling Backups to SD Memory Cards*

Additional Information

- To restore the backup files that are saved on the Virtual SD Memory Card to the Controller, specify the shared folder being set on the Virtual SD Memory Card with the restore operation of the Controller backup functions for Industrial PC Support Utility.
- You can also restore with the Sysmac Studio Controller backup functions. In this case, you need to use an external storage to save the backup files, which are saved in the shared folder being set on the Virtual SD Memory Card, in the computer that the Sysmac Studio is installed.

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 the present values saved on a non-volatile memory to the computer and restore them from the computer.

Function name		Description	Operating method		Reference
			System-defined variables	Sysmac Studio	
Sysmac Studio Controller backups	Backing up data	The Controller data is saved in a backup file on the computer.		✓	*9-4-1 Backup (Controller to Computer)*
	Restoring data	The data in a backup file on the computer is transferred to the Controller.		✓	*9-4-2 Restore (Computer to Controller)*
	Verifying data	The Controller data and the data in a backup file on the computer are compared.		✓	*9-4-3 Verify (between Controller and Computer)*
Sysmac Studio 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.		✓	*9-5 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 saved on a non-volatile memory to an XML file on the computer.		✓	*9-6 Sysmac Studio Variable and Memory Backup Functions*
	Restoring data	You can restore the present values of data that is saved on a non-volatile memory from the computer to the Controller.		✓	

Functions That Save Data to the Folder in Windows

The Controller backup functions for Industrial PC Support Utility are used to back up, restore, and compare data on the folder in Windows.

Refer to the *NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat. No. W568) for details on the Controller backup functions for Industrial PC Support Utility.

9-1-4 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 given in the following table.

(OK: Applicable, NA: Not applicable)

Type of backup function		Data group				
		User program and settings	IP address of built-in EtherNet/IP port*1	Present values of variables	Units and slaves settings	Absolute encoder home offsets
SD Memory Card backups	Backing up data	OK	OK	OK*2	OK	OK
	Restoring data*3	OK	OK	OK*2	OK	OK
	Verifying data*3	OK*4	OK	NA	OK	NA
Sysmac Studio Controller backups	Backing up data	OK	OK	OK*2	OK*5	OK
	Restoring data	OK	OK	OK*2	OK*5	OK
	Verifying data	OK*4	OK	NA	OK*5	NA
Controller backups for Industrial PC Support Utility	Backing up data	OK	OK	OK*2	OK*5	OK
	Restoring data	OK	OK	OK*2	OK*5	OK
	Verifying data	OK*4	OK	NA	OK*5	NA
Importing and exporting Sysmac Studio backup file data	Exporting backup file data	OK*6	OK	NA	NA	NA
	Importing backup file data	OK*6	OK	NA	OK	NA
Sysmac Studio variable and memory backup functions	Backing up and restoring data	NA	NA	OK*2	NA	OK

*1. With a combination of the Controller with unit version 1.14 or later and Sysmac Studio version 1.18 or higher, IP address of the Built-in EtherNet/IP Port Settings can be used as a data group. IP address is included in the user program and settings other than the above combination.
*2. The backup data is processed only for the present values of variables that are specified for retention with the Retain attribute.
*3. 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.
*4. Of the user program and setting data groups, the Data Trace Settings are not compared.
*5. If the EtherCAT slaves are specified for the backup, parameters for the EtherCAT slaves are backed up.
*6. 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 *9-6 Sysmac Studio Variable and Memory Backup Functions* for details on the Sysmac Studio variable and memory backup functions.

9-1-5 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)

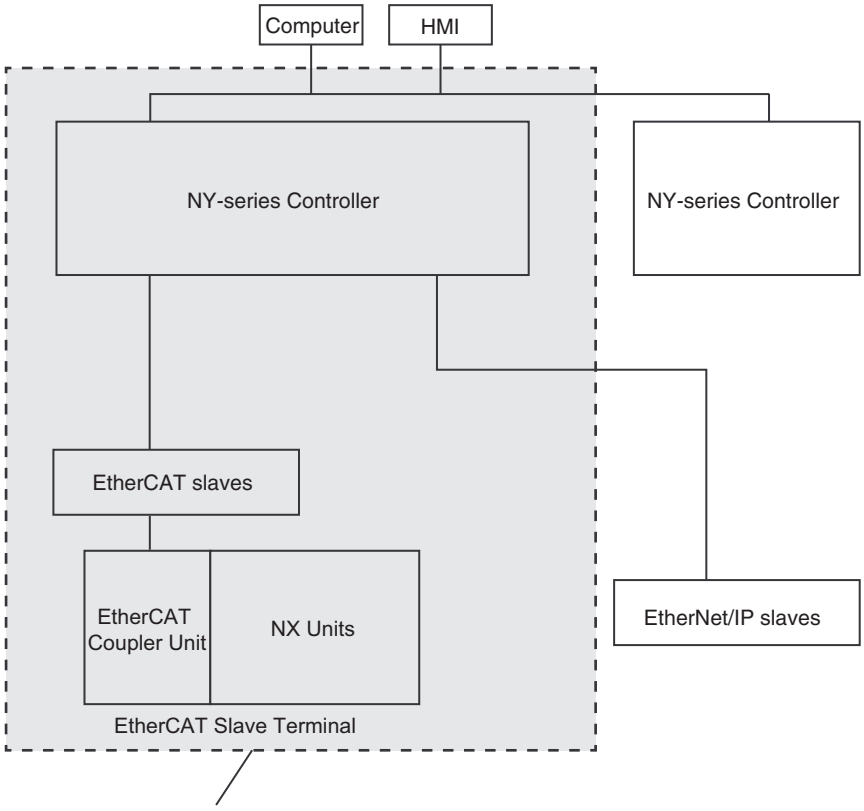
Type of backup function	Units/slaves			
	NY-series Controller	EtherCAT slaves*1	EtherNet/IP slaves	Computer and HMIs
SD Memory Card backups	OK	OK*2	NA	NA
Sysmac Studio Controller backups	OK	OK*2	NA	NA
Importing and exporting Sysmac Studio backup file data	OK	OK*3	NA	NA
Sysmac Studio variable and memory backup functions	OK	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. This does not apply to Safety Control Units. Refer to the *NX-series Safety Control Units User's Manual* (Cat. No. Z930) for information on importing and exporting settings for a Safety Control Unit.

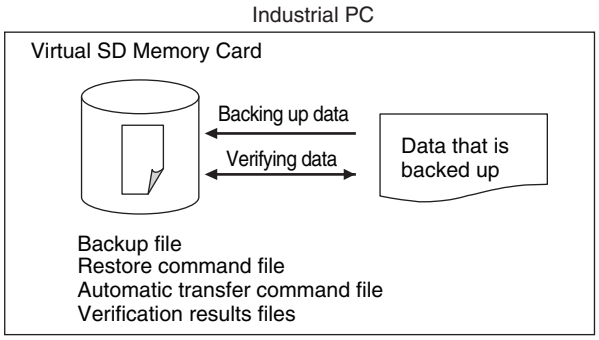
*3. Only importing data is possible. Exporting is not possible.

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.



9-2 SD Memory Card Backups

You can use Virtual SD Memory Cards to back up 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 Virtual 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	Verifying data
Backup file	This file contains the Controller data that is handled by the functions that are related to data backup.	Created.	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.
Automatic transfer command file	This file is not used in an NY-series Controller.	Created.	Nothing is done.
Verification results file	This file contains the verification results after data is verified.	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
System-defined variables*2*3	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 Memory 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 Virtual SD Memory Card.
*2. This method is used to control the backup functions from an HMI. You cannot access these system-defined variables from the user program.
*3. 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 Virtual SD Memory Card)

This operation is used to save data in the Controller to the Virtual SD Memory Card.

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 Virtual 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 Virtual 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 Virtual SD Memory Card will remain the recognition after completion of the backup.

Procedure

● 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.

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-4 Specifications for Individual System-defined Variables* for details on system-defined variables.

Variable Member name	Meaning	Function	Data type	R/W
_Card1BkupCmd*1	SD Memory Card Backup Commands		_sBKUP_CMD	RW
ExecBkup*1	Execute Backup Flag	Change this variable to TRUE to back up Controller data to a Virtual SD Memory Card.	BOOL	RW
CancelBkup*1	Cancel Backup Flag	Change this variable to TRUE to cancel backing up data to a Virtual SD Memory Card.	BOOL	RW
DirName*1	Directory Name	Use this variable to specify the directory name in the Virtual SD Memory Card for which to back up data.	STRING(64)	RW
_Card1BkupSta*1	SD Memory Card Backup Status		_sBKUP_STA	R
Done*1	Done Flag	TRUE when a backup is completed.	BOOL	R
Active*1	Active Flag	TRUE when a backup is in progress.	BOOL	R
Err*1	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

*1. You cannot access these system-defined variables from the user program.

Additional Information

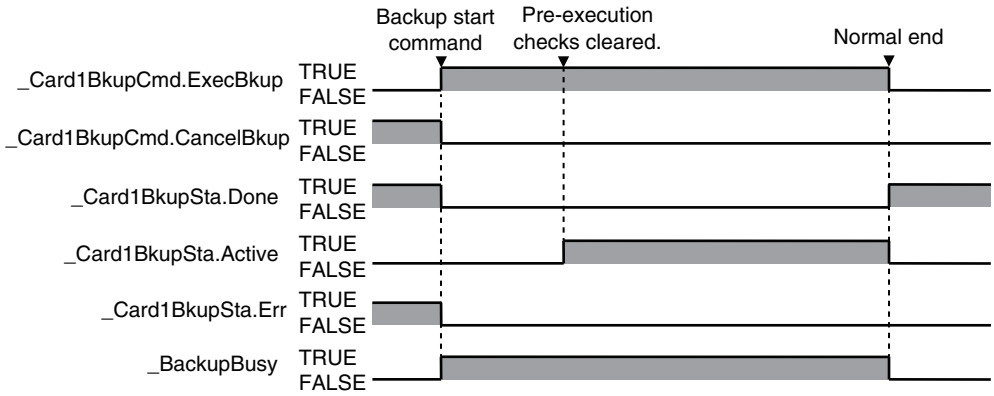
- Refer to the *NA-series Programmable Terminal Software User's Manual* (Cat. No. V118) for information on mapping variables when you connect an NA-series PT to the NY-series Controller.
- Refer to *A-7 Registering a Symbol Table on the CX-Designer* for the procedure to register these system-defined variables in the variable table of the CX-Designer when you connect an NS-series PT to the NY-series Controller.

Timing Charts

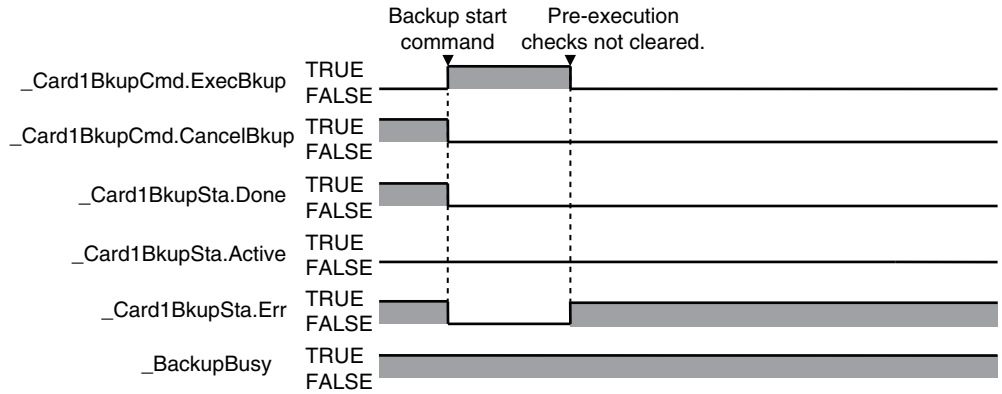
The operation of the system-defined variables when they are used to backup data is shown below.

In the charts, “pre-execution checks” indicates processing to check whether there is a Virtual SD Memory Card 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.

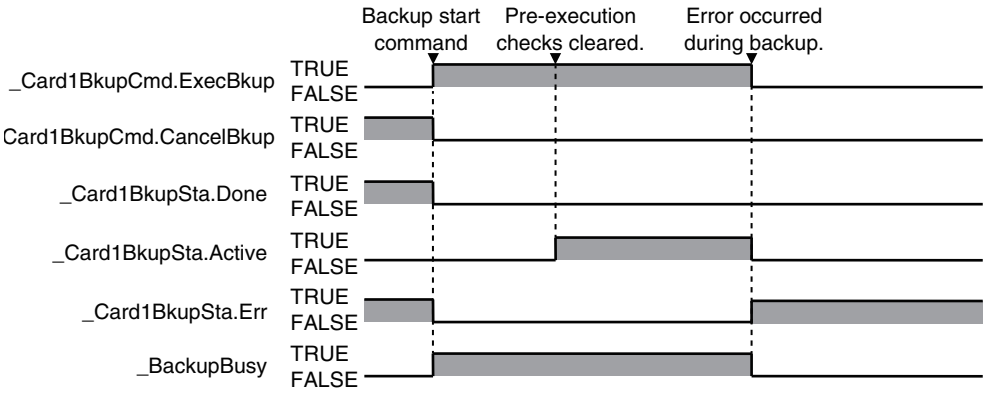
● Normal Operation



● Operation When the Backup Cannot Start Because Another Backup Function Is in Progress

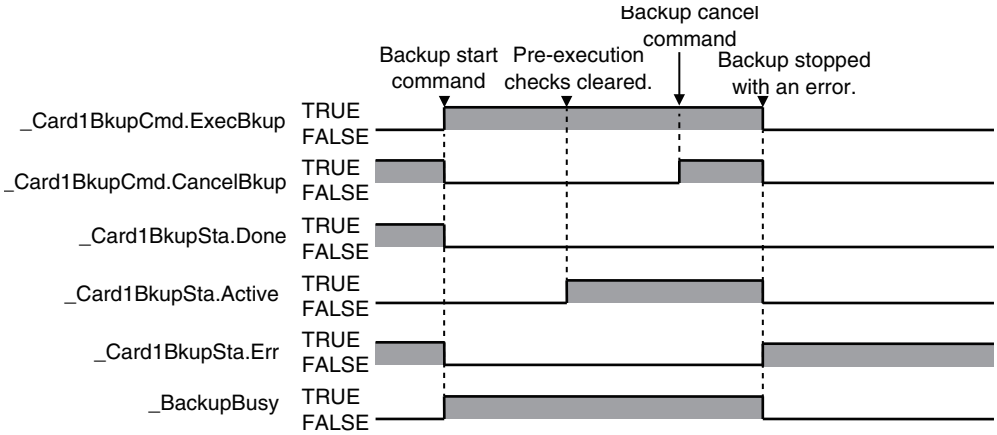


● Operation When the Backup Fails After a Normal Start



● 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 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.

Controller	Operating mode	Connected EtherCAT slaves	Number of user-defined POUs	User program memory size (Mbytes)	Backup time (s)	Backup file size (Mbytes)
NY5□2-□□□□	PROGRAM mode	*1	72	3.82	Approx. 155	20.27

*1. Thirty-two R88D-KNA-ECT AC Servo Drives and five NX-ECC201 EtherCAT Coupler Units. The EtherCAT Coupler Unit consists of the following NX Units.
Five each of the NX-PD1000, NX-AD4608, and NX-DA3605. Sixty-four each of the NX-ID3343 and NX-OD3153.

9-2-2 Verify (between Controller and Virtual SD Memory Card)

You can compare the Controller data and the data in a backup file on the Virtual SD Memory Card.

Processing Contents

- The Controller data and the data in a backup file that is saved in the specified directory of the Virtual 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 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.
- If there is not a restore command file in the specified directory of the Virtual 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 Virtual SD Memory Card will remain the recognition after completion of the backup.

Procedure

● *_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) changes to TRUE and the *_Card1BkupSta.VefyRslt* (Verify Result Flag) system-defined variables 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.
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 verify data are shown below. Refer to *A-4 Specifications for Individual System-defined Variables* for details on system-defined variables.

Variable name / Member name	Meaning	Function	Data type	R/W
_Card1BkupCmd*1	SD Memory Card Backup Commands		_sBKUP_CMD	RW
ExecVefy*1	Execute Verify Flag	Change this variable to TRUE to compare the Controller data to a backup file in the Virtual SD Memory Card.	BOOL	RW
CancelVefy*1	Cancel Verify Flag	Change this variable to TRUE to cancel comparing the Controller data to a backup file in the Virtual SD Memory Card.	BOOL	RW
DirName*1	Directory Name	Use this variable to specify the directory name in the Virtual SD Memory Card for which to back up data.	STRING(64)	RW
_Card1VefySta*1	SD Memory Card Verify Status		_sVEFY_STA	R
Done*1	Done Flag	TRUE when a verification is completed.	BOOL	R
Active*1	Active Flag	TRUE when a verification is in progress.	BOOL	R
VefyRslt*1	Verify Result Flag	TRUE if the data was the same. FALSE if differences were found.	BOOL	R
Err*1	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

*1. You cannot access these system-defined variables from the user program.

Additional Information

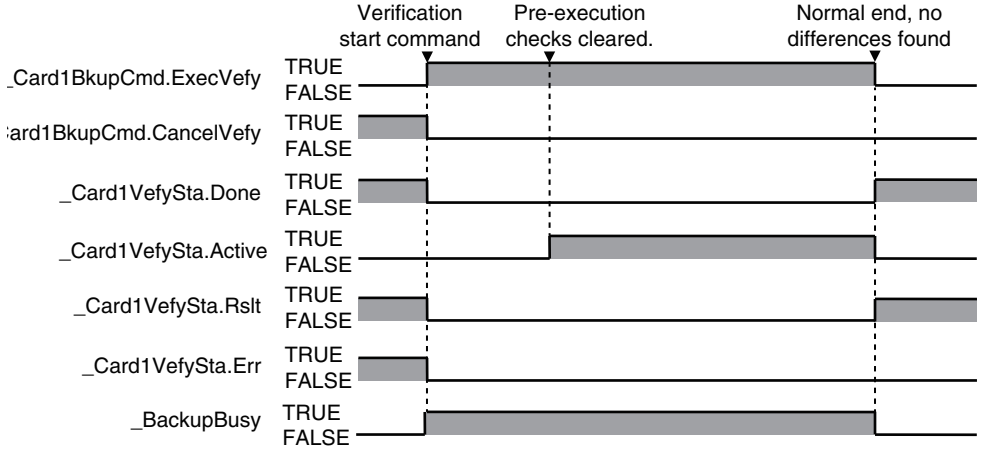
- Refer to the *NA-series Programmable Terminal Software User's Manual* (Cat. No. V118) for information on mapping variables when you connect an NA-series PT to the NY-series Controller.
- Refer to *A-7 Registering a Symbol Table on the CX-Designer* for the procedure to register these system-defined variables in the variable table of the CX-Designer when you connect an NS-series PT to the NY-series Controller.

Timing Charts

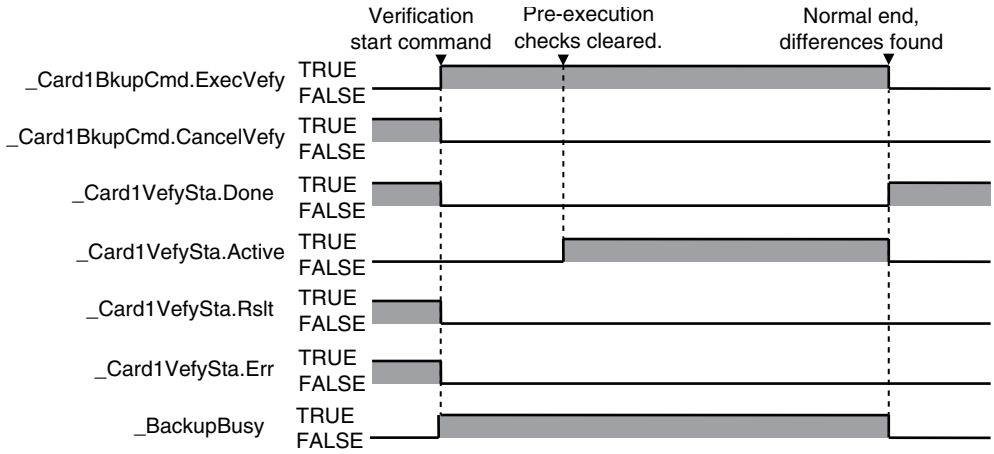
The operation of the system-defined variables when they are used to verify data is shown below.

In the charts, “pre-execution checks” indicates processing to check whether there is a Virtual SD Memory Card 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.

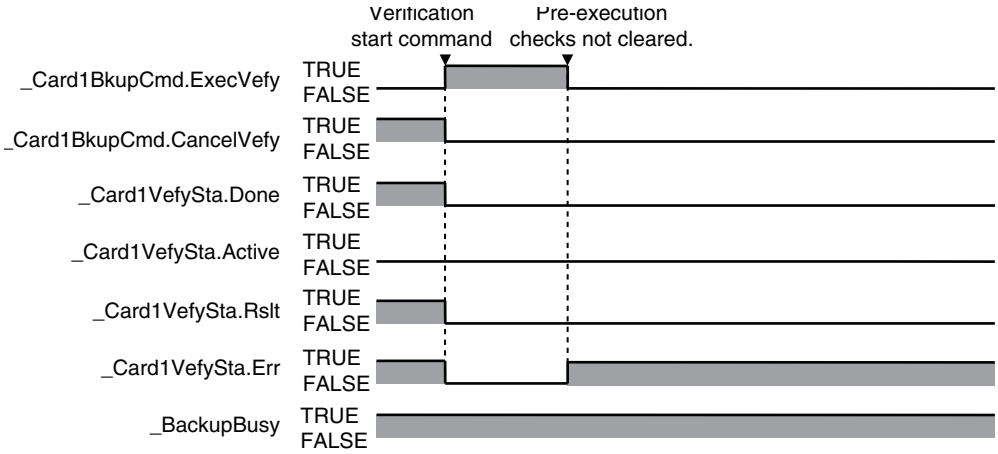
● Normal End with No Differences Found



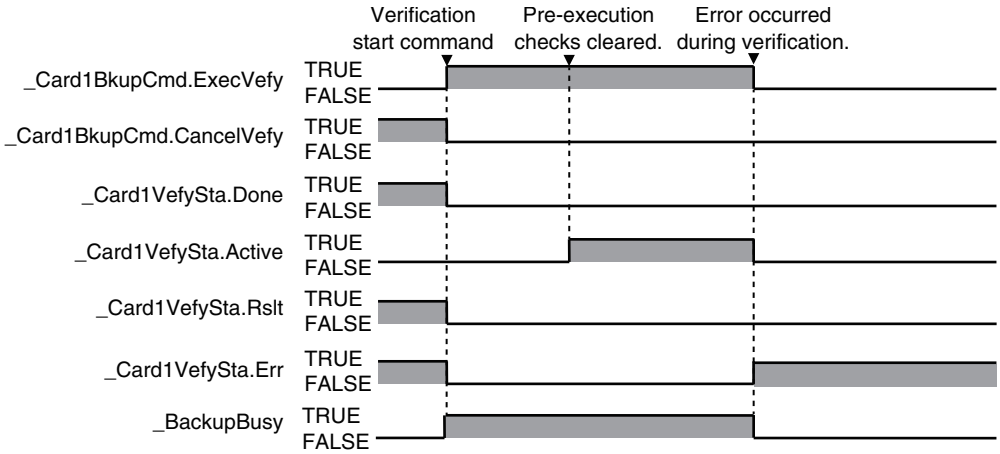
● Normal End with Differences Found



● Operation When the Verification Cannot Start Because Another Backup Function Is in Progress

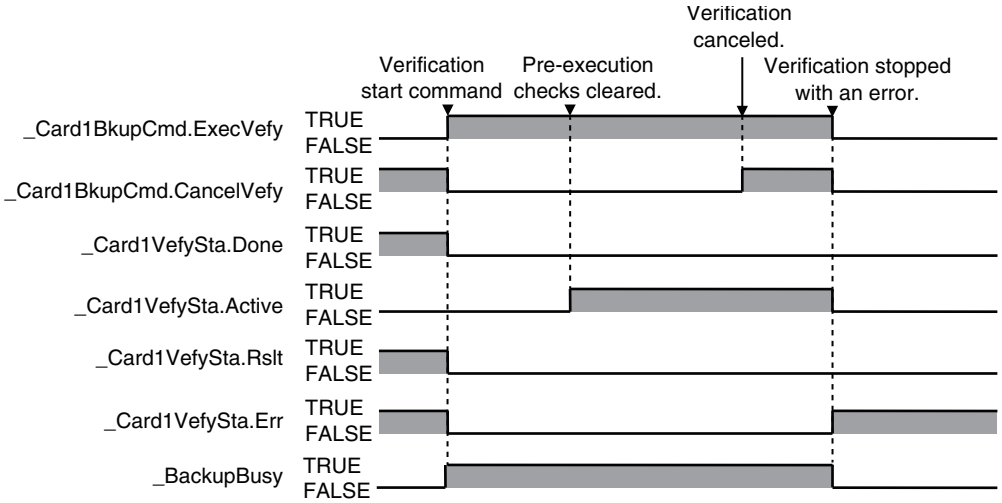


● Operation When the Verification Fails After a Normal Start



● 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

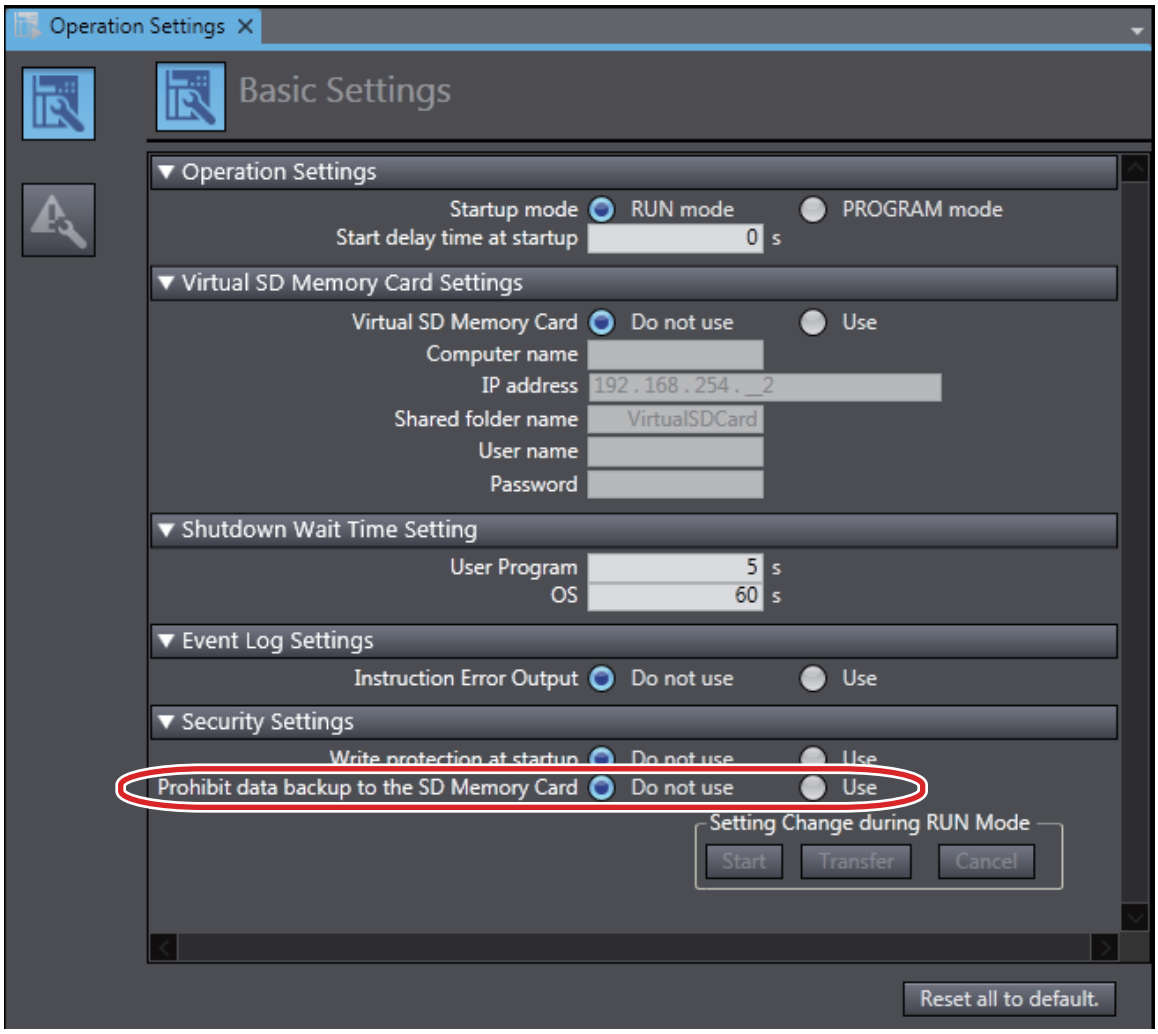
You can disable the backup function from writing data to the Virtual SD Memory Card to protect your programming assets.

The following two functions are applicable for disabling backup to the Virtual SD Memory Card.

- Backups using system-defined variables
- Backups from the SD Memory Card Window on the Sysmac Studio

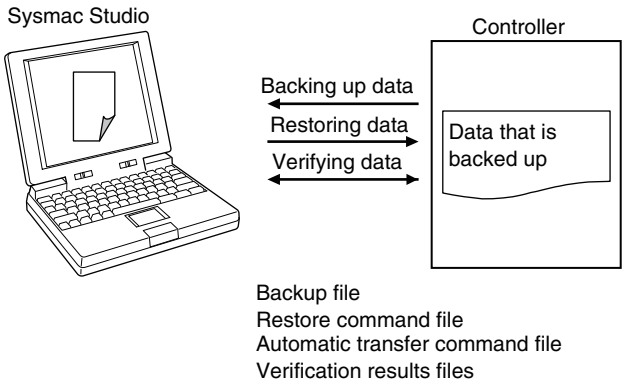
Backup function using the BackupToMemoryCard instruction is not applicable. This means that you can backup data using the BackupToMemoryCard instruction even if the *Prohibit data backup to the SD Memory Card* setting is set to be used.

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.



9-4 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	Restoring 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 is not used in an NY-series Controller.	Created.	Nothing is done.	Nothing is done.

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 Controller 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 Controller, or some other failure.

9-4-1 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.
- The backing up conditions for data groups are given in the following table.

Data group	Restoring condition
User program and settings	The CPU Unit must be selected.
IP address of built-in EtherNet/IP port*1	The CPU Unit must be selected.
Present values of variables	The CPU Unit must be selected.
Event logs	The CPU Unit must be selected.
Units and slaves settings	The EtherCAT slaves must be selected.
Absolute encoder home offsets	The CPU Unit must be selected.

*1. A Controller with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required.

- 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.

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.

9-4-2 Restore (Computer to Controller)

The data in a backup file in the specified directory on the computer is restored 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 restored 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.
IP address of built-in EtherNet/IP port*1	The IP address of built-in EtherNet/IP port must be selected.
Present values of variables	The present values of variables that are specified for retention with the Retain attribute must be selected.
Units and slaves settings	The EtherCAT slaves must be selected.
Absolute encoder home offsets	The absolute encoder home offsets must be selected.

*1. A Controller with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required.

- 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.

9-4-3 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.
IP address of built-in EtherNet/IP port*1	The IP address of built-in EtherNet/IP port must be selected.
Units and slaves settings	The EtherCAT slaves must be selected.

*1. A Controller with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required.

- 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.

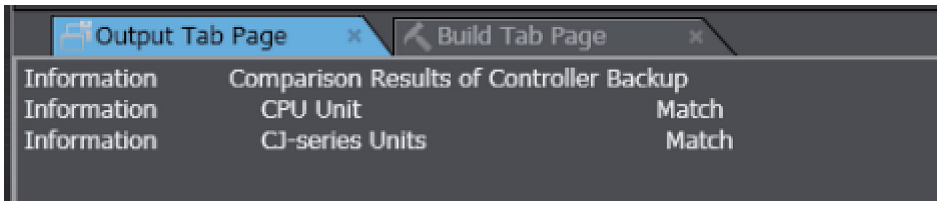
Procedure

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.



9-5 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	Units and slaves settings	Absolute encoder home offsets
			IP address of built-in EtherNet/IP port*1			
Importing and exporting Sysmac Studio backup file data	Exporting backup file data	OK*2	OK	NA	NA	NA
	Importing backup file data	OK*3	OK	NA	OK	NA

*1. With a combination of the Controller with unit version 1.14 or later and Sysmac Studio version 1.18 or higher, IP address of the Built-in EtherNet/IP Port Settings can be used as a data group. IP address is included in the user program and settings other than the above combination.

*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
The operation authority verification
Data Trace Settings

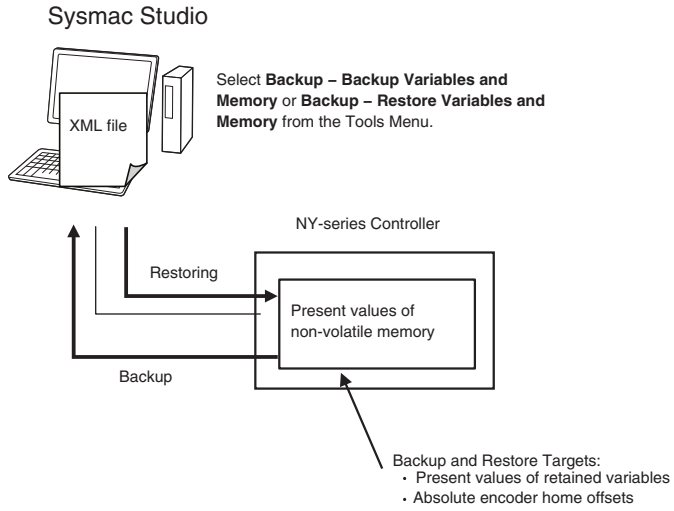
*3. The following data is not processed:
- Controller names: the built-in EtherNet/IP port name
- The operation authority verification
- Data Trace Settings

Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for more information on these functions.

9-6 Sysmac Studio Variable and Memory Backup Functions

You can back up the present values of the non-volatile memory in the Controller to an XML file on your computer or restore the non-volatile memory from a previously saved backup file.

This section describes the applicable data, operating procedures, and Controller model compatibility for the Sysmac Studio variable and memory backup functions.



9-6-1 Applicable Data for Sysmac Studio Variable and Memory Backup Functions

The applicable data for Sysmac Studio variable and memory backup functions are given below.

- Present values of variables with a Retain attribute
- Absolute encoder home offsets

9-6-2 Using Sysmac Studio Variable and Memory Backup Functions

The Sysmac Studio procedure is as follows:

Place the Sysmac Studio online with the Controller, 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.

9-6-3 Compatibility between NY-series Controller Models

With the Sysmac Studio variable and memory backup functions, you can restore the data even if the models of the NY-series Controller for backing up and restoring data are different.

9-7 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 *9-8 Backup Functions When EtherCAT Slave Terminals Are Connected*.

9-7-1 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 Slave 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 initial parameter setting function.

9-7-2 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.	Possible.	Possible.
	No disconnected slaves in actual configuration	Possible.	Possible.	Possible.
Slave disconnected for “Disconnect” designation in network configuration	Disconnected slaves in actual configuration	Not possible.*1	Possible. Data for disconnected slaves is also restored.	Possible. Data for disconnected slaves is also verified.
	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. An EtherCAT Slave Backup Failed event is recorded in the event log.
*2. An EtherCAT Slave Restore Operation Failed event is 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 Fail-soft 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).

9-7-3 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.

9-7-4 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-1SN□□□-ECT AC Servo Drives	*1
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	The parameters in the Inverter are not restored. Refer to *Procedure to Write Parameters for an 3G3AX-MX2-ECT or 3G3AX-RX-ECT Inverter* on page 9-30 to write the parameters from the Sysmac Studio to the Inverter. If you execute verification without writing the parameters, the verification result for the inverter will be Not matched in the EtherCAT slaves verification result file.
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 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.

EtherCAT slaves	Precautions
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 User's Manual* (Cat. No. Z332) for information on saving the settings and loading them to the Controller.
E3NW-ECTE3X-ECT Digital Sensors	The parameters in the Sensor are not backed up, restored, or verified.
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 to Write Parameters for an 3G3AX-MX2-ECT or 3G3AX-RX-ECT Inverter

The parameters in the Inverter are not restored.

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.

   The Inverter parameters are displayed 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.

9-8 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 Units User's Manual* (Cat. No. Z930) for information on importing and exporting settings for a Safety Control Unit.

9-8-1 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	Backing up data	Restoring data	Verifying data
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.

9-8-2 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		
	Backing up data	Restoring data	Verifying data
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.

9-8-3 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.

9-9 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.

9-9-1 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. These files are created when data is backed up.

● Restore Command File

This file specifies the data groups to transfer by restoring data from a Virtual 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 is not used in an NY-series Controller.

● Verification Results Files

The verification results files contain the results of comparing the Controller data and the data in a backup file on the Virtual SD Memory Card.

There are three 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.

9-9-2 Specifications of a Backup File

This section describes the file name, creation timing, and created directory for a backup file.

File Name

The backup file name of the NY-series Controller is given below.

File	File name
Backup file	NYBackup.dat

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	System-defined variables	When backup is executed	Directory on the Virtual 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 Virtual SD Memory Card that you specified with the Sysmac Studio
	Special instruction	When backup is executed	The directory on the Virtual 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
Controller backups for Industrial PC Support Utility	Industrial PC Support Utility Controller Backup Dialog Box	When backup is executed	Directory in the Industrial PC that you specified with the Industrial PC Support Utility
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

9-9-3 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	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
Controller backups for Industrial PC Support Utility	Industrial PC Support Utility 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 “yes”.
; --- IP Address of Built-in EtherNet/IP Port Settings. --- ; "yes":will be restored, "no":will not be restored IPAdr=yes	IP address of built-in EtherNet/IP port yes/no: Restore/Do not restore.
; --- 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.
:---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.

4 An entry of IP Address of Built-in EtherNet/IP Port Settings is not created if the backup is performed in the Controller with unit version 1.12. In the Controller with unit version 1.14 or later, if an entry of IP Address of Built-in EtherNet/IP Port Settings for which the restore command file is not created is used, the operation is performed as “IPAdr=yes”. Refer to *Compatibility between Restore Command Files* on page 9-37 for compatibility between the restore command file with unit version 1.12 and the restore command file with unit version 1.14 or later.

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.

Compatibility between Restore Command Files

The following table shows the compatibility between the restore command file with unit version 1.12 and the restore command file with unit version 1.14 or later.

Unit version of Controller that creates the restore command file and backup file	Unit version of Controller where data is being restored	
	Version 1.12	Version 1.14 or later
Version 1.12	Restorable. IP Address of Built-in EtherNet/IP Port Settings is restored.	Restorable. IP Address of Built-in EtherNet/IP Port Settings is restored.
Version 1.14 or later	Not restorable. In the Controller with unit version 1.12, because the entry of IP Address of Built-in EtherNet/IP Port Settings cannot be interpreted, a Restore Operation Failed to Start error in an observation level occurs. The error details will be “0104 hex: The contents of the restore command file are not correct”.	Restorable. IP Address of Built-in EtherNet/IP Port Settings corresponds to “yes/no” of “IPAdr”.

9-9-4 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

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 Virtual SD Memory Card 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. --- Result=Matched [UnitConfig] ; --- Unit/Slave Parameters. --- Result=Not matched	User program and settings Matched: No differences were found, Not matched: Differences were found. Units and slaves settings Matched: No differences were found, Not matched: Differences were found.

Note 1 The verification results are given only for the data groups that were compared.

2 The verification results of IP Address of Built-in EtherNet/IP Port Settings are including in an entry of user program and settings even for the Controller with unit version 1.14 or later.

9-9-5 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

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 Virtual SD Memory Card 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

*1. 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.

9-9-6 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

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

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 Virtual SD Memory Card 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

9-10 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 Controller where the data was backed up from does not match the model number where data is being restored.
- When the unit versions of the Controller, 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).

9-10-1 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)	Restoring with Sysmac Studio Controller backup functions (computer to Controller)	Restoring with Controller backup functions for Industrial PC Support Utility (Windows 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 Controller backup functions for Industrial PC Support Utility (Controller to Windows)	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

Operation authority verification

Data Trace Settings

Present values of variables

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 *9-6 Sysmac Studio Variable and Memory Backup Functions* for details on these functions.

9-10-2 Compatibility between NY-series Controller Models

The following table shows the file compatibility when the Controller model where the data was backed up from is different from the group where the data is being restored.

(C: Compatible, N: Not compatible.)

Controller model where data was backed up	Controller model to restore to
	NY5□2-1500 NY5□2-1400 NY5□2-1300
NY5□2-1500 NY5□2-1400 NY5□2-1300	C

Additional Information

NC Integrated Controller are not compatible.

Even if the Controller models are compatible, there may be restrictions between various Controller models.

The following table shows which restoration function can be used based on whether the Controller models are compatible.

(R: Restored, x: Not restored)

Compatibility between Controller models	Function used to restore data				
	Restoring with SD Memory Card backup functions (SD Memory Card to Controller)	Restoring with Sysmac Studio Controller backup functions (computer to Controller)*1	Restoring with Controller backup functions for Industrial PC Support Utility (Windows to Controller)	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	R	R	R
Not compatible	x*3	x	x	x	x

*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.

9-10-3 Compatibility between Unit Versions of NY-series Controllers

The following table shows the compatibility of backup files when the unit versions of the Controller 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 Controller	Function used to restore data			
	Restoring with SD Memory Card backup functions (SD Memory Card to Controller)	Restoring with Sysmac Studio Controller backup functions (computer to Controller)	Restoring with Controller backup functions for Industrial PC Support Utility (Windows to Controller)	Restoring with Sysmac Studio variable and memory backup functions (computer to Controller)
Unit version of Controller where data is being restored is newer.	R	R	R	R
Unit version of Controller where data is being restored is older.	x	x	x	R

9-11 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 Controller 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 Memory All Clear operation
- 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 NY-series Controller 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 NY-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

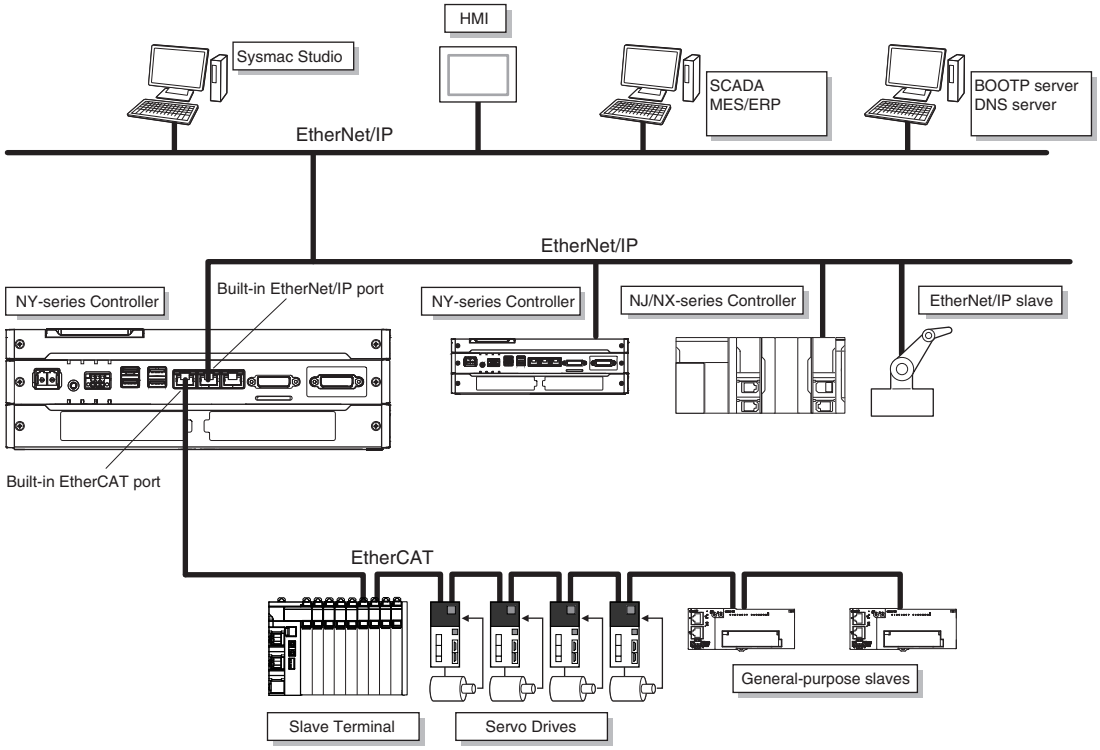
Step 5. Checking Operation and Starting Operation on the Actual System

Refer to *1-4 Overall Operating Procedure for the NY-series Controller* for details.

10-1-1 Introduction

● NY-series System

You can use the NY-series System to build the communications system shown below.



Connection		Connection method
Sysmac Studio connection		Use the built-in EtherNet/IP port.
Connections between Controllers	Connections with NJ/NX/NY-series Controller	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 systems	Connections to SCADA, MES/ERP	Use the built-in EtherNet/IP port.
Connections to servers	Connections to FTP servers, BOOTP servers, or DNS servers	Use the built-in EtherNet/IP port.

Additional Information

Refer to the *NY-series Industrial Panel PC / Industrial Box PC Setup User's Manual* (Cat. No. W568) for information on the communications system that can be used in Windows.

10-2 Connection with Sysmac Studio

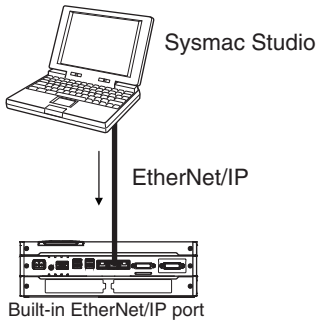
This section describes the configurations for connecting the Sysmac Studio to an NY-series Controller.

10-2-1 Configurations That Allow Online Connections

You can connect online from the Sysmac Studio to a built-in EtherNet/IP port of the NY-series Controller.

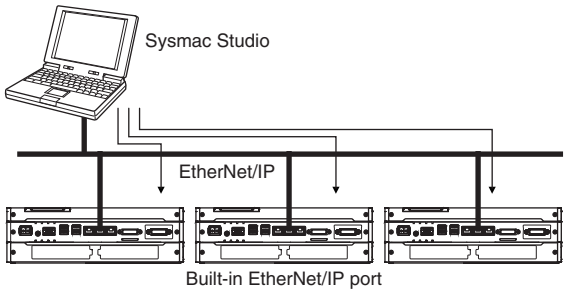
● Connecting with EtherNet/IP

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.

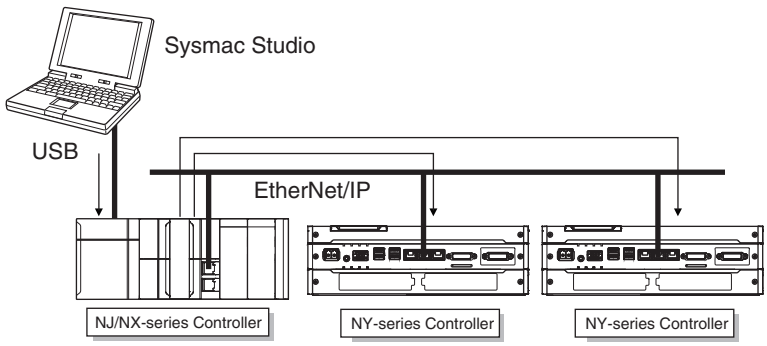
1:N Connections



- 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

You can connect to the NY-series built-in EtherNet/IP port through a USB port of the NJ/NX-series Controller.



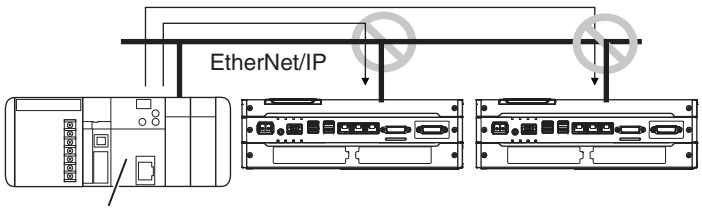
- Specify the IP address of the remote node from the Sysmac Studio.
- You can use either a cross cable or a straight cable.

Note Connect a computer and the NJ/NX-series 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

● Routing through CS/CJ-series EtherNet/IP Units/Ports

You cannot connect to an NY-series Controller 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).



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

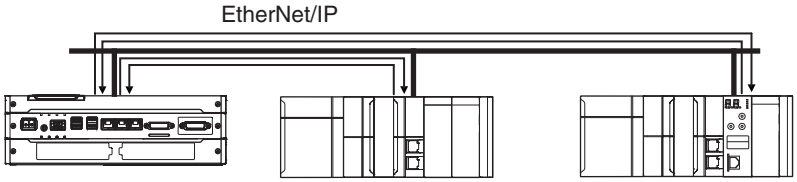
EtherNet/IP

You can use the built-in EtherNet/IP port.

For information on the built-in EtherNet/IP port, refer to the *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherNet/IP Port User's Manual* (Cat. No. W563).

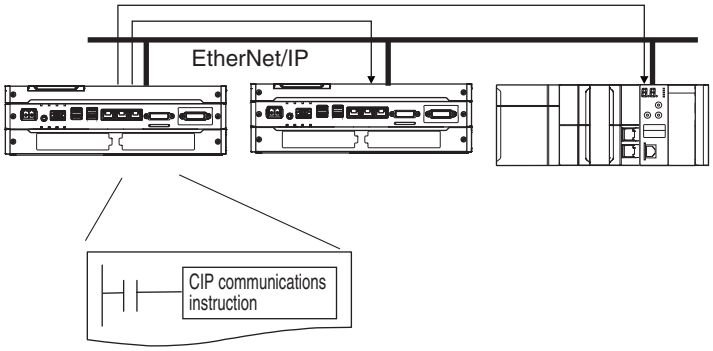
● Tag Data Links

You can create tag data links between NJ/NX/NY-series Controllers on an EtherNet/IP network.



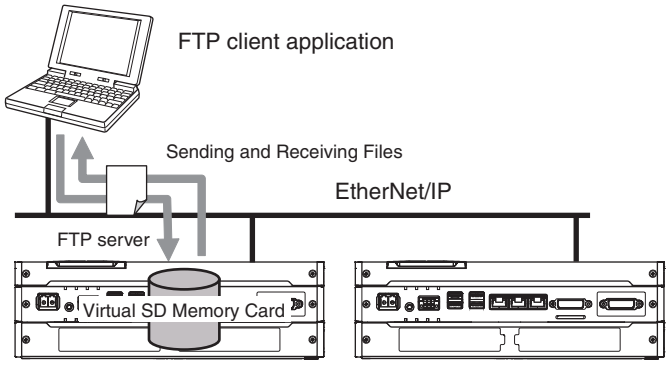
● Message Communications

You can send CIP messages from the user program.



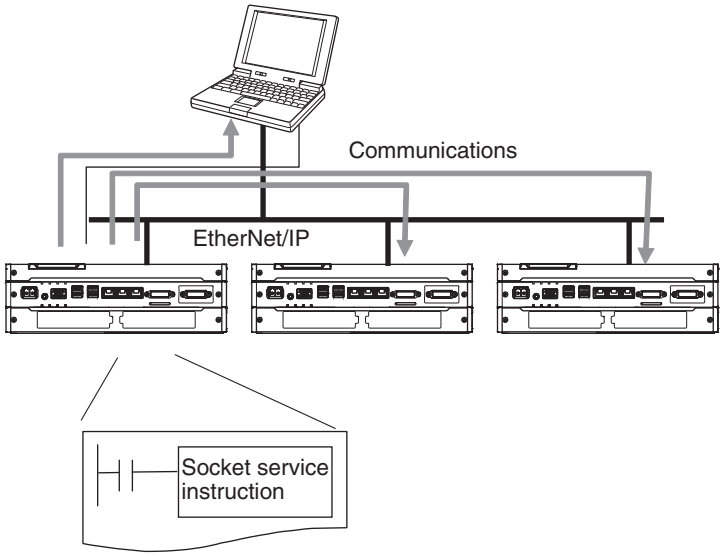
Sending and Receiving Files

You can send and receive files on the Virtual SD Memory Card in the NY-series Controller from an FTP client application.



Socket Services

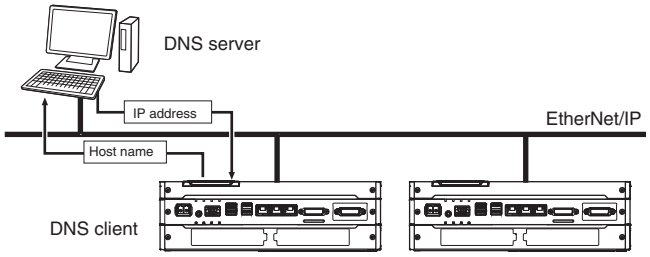
You can directly use TCP or UDP from the user program to send and receive any data with remote nodes between a host computer and the Controller, or between Controllers. The socket services are supported only for the built-in EtherNet/IP ports.



Specifying Host Names

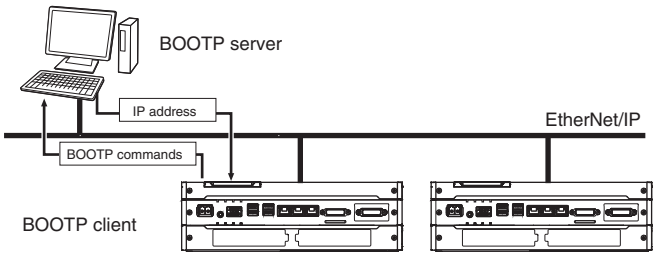
You can use the DNS client or set up your Hosts so that you can specify the IP address of the SNMP manager or the target destination of a socket instruction or CIP communications instruction with a host name instead of an IP address.

Example: Setting Host Names on the DNS Server



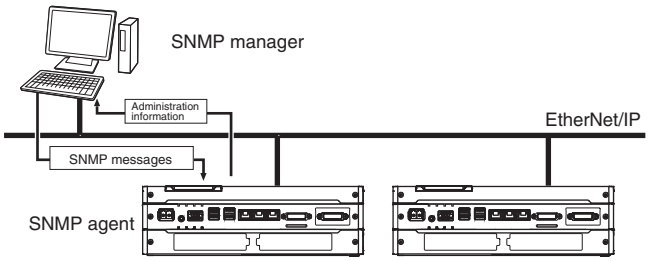
Obtaining an IP Address When the Power Is Turned ON

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.

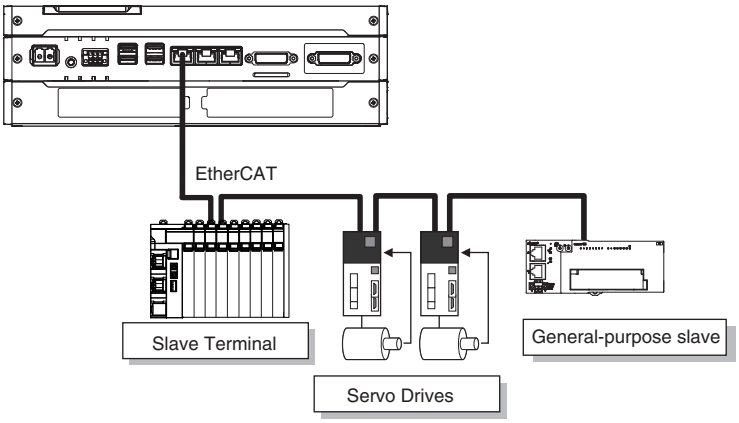


10-3-2 Connection Configuration between Controllers and Slaves

EtherCAT

High-speed, high-precision communications are possible with Servo Drives and general-purpose slaves.

Refer to the *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherCAT Port User's Manual* (Cat. No. W562) for details.

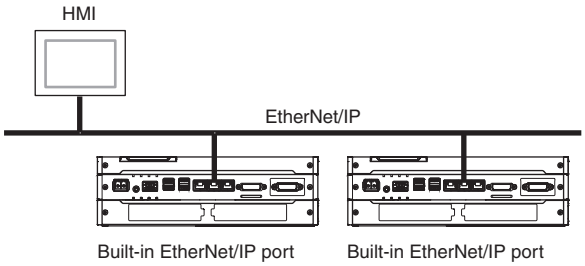


10-4 Connection with HMIs

This section shows the connection configurations used to connect HMIs to the NY-series Controller.

● EtherNet/IP

You can use a built-in EtherNet/IP port to connect to an HMI.



For information on the built-in EtherNet/IP port, refer to the *NY-series Industrial Panel PC / Industrial Box PC Built-in EtherNet/IP Port User's Manual* (Cat. No. W563).

11

Example of Actual Application Procedures

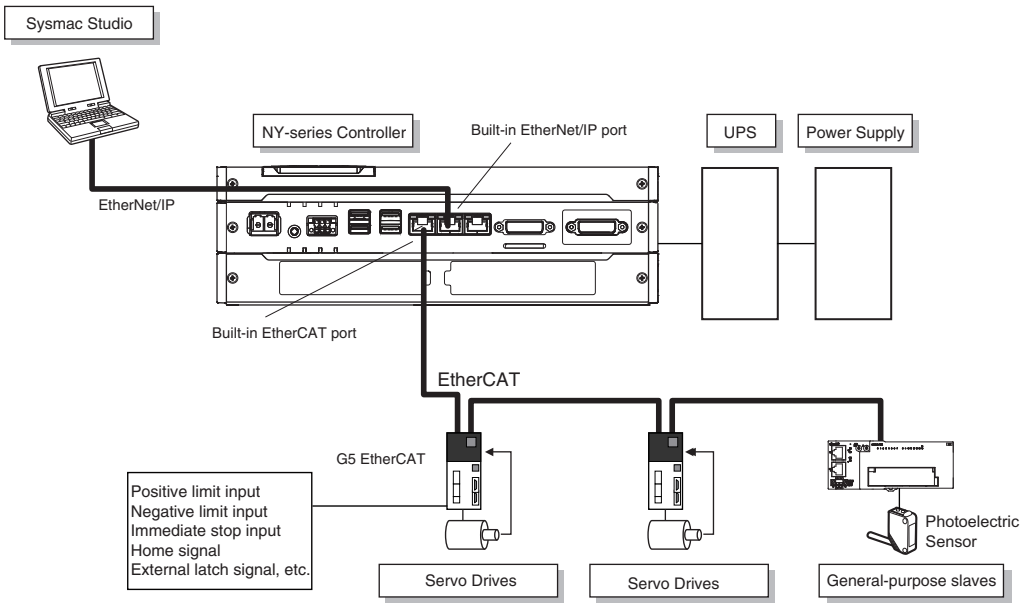
This section describes the procedures that are used to actually operate an NY-series Controller.

11-1 Example Application

This section describes an example application for an NY-series Controller.

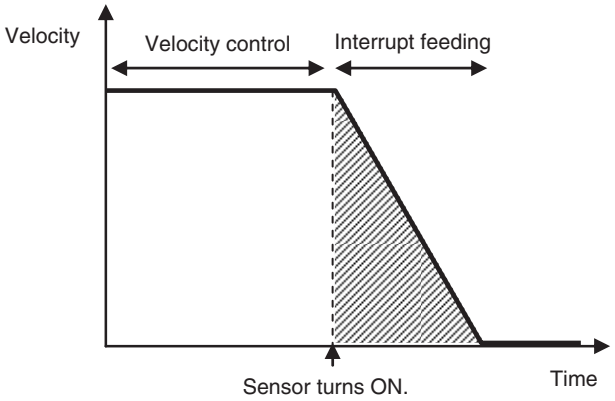
11-1-1 System Configuration

Unit name		Qty	Connected device
Power Supply		1	---
UPS (Uninterruptible Power Supply)		1	---
NY-series Industrial Box PC		1	---
EtherCAT slaves	Servo Drives (G5 EtherCAT)	2	---
	I/O Terminal	1	Photoelectric Sensor



11-1-2 Operation

Interrupt feeding starts when the sensor signal changes to ON during velocity control.



The vertical position changes based on the input from the Photoelectric Sensor.

11-2 Overview of the Example Procedure

This section describes examples of the actual operating procedures for an NY-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:

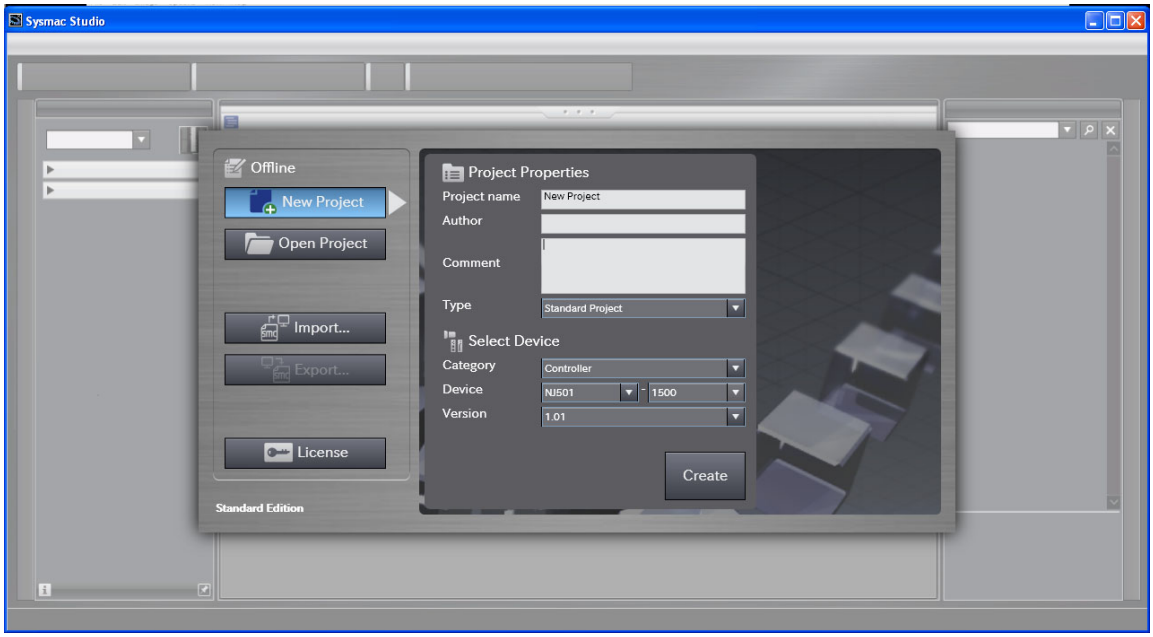
- The separation of variables into those that you use in more than one POU (global variables) and variables that you use in only specific POUs (local variables)
- Defining the variable names for the device variables that you use to access slaves
- Defining the attributes of variables, such as the Name 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 slave configurations, register global variables and device variables, create axes (axis variables), and set the Controller Setup.

Start the Sysmac Studio.

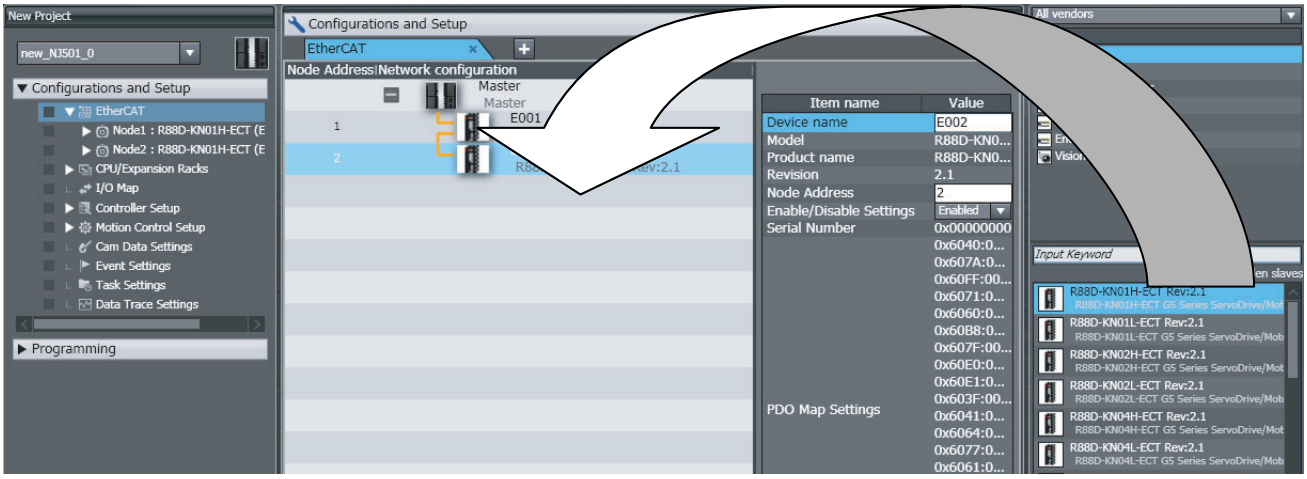
Create a project in Sysmac Studio.



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**.

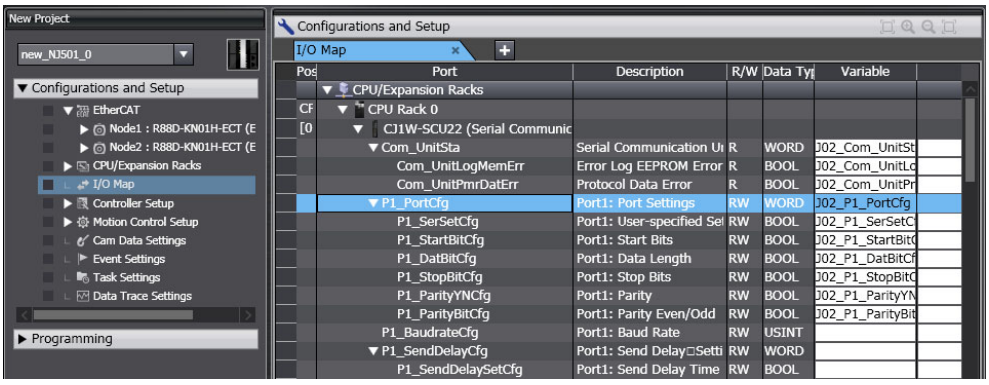
2 Register the global variables in the global variable table.

● 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 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.

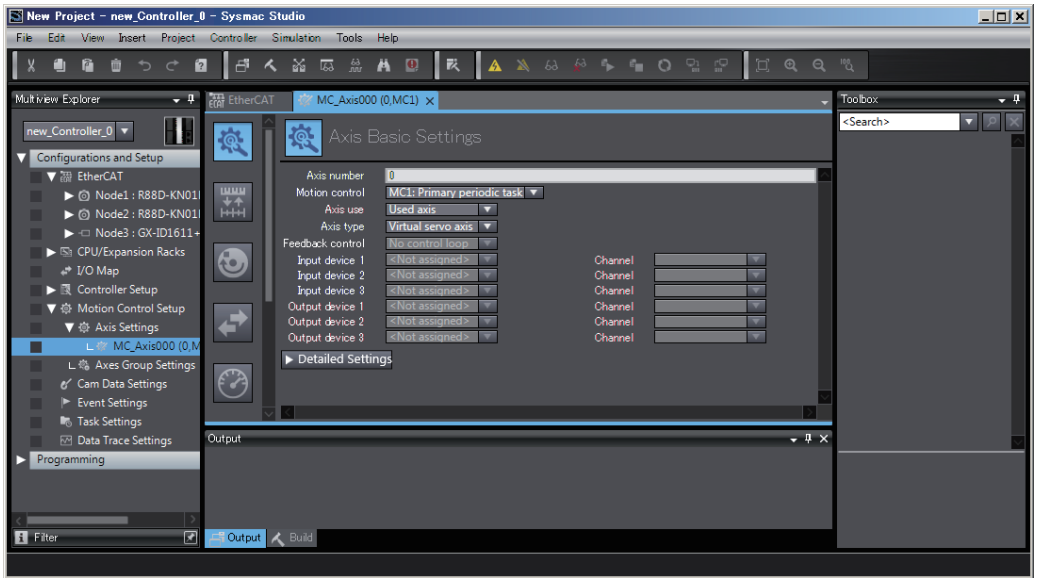


By default, device variables are registered in the global variable table. If necessary, you can change the variable type from a global variable to a local variable (internal variable) for a POU.

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**.
- 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.

● Initial Settings for the PLC Function Module:

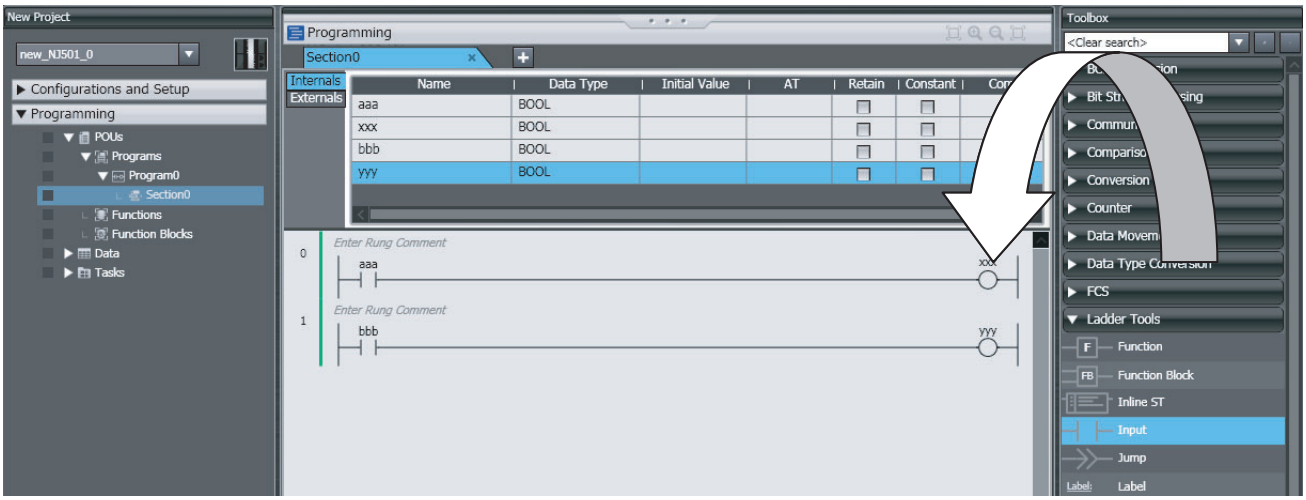
The Controller Setup includes the Startup Mode and other parameters.

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 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 NY-series Controller.
2. Connect the computer and the NY-series Controller with EtherNet/IP.
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.

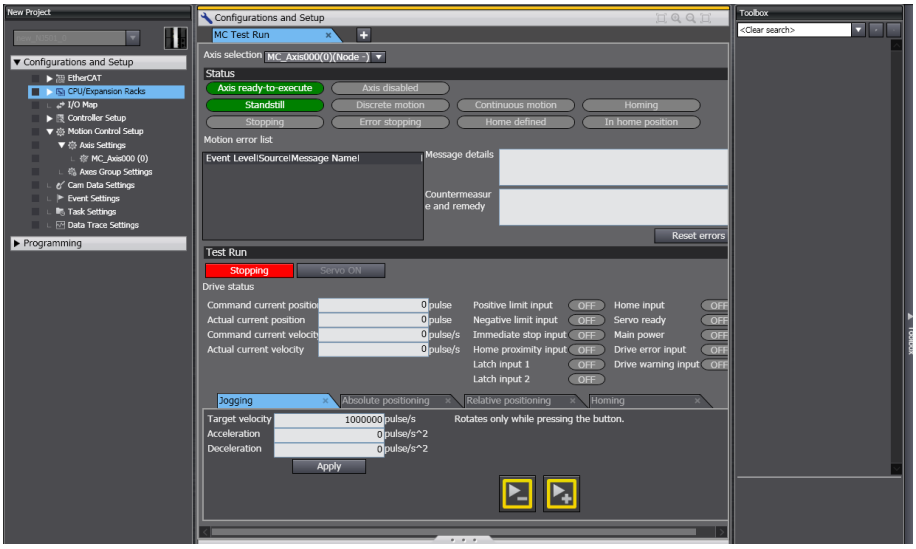
Note Use the Synchronize Menu of the Sysmac Studio to upload and download the project.

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 NY-series Controller 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 NY-series Controller 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.

A

Appendices

The appendices provide the NY-series Controller 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 functional and performance specifications of the control function of the NY-series Industrial Panel PC / Industrial Box PC with Machine Automation Control Software.

Refer to the *NY-series Industrial Panel PC Hardware User's Manual* (Cat. No. W557) or the *NY-series Industrial Box PC Hardware User's Manual* (Cat. No. W556) for the general specifications.

A-1-1 Performance Specifications

This section gives the performance specifications of the control function of the NY-series Industrial Panel PC / Industrial Box PC.

Item				NY5□2-		
				15□□	□4□□	13□□
Processing time	Instruction execution times	LD instruction		0.33 ns or more		
		Math instructions (for long real data)		1.2 ns or more		
Programming	Program capacity*1	Size		40 MB		
		Quantity	Number of POU definitions	3,000		
			Number of POU instances	24,000		
	Memory capacity for variables	Retain attributes	Size	4 MB		
			Number of variables	40,000		
		No Retain attributes	Size	64 MB		
			Number of variables	180,000		
	Data types	Number of data types		4,000		
Motion control	Number of controlled axes	Maximum number of controlled axes*2		64 axes	32 axes	16 axes
		Maximum number of used real axes*3		64 axes	32 axes	16 axes
		Maximum number of axes for single-axis control		64 axes	32 axes	16 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			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		
		Maximum number of cam tables		640 tables		
	Position units			Pulse, mm, μm, nm, degree, and inch		
	Override factors			0.00%, or 0.01% to 500.00%		

Item				NY5□2-		
				15□□	□4□□	13□□
Built-in EtherNet/IP port	Number of ports			1		
	Physical layer			10BASE-T, 100BASE-TX, or 1000BASE-T		
	Frame length			1,514 bytes max.		
	Media access method			CSMA/CD		
	Modulation			Baseband		
	Topology			Star		
	Baud rate			1 Gbps (1000BASE-T)		
	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 limitations when an Ethernet switch is used.		
	CIP service: Tag data links (cyclic communications)	Maximum number of connections		128		
		Packet interval[*4]		Can be set for each connection. 1 to 10,000 ms in 1-ms increments		
		Permissible communications band		20,000 pps[*5] (including heartbeat)		
		Maximum number of tage sets		128		
		Tag types		Network variables		
		Number of tags per connection (i.e., per tag set)		8 (7 tags if Controller status is included in the tag set.)		
		Maximum number of tags		256		
		Maximum link data size per node (total size for all tags)		184,832 bytes		
		Maximum data size per connection		1,444 bytes		
		Maximum number of registrable tag sets		128 (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.)		
		Multi-cast packet filter[*6]		Supported		
	CIP message service : Explicit messages	Class 3 (number of connections)		64 (clients plus server)		
		UCMM (non-connection type)	Maximum number of clients that can communicate at one time	32		
			Maximum number of servers that can communicate at one time	32		
	Number of TCP sockets			30		

Item			NY5□2-		
			15□□	□4□□	13□□
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		128		
	Range of node addresses that can be set		1 to 512		
	Maximum process data size		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		500, 1,000, 2,000, 4,000, or 8,000 μs		
	Sync jitter		1 μs max.		
Unit configuration	Maximum number of connectable Units	Maximum number of NX Units for entire controller	4,096 (On EtherCAT Slave Terminals)		
	Maximum number of Expansion Racks		0		
Internal clock			At ambient temperature of 55°C: −3.5 to +0.5 minute error per month At ambient temperature of 25°C: −1.5 to +1.5 minute error per month At ambient temperature of 0°C: −3 to +1 minute error per month		

*1 This is the capacity for the execution objects and variable tables (including variable names).
*2 This is the total for all axis types.
*3 This is the total number of axes that are set as servo axes or encoder axes and are also set as used axes.
*4 Data will be refreshed at the set interval, regardless of the number of nodes.
*5 “pps” means packets per second, i.e., the number of communications packets that can be sent or received in one second.
*6 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.

A-1-2 Function Specifications

This section gives the functional specifications of the control function of the NY-series Industrial Panel PC / Industrial Box PC.

Item				NY5□2-□□□
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	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		---
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
		Structures	Function	A derivative data type that groups together data with different data types.
			Maximum number of members	2,048
			Nesting maximum levels	8
			Member data types	Basic data types, structures, unions, enumerations, or array variables
			Specifying member offsets	You can use member offsets to place structure members at any memory locations.
		Union	Function	A derivative data type that enables access to the same data with different data types.
			Maximum number of members	4
			Member data types	BOOL, BYTE, WORD, DWORD, and LWORD
		Enumeration	Function	A derivative data type that uses text strings called enumerators to express variable values.

Item				NY5□2-□□□
Programming	Data type attributes	Array specifications	Function	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.
			Maximum number of dimensions	3
			Maximum number of elements	65,535
			Array specifications for FB instances	Supported
		Range specifications		You can specify a range for a data type in advance. The data type can take only values that are in the specified range.
	Libraries			You can use user libraries.
Motion control	Control modes			Position control, Velocity control, and Torque control
	Axis types			Servo axes, Virtual servo axes, Encoder axes, and Virtual encoder axes
	Positions that can be managed			Command positions and actual positions
	Single axes	Single-axis position control	Absolute positioning	Positioning is performed for a target position that is specified with an absolute value.
			Relative positioning	Positioning is performed for a specified travel distance from the command current position.
			Interrupt feeding	Positioning is performed for a specified travel distance from the position where an interrupt input was received from an external input.
			Cyclic synchronous absolute positioning	A positioning command is output each control period in Position Control Mode.
		Single-axis velocity control	Velocity control	Velocity control is performed in Position Control Mode.
			Cyclic synchronous velocity control	A velocity command is output each control period in Velocity Control Mode.
		Single-axis torque control	Torque control	The torque of the motor is controlled.
		Single-axis synchronized 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 output as the command position.
		Single-axis manual operation	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.

Item				NY5□2-□□□
Motion control	Single axes	Auxiliary functions 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 override factors	The target velocity of an axis can be changed.
			Changing the current position	The command current position or actual current position of an axis can be changed to any position.
			Enabling external latches	The position of an axis is recorded when a trigger occurs.
			Disabling external latches	The current latch is disabled.
			Zone monitoring	You can monitor the command position or actual position of an axis to see when it is within a specified range (zone).
			Enabling digital 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 compensation	The function which compensate the position for the axis in operation.
			Start velocity	You can set the initial velocity when axis motion starts.
	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.

Item				NY5□2-□□□
Motion control	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.
		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 1S-series Servomotor or 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		128
Communications	EtherNet/IP port	Communications protocol		TCP/IP and UDP/IP
		TCP/IP function	CIDR	Classless inter-domain routing is the function used to allocate IP addresses that do not use classes (Class A to C).
			IP Forwarding	This function is used to transfer IP packets over multiple interfaces.
			Packet Filter*2	This function is used to determine passing or blocking IP packets based on source IP address and TCP port number.
			NAT	This function is used to convert two IP addresses and transfer.

Item				NY5□2-□□□
Communications	EtherNet/IP port	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.
			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.
	EtherCAT port	Network scanning		Information is read from connected slave devices and the slave configuration is automatically generated.
		DC (distributed clock)		Time is synchronized by sharing the EtherCAT system time among all EtherCAT devices (including the master).
		Packet monitoring		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.
		Enable/disable settings for slaves		The slaves can be enabled or disabled as communications targets.
		Disconnecting/reconnecting slaves		Temporarily disconnects a slave from the EtherCAT network for maintenance, such as for replacement of the slave, and then connects the slave again.
		Support application protocol	CoE	SDO messages of the CAN application can be sent to slaves via EtherCAT
	Communications instructions			FTP client instructions, CIP communications instructions, socket communications instructions, SDO message instructions, and Modbus RTU protocol instructions
System management	Event logs	Function		Events are recorded in the logs
		Maximum number of events	System event log	2,048
			Access event log	1,024
			User-defined event log	1,024
Debugging	Online editing			Programs, function blocks, functions, and global variables can be changed online. More than one operators can change POUs individually via network.
	Forced refreshing			The user can force specific variables to TRUE or FALSE.
		Maximum number of forced variables	Device variables for EtherCAT slaves	64
	MC Test Run			Motor operation and wiring can be checked from the Sysmac Studio.
	Synchronizing			The project file in the Sysmac Studio and the data in the CPU Unit can be made the same when online.
	Differential monitoring			You can monitor when a variable changes to TRUE or changes to FALSE.
		Maximum number of monitored variables		8

Item				NY5□2-□□□
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.
			Continuous trace	Data tracing is executed continuously and the trace data is collected by the Sysmac Studio.
		Maximum number of simultaneous data traces		4
		Maximum number of records		10,000
		Maximum number of sampled variables		192 variables
		Timing of sampling		Sampling is performed for the specified task period, at the specified time, or when a sampling instruction is executed.
		Triggered traces		Trigger conditions are set to record data before and after an event.
			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 (≠)
			Delay	Trigger position setting: A slider is used to set the percentage of sampling before and after the trigger condition is met.
	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
			Maximum number of message languages	9 (Sysmac Studio) 2 (NS-series PT)
		User-defined errors	Function	User-defined errors are registered in advance and then records are created by executing instructions.
			Levels	8
			Maximum number of message languages	9
Security	Protecting software assets and preventing operating mistakes	CPU Unit names and serial IDs		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.
		Protection	User program transfer with no restoration information	You can prevent reading data in the CPU Unit from the Sysmac Studio.
			CPU Unit write-protection	You can prevent writing data to the CPU Unit from the Sysmac Studio or SD Memory Card.
			Overall Project file protection	You can use passwords to protect .smc files from unauthorized opening on the Sysmac Studio.
			Data protection	You can use passwords to protect POUs on the Sysmac Studio.
		Verification of operation authority		Online operations can be restricted by operation rights to prevent damage to equipment or injuries that may be caused by operating mistakes.
			Number of groups	5
		Verification of user program execution ID		The user program cannot be executed without entering a user program execution ID from the Sysmac Studio for the specific hardware (CPU Unit).
SD Memory Card functions	Save location			Shared folder[*3]
	Application	SD Memory Card operation instructions		You can access Virtual SD Memory Cards from instructions in the user program.
		File operations from the Sysmac Studio		You can perform file operations for Controller files in the Virtual SD Memory Card and read/write general-purpose document files on the computer.
		File operations by the FTP client and sever functions		You can save and read files by the FTP client and server functions.

Item				NY5□2-□□□
Backing up data	SD Memory Card backups	Operating methods	Specification with system-defined variables	Backup and verification operations are performed by manipulating system-defined variables.
			SD Memory Card Window in Sysmac Studio	Backup and verification operations are performed from the SD Memory Card Window of the Sysmac Studio.
			Special instruction	The special instruction is used to backup data.
		Protection	Disabling backups to SD Memory Cards	Backing up data to a Virtual SD Memory Card is prohibited.
	Sysmac Studio Controller backups			The Sysmac Studio is used to backup, restore, or verify Controller data.

*1 Inline ST is supported. (Inline ST is ST that is written as an element in a ladder diagram.)
*2 The function is available only for the internal port.
*3 The folder exists in HDD or SSD on which the Windows operates. The folder on the Controller is used as the Virtual SD Memory Card.

A-2 Calculating Guidelines for the Real Processing Times of Tasks for the NY-series System

This section describes how to calculate guidelines for the average real processing times of tasks on paper for the NY-series System.

You must use the physical Controller to check the real processing times of tasks and task execution times. For details, refer to *5-9 Task Design Methods and I/O Response Times*.

Precautions for Correct 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, the load on Windows, the connection of Industrial Monitors, the usage of PCIe cards, 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 is 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 5-58.

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-16 periodic task	Priority-17 and priority-18 periodic tasks
I/O refresh processing		I/O is refreshed for EtherCAT slaves.	Performed.	Not 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 output processing	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 = EtherCAT slave processing time

The following describes how to determine the EtherCAT slave processing time.

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] = 0.0008 × pDout + 0.0028 × pDin + 0.082 × pDinout
+ (1.24 × Snum + 0.01 × 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]

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-16 and *Instruction Configuration for Standard ST Instructions* on page A-18 for information on the instruction configuration.

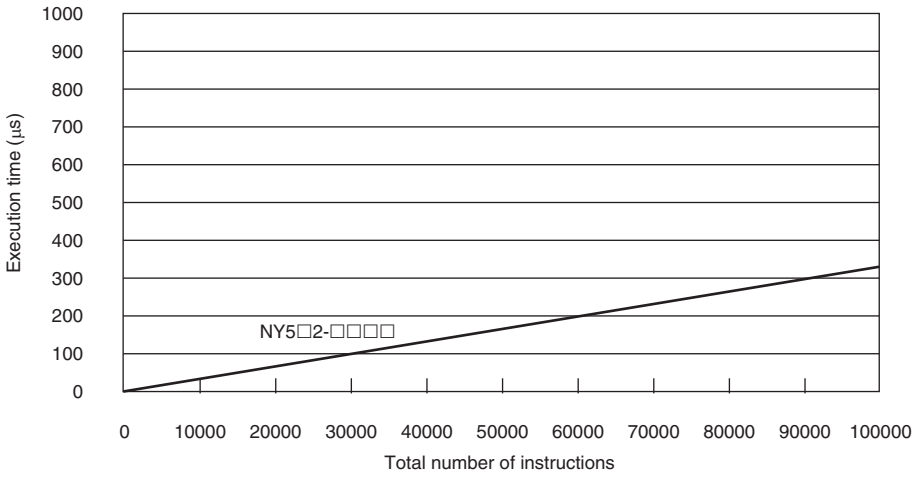
● Simple Estimate

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



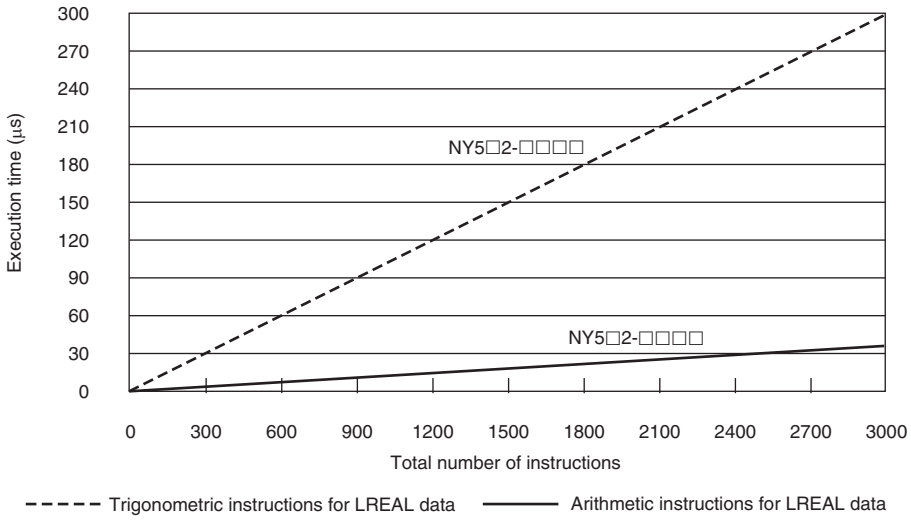
- Instruction Configuration for Standard Ladder Diagram Instructions

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



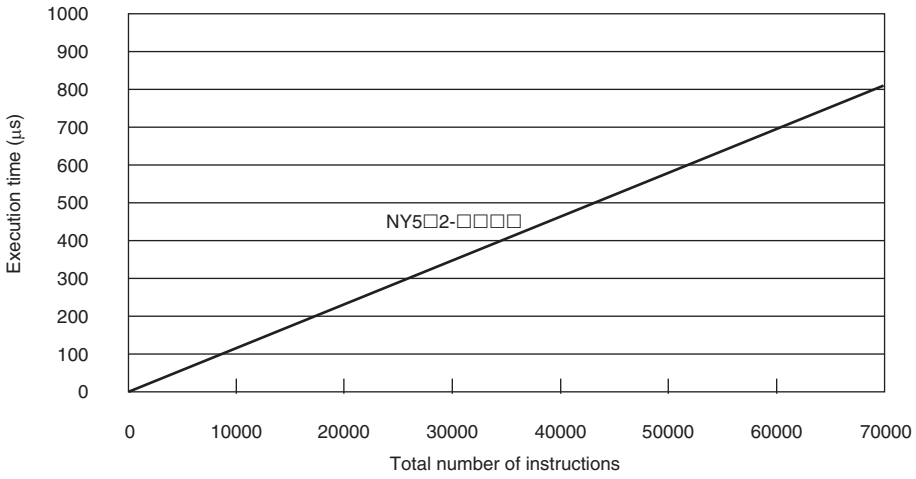
- Configuration of Arithmetic Instructions for LREAL Data

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%

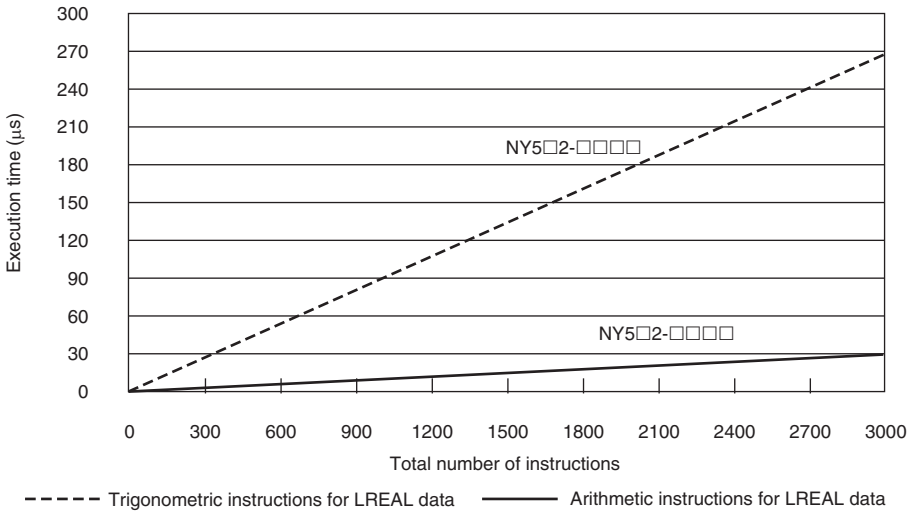
Execution Time for Standard ST Instructions



- 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%

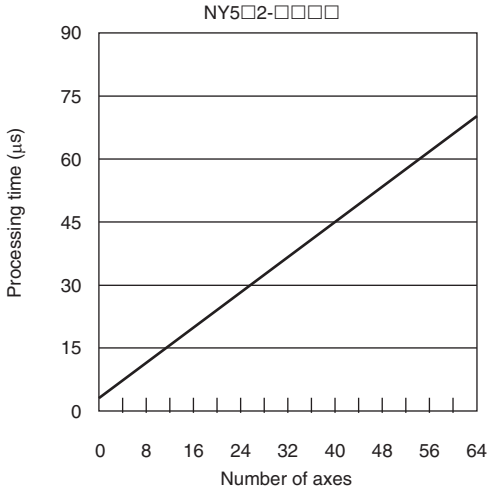
- Configuration of Trigonometric Instructions for LREAL Data

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)
	NY5□2-□□□□
Primary periodic task	258
Priority-16, 17, or 18 periodic task	20

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 NY5□2-□□□□ Controller with a unit version of 1.12, 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-1SN□□□-ECT Servo Drives: 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-1SN□□□-ECT Servo Drives, 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-1SN□□□-ECT	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

= $0.0008 \times \text{pDout} + 0.0028 \times \text{pDin} + 0.082 \times \text{pDinout} + (1.24 \times \text{Snum} + 0.01 \times \text{Clen})$

= $0.0008 \times 156 + 0.0028 \times 196 + 0.082 \times 197 + (1.24 \times 7 + 0.01 \times 10)$

= $0.1248 + 0.5488 + 16.154 + (8.68 + 0.1)$

$\approx$ 26 [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 NY5□2-□□□□ is 16.5 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 NY5□2-□□□□ is 2.5 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 NY5□2-□□□□ is 10 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 = 16.5 + 2.5 + 10

= 29 [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 NY5□2-□□□□ is read as 7 μs.

● Common Processing Time

Because the task is the primary periodic task, the common processing time for the NY5□2-□□□□ is 258 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
= 26 + 29 + 7 + 258
= 320 [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 320 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, the load on Windows, the connection of Industrial Monitors, the usage of PCIe cards, 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 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 NY-series Controller	None
PLC Function Module	_PLC
Motion Control Function Module	_MC
EtherCAT Master Function Module	_EC
EtherNet/IP Function Module	_EIP, _EIP1, and _EIPIn1

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 NY-series Controller 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-4 Specifications for Individual System-defined Variables* for details on the specifications for individual system-defined variables.

A-3-1 System-defined Variables for the Overall NY-series Controller (No Category)

● Functional Classification: Clock

Variable name	Meaning	Function	Data type	Range of values	Reference
_CurrentTime	System Time	Contains the Controller's internal clock data.	DATE_ AND_ TIME	DT#2000-01-01-00:00:00 to DT#2099-12-31-23:59:59	page A-53

● 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-53
*_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-53
*_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-54
*_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-54
*_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-54
*_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-54

Variable name	Meaning	Function	Data type	Range of values	Reference
*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-55

● 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-55
_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-55

● Functional Classification: SD Memory Card

Variable name	Meaning	Function	Data type	Range of values	Reference
_Card1Ready	SD Memory Card Ready Flag	TRUE when the Virtual SD Memory Card is recognized in an NY-series Controller. FALSE when the Virtual SD Memory Card is not recognized. TRUE: The Card can be used. FALSE: The Card cannot be used.	BOOL	TRUE or FALSE	page A-56
_Card1Protect	SD Memory Card Write Protected Flag	The NY-series Controller does not use this variable. The value is always FALSE.	BOOL	TRUE or FALSE	page A-56
_Card1Err	SD Memory Card Error Flag	The NY-series Controller does not use this variable. The value is always FALSE.	BOOL	TRUE or FALSE	page A-56
_Card1Access	SD Memory Card Access Flag	The NY-series Controller does not use this variable. The value is always FALSE.	BOOL	TRUE or FALSE	page A-56
_Card1Deteriorated	SD Memory Card Life Warning Flag	The NY-series Controller does not use this variable. The value is always FALSE.	BOOL	TRUE or FALSE	page A-56
_Card1PowerFail	SD Memory Card Power Interruption Flag	TRUE when the power supply to the Controller was interrupted during access to the Virtual SD Memory Card. TRUE: Power was interrupted during Virtual SD Memory Card access. FALSE: Normal	BOOL	TRUE or FALSE	page A-57

Variable name / Member name	Meaning	Function	Data type	Range of values	Reference
_Card1BkupCmd	SD Memory Card Backup Commands		_sBKUP_CMD		page A-57
ExecBkup	Execute Backup Flag	Change this variable to TRUE to back up Controller data to a Virtual 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-57
CancelBkup	Cancel Backup Flag	Change this variable to TRUE to cancel backing up data to a Virtual 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-57
ExecVefy	Execute Verify Flag	Change this variable to TRUE to compare the Controller data to a backup file in the Virtual 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-57
CancelVefy	Cancel Verify Flag	Change this variable to TRUE to cancel comparing the Controller data to a backup file in the Virtual 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-58
DirName	Directory Name	Used to specify the directory name in the Virtual 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-58

Variable name / Member name	Meaning	Function	Data type	Range of values	Reference
_Card1BkupSta	SD Memory Card Backup Status		_sBKUP_STA		page A-58
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-58
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-58
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-59
_Card1VefySta	SD Memory Card Verify Status		_sVEFY_STA		page A-59
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-59
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-59
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-59
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-59

● Functional Classification: Backup

Variable name	Meaning	Function	Data type	Range of values	Reference
_BackupBusy	Backup Function Busy Flag	TRUE when a backup, restoration, or verification is in progress.	BOOL	TRUE or FALSE	page A-60

● 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 Controller 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-60
_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-60
_RetainFail	Retention Failure Flag	TRUE at the following time (failure of retention during power interruptions). • When an error is detected by checking the battery-backup memory or non-volatile memory at power ON. FALSE at the following times (no failure of retention during power interruptions). • When no error is detected by checking the battery-backup memory or non-volatile memory at power ON. • 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-60
_SelfTest_UPSSignal	UPS Signal Detection Flag	TRUE when a temporary power interruption signal from UPS is detected.	BOOL	TRUE or FALSE	page A-61
_RequestShutdown	Request Shutdown Flag	TRUE when the power supply button is pressed while running.	BOOL	TRUE or FALSE	page A-61

Functional Classification: OS (Windows)

Variable name	Meaning	Function	Data type	Range of values	Reference
_OSRunning	OS Running Flag	TRUE when the Controller observes that OS (Windows) is running.	BOOL	TRUE or FALSE	page A-61
_OSHalted	OS Halted Flag	TRUE when the Controller observes that OS (Windows) is stopped.	BOOL	TRUE or FALSE	page A-61
_OSErrorState	OS Error State Flag	TRUE when the Controller determines that an error occurred in OS (Windows).	BOOL	TRUE or FALSE	page A-61

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-62
P_Off	Always FALSE Flag	This flag is always FALSE.	BOOL	FALSE	page A-62
P_CY	Carry Flag	This flag is updated by some instructions.	BOOL	TRUE or FALSE	page A-62
P_First_RunMode	First RUN Period Flag	This flag is TRUE for only one task period after the operating mode of the Controller 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 Controller begins operation. **Note** You cannot use this system-defined variable inside functions.	BOOL	TRUE or FALSE	page A-62
P_First_Run	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-62
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-63

● Functional Classification: Version

Variable name	Meaning	Function	Data type	Range of values	Reference
_UnitVersion	Unit Version	Contains the unit version of the Controller. 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-63
_HardwareRevision	Hardware Revision	Contains the hardware revision of the Controller. Contains - if the hardware revision is in blank, and A to Z for other cases.	STRING[2]	- or A to Z	page A-63

● Functional Classification: Self-diagnosis

Variable name	Meaning	Function	Data type	Range of values	Reference
_SelfTest_HighTemperature	CPU Unit High Temperature Flag	TRUE when the internal temperature of the Controller is too high.	BOOL	TRUE or FALSE	page A-64
_SelfTest_LowBattery	Low Battery Flag	TRUE when the battery is disconnected or the battery voltage is dropped.	BOOL	TRUE or FALSE	page A-64
_SelfTest_LowFanRevolution	Low FAN Revolution Flag	TRUE when the fan is disconnected or the rotation speed of a fan is decreased.	BOOL	TRUE or FALSE	page A-64

● Functional Classification: PLC Built-in

Variable name	Meaning	Function	Data type	Range of values	Reference
_DeviceOutHoldCfg (Ver.1.14)	Device Output Hold Configuration	It is 16#A5A5 if you retain the target device output when the operating mode is changed or when downloaded. In the case other than 16#A5A5, the target device output is initialized when the operating mode is changed or when downloaded.	BOOL	16#0000 to 16#FFFF	page A-64
_DeviceOutHoldStatus (Ver.1.14)	Device Output Hold Status	It is TRUE if the target device output is retained when the operating mode is changed or when downloaded. When the device output hold configuration is other than 16#A5A5, or when a major fault level Controller error occurs, the target device output is initialized and changes to FALSE.	BOOL	TRUE or FALSE	page A-64

A-3-2 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-65
.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-65
.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-65
.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-65
.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-66

● 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-66

A-3-3 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-67
_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-67
_MC_AX_ErrSta	Axis Error Status	Shows the error status for each axis. The status of up to 64 axes 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	16#0000 to 16#00F0	page A-67
_MC_GRP_ErrSta	Axes Group Error Status	Shows the error status for each axes group. The error status for up to 32 axes groups 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..31] OF WORD	16#0000 to 16#00F0	page A-67
_MC_COM	Common Variable	Shows the status that is common to the Motion Control Function Module. Refer to the *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561) for details on structure members.	_sCOMMON_REF	---	page A-68
_MC_GRP	Axes Group Variables	NY-series Controller: 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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561) for details on structure members.	ARRAY[0..31] OF _sGROUP_REF	---	page A-68

Variable name	Meaning	Function	Data type	Range of values	Reference
_MC_AX	Axis Variables	NY-series Controller: 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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561) for details on structure members.	ARRAY[0..63] OF _sAXIS_REF	---	page A-68

A-3-4 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#40F0	page A-69
_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-69
_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-69
_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-69
_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). Refer to information on the meanings of the error status bits at the end of this appendix for details.	Array [1..512] OF WORD	16#0000 to 16#00F0	page A-70
_EC_MacAdrErr	MAC Address Error	TRUE if there is an illegal MAC address.	BOOL	TRUE or FALSE	page A-70
_EC_LanHwErr	Communications Controller Error	TRUE if there is a communications controller hardware error.	BOOL	TRUE or FALSE	page A-70
_EC_LinkOffErr	Link OFF Error	TRUE if the communications controller link is not established.	BOOL	TRUE or FALSE	page A-70
_EC_NetCfgErr	Network Configuration Information Error	TRUE if there is illegal network configuration information.	BOOL	TRUE or FALSE	page A-70
_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-71
_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-71
_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-71
_EC_PDTimeoutErr	Process Data Reception Timeout Error	TRUE if a timeout occurs while receiving process data.	BOOL	TRUE or FALSE	page A-71

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-71
_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-71
_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-72
_EC_SlavAppErr	Slave Application Error	TRUE if there is an error in the slave's application status register.	BOOL	TRUE or FALSE	page A-72
_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-72
_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-72
_EC_IndataInvalidErr (Ver.1.14)	Input Process Data Invalid Error	TRUE if the Input Data Invalid state continued for the following period because the EtherCAT master could not perform process data communications normally when it was in the Operational state. • When the task period is 10 ms or shorter: 100 ms • When the task period is longer than 10 ms: 10 periods o the task	BOOL	TRUE or FALSE	page A-72
_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	TRUE or FALSE	page A-72
_EC_CycleExceeded	EtherCAT Communications Cycle Exceeded	TRUE if the Controller cannot establish communications within the set communications period at startup.	BOOL	TRUE or FALSE	page A-74

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	Communications 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 Configuration Information Error	Minor fault level
				_EC_NetCfgCmpErr	Network Configuration Verification Error	
				_EC_NetTopologyErr	Network Configuration Error	
				_EC_PDCommErr	Process Data Communications Error	
				_EC_PDTimeoutErr	Process Data Reception Timeout Error	
				_EC_PDSendErr	Process Data Transmission 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 Communications 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 troubleshooting 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	TRUE or FALSE	page A-74
_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	TRUE or FALSE	page A-74
_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	TRUE or FALSE	page A-74
_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	TRUE or FALSE	page A-75
_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	TRUE or FALSE	page A-75
_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	TRUE or FALSE	page A-75
_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-75
_EC_PktMonStop	Packet Monitoring Stopped	TRUE when packet monitoring is stopped.	BOOL	TRUE or FALSE	page A-76
_EC_LinkStatus	Link Status	TRUE if the communications controller link status is Link ON.	BOOL	TRUE or FALSE	page A-76
_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-76
_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-76

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.	BOOL	TRUE or FALSE	page A-76

Note All system-defined variables that are related to the status of EtherCAT communications give the current status.

● Functional Classification: EtherCAT Communications Diagnosis/Statistics Log

Variable name	Meaning	Function	Data type	Range of values	Reference
_EC_StatisticsLogEnable	Diagnosis/Statistics Log Enable	Changes to TRUE when the diagnosis/statistics log is started. Changes to FALSE when the diagnosis/statistics log is ended.	BOOL	TRUE or FALSE	page A-77
_EC_StatisticsLogCycleSec	Diagnosis/Statistics Log Cycle	Specifies the interval to write the diagnostic and statistical information of the diagnosis/statistics log in units of seconds. When 0 is specified, the diagnostic and statistical information is written only once when the diagnosis/statistics log is ended. **Note** The write interval does not change even if you change the value of this system-defined variable while the diagnosis/statistics log operation is in progress.	UINT	0, or 30 to 1800	page A-77
_EC_StatisticsLogBusy	Diagnosis/Statistics Log Busy	TRUE while the diagnosis/statistics log operation is in progress.	BOOL	TRUE or FALSE	page A-77
_EC_StatisticsLogErr	Diagnosis/Statistics Log Error	TRUE when the diagnosis/statistics log failed to start or it is impossible to write into the log. The value of this flag is determined when *_EC_StatisticsLogBusy* (Diagnosis/Statistics Log Busy) changes to FALSE after the diagnosis/statistics log operation is started. The error end is caused by the following. • Another records cannot be added in the log file because the capacity of the Virtual SD Memory Card is fully used. • There is no Virtual SD Memory Card. • The function cannot be started because the value specified for *_EC_StatisticsLogCycleSec* (Diagnosis/Statistics Log Cycle) is invalid.	BOOL	TRUE or FALSE	page A-77

A-3-5 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. NY-series Controllers: Represents the collective status of the following error flags. • *_EIP1_PortErr* (Communications Port1 Error) • *_EIPIn1_PortErr* (Internal Port1 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-78
_EIP_PortErr	Communications Port Error	This is the error status variable for the communications port. NY-series Controllers: 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.	WORD	16#0000 to 16#00F0	page A-78

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.	WORD	16#0000 to 16#00F0	page A-78
_EIPIn1_PortErr	Internal Port1 Error	This is the error status variable for the internal port 1. It represents the collective status of the following error flags. • *_EIPIn1_IPAdrCfgErr* (Internal Port1 IP Address Setting Error) • *_EIP1_IPAdrDupErr* (Internal Port1 IP Address Duplication 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.	WORD	16#0000 to 16#00F0	page A-79

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_CipErr	CIP Communications Error	This is the error status variable for CIP communications. NY-series Controller: 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-79
_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 Application 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-80
_EIP_MacAdrErr	MAC Address Error	NY-series Controller: Indicates that an error occurred when the MAC address was read on the communications port 1 at startup. TRUE: Error FALSE: Normal	BOOL	TRUE or FALSE	page A-80
_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	BOOL	TRUE or FALSE	page A-80
_EIP_LanHwErr	Communications Controller Error	NY-series Controller: Indicates that a communications controller failure occurred on the communications port 1. TRUE: Failure FALSE: Normal	BOOL	TRUE or FALSE	page A-80
_EIP1_LanHwErr	Port1 Communications Controller Error	Indicates that a communications controller failure occurred on the communications port 1. TRUE: Failure FALSE: Normal	BOOL	TRUE or FALSE	page A-81
_EIP_EtnCfgErr	Basic Ethernet Setting Error	NY-series Controller: 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	BOOL	TRUE or FALSE	page A-81

Variable name	Meaning	Function	Data type	Range of values	Reference
_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	BOOL	TRUE or FALSE	page A-81
_EIP_IPAdrCfgErr	IP Address Setting Error	NY-series Controller: 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. FALSE: Normal	BOOL	TRUE or FALSE	page A-81
_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. FALSE: Normal	BOOL	TRUE or FALSE	page A-82
_EIPIn1_IPAdrCfgErr	Internal Port1 IP Address Setting Error	Indicates the IP address setting errors for the internal port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. FALSE: Normal	BOOL	TRUE or FALSE	page A-82
_EIP_IPAdrDupErr	IP Address Duplication Error	NY-series Controller: 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.	BOOL	TRUE or FALSE	page A-82
_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.	BOOL	TRUE or FALSE	page A-82
_EIPIn1_IPAdrDupErr	Internal Port1 IP Address Duplication Error	Indicates that the same IP address is assigned to more than one node for the internal port 1. TRUE: Duplication occurred. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-82
_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: Normal	BOOL	TRUE or FALSE	page A-83

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_BootpErr	BOOTP Server Error	NY-series Controller: 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.	BOOL	TRUE or FALSE	page A-83
_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.	BOOL	TRUE or FALSE	page A-83
_EIP_IPRTblErr	IP Route Table Error	NY-series Controller: 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	BOOL	TRUE or FALSE	page A-83
_EIP_IdentityErr	Identity Error	NY-series Controller: 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	BOOL	TRUE or FALSE	page A-83
_EIP_TDLinkCfgErr	Tag Data Link Setting Error	NY-series Controller: 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	BOOL	TRUE or FALSE	page A-84
_EIP_TDLinkOpnErr	Tag Data Link Connection Failed	NY-series Controller: 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.	BOOL	TRUE or FALSE	page A-84
_EIP_TDLinkErr	Tag Data Link Communications Error	NY-series Controller: Indicates that a timeout occurred in a tag data link connection for CIP communications 1. TRUE: A timeout occurred. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-84

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TagAdrErr	Tag Name Resolution Error	NY-series Controller: 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 Controller. • There is no network variable in the Controller that corresponds to the tag setting. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-84
_EIP_MultiSwONErr	Multiple Switches ON Error	NY-series Controller: 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.	BOOL	TRUE or FALSE	page A-85
_EIP_TcpAppCfgErr	TCP Application Setting Error	TRUE: At least one of the set values for a TCP application (FTP, NTP, SNMP) is incorrect. Or, a read operation failed. FALSE: Normal	BOOL	TRUE or FALSE	page A-85
_EIP_NTPSrvErr	NTP Server Connection Error	Always FALSE for an NY-series Controller.	BOOL	TRUE or FALSE	page A-85
_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-85

Hierarchical Relationship of System-defined Variables Related to EtherNet/IP Errors in the NY-series Controller

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*, *_EIPIn1_PortErr*, *_EIP_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
		_EIPIn1_PortErr	Internal Port1 Error	_EIPIn1_IPAdrCfgErr	Internal Port1 IP Address Setting Error
				_EIPIn1_IPAdrDupErr	Internal Port1 IP Address Duplication Error
				_EIP_DNSCfgErr	DNS Setting 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 Application 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	NY-series Controller: 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.	BOOL	TRUE or FALSE	page A-85
_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.	BOOL	TRUE or FALSE	page A-86
_EIPIn1_EtnOnlineSta	Internal Port1 Online	Indicates that the built-in EtherNet/IP port's communications can be used via the internal 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.	BOOL	TRUE or FALSE	page A-86
_EIP_TDLinkRunSta	Tag Data Link Communications Status	NY-series Controller: Indicates that at least one connection is in normal operation in CIP communications 1. TRUE: Normal operation FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-86
_EIP_TDLinkAllRunSta	All Tag Data Link Communications Status	NY-series Controller: 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.	BOOL	TRUE or FALSE	page A-86
_EIP_RegTargetSta[255]	Registered Target Node Information	NY-series Controller: 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.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-87

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_EstbTargetSta [255]	Normal Target Node Information	NY-series Controller: 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.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-87
_EIP_TargetPLCModeSta [255]	Target PLC Operating Mode	NY-series Controller: 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.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-87
_EIP_TargetPLCErr [255]	Target PLC Error Information	NY-series Controller: 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.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-87

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TargetNodeErr [255]	Target Node Error Information	NY-series Controller: 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).	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-88
_EIP_NTPResult	NTP Operation Information	---	_sNTP_ RESULT		page A-88
.ExecTime	NTP Last Operation Time	NY-series Controller: No change from the initial value.	DATE_AND_ TIME	Depends on data type.	page A-88
.ExecNormal	NTP Operation Result	NY-series Controller: No change from the initial value.	BOOL	TRUE or FALSE	page A-88

Additional Information

Communications Status with Target Node

The communications status with the target node of an NY-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.

● Functional Classification: EtherNet/IP Communications Switches

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TDLinkStartCmd	Tag Data Link Communications Start Switch	NY-series Controller: 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.	BOOL	TRUE or FALSE	page A-89
_EIP_TDLinkStopCmd	Tag Data Link Communications Stop Switch	NY-series Controller: 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.	BOOL	TRUE or FALSE	page A-89

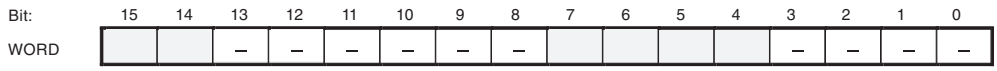
A-3-6 Meanings of Error Status Bits

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)
- *_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)
- *_EIP_CipErr* (CIP Communications 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). 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	Meaning
15	Reserved.
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 *_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-4 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-4-1 System-defined Variables for the Overall NY-series Controller (No Category)

● Functional Classification: Clock

Variable name	_CurrentTime				
Meaning	System Time			Global/local	Global
Function	This variable contains the Controller's internal clock data.				
Data type	DATE_AND_TIME			Range of values	DT#2000-01-01-00:00:00 to DT#2099-12-31-23:59:59
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-3-6 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 • ResetECError • ResetMCError • MC_Reset • MC_GroupReset You can access this variable from the user program only with the following instructions. • GetPLCError • 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 Virtual SD Memory Card is recognized in an NY-series Controller. 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	The NY-series Controller does not use this variable. The value is always FALSE.				
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	The NY-series Controller does not use this variable. The value is always FALSE.				
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	The NY-series Controller does not use this variable. The value is always FALSE.				
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	The NY-series Controller does not use this variable. The value is always FALSE.				
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 Controller was interrupted during access to the Virtual SD Memory Card. TRUE: Power was interrupted during Virtual 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			Member name	.ExecBkup
Meaning	Execute Backup Flag			Global/local	Global
Function	Change this variable to TRUE to back up Controller data to a Virtual 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	---		

Variable name	_Card1BkupCmd			Member name	.CancelBkup
Meaning	Cancel Backup Flag			Global/local	Global
Function	Change this variable to TRUE to cancel backing up data to a Virtual 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	---		

Variable name	_Card1BkupCmd			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 Virtual 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	---		

Variable name	_Card1BkupCmd			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 Virtual 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	---		

Variable name	_Card1BkupCmd			Member name	.DirName
Meaning	Directory Name			Global/local	Global
Function	Used to specify the directory name in the Virtual 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	---		

Variable name	_Card1BkupSta			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	---		

Variable name	_Card1BkupSta			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	---		

Variable name	_Card1BkupSta			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	---		

Variable name	_Card1VefySta			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	---		

Variable name	_Card1VefySta			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	---		

Variable name	_Card1VefySta			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	---		

Variable name	_Card1VefySta			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	---		

● Functional Classification: Backup

Variable name	_BackupBusy				
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	---		

● 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 Controller 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 by checking the battery-backup memory or non-volatile memory at power ON. FALSE at the following times (no failure of retention during power interruptions). • When no error is detected by checking the battery-backup memory or non-volatile memory at power ON. • 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	---		

Variable name	_SelfTest_UPSSignal				
Meaning	UPS Signal Detection Flag			Global/local	Global
Function	TRUE when a temporary power interruption signal from UPS is detected.				
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	• IPC_EnableShutdown		

Variable name	_RequestShutdown				
Meaning	Request Shutdown Flag			Global/local	Global
Function	TRUE when the power supply button is pressed while running.				
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	• IPC_EnableShutdown		

● Functional Classification: OS (Windows)

Variable name	_OSRunning				
Meaning	OS Running Flag			Global/local	Global
Function	TRUE when the Controller observes that OS (Windows) is running.				
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	• IPC_GetOSStatus • IPC_ShutdownOS • IPC_RebootOS		

Variable name	_OSHalted				
Meaning	OS Halted Flag			Global/local	Global
Function	TRUE when the Controller observes that OS (Windows) 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	• IPC_GetOSStatus • IPC_ShutdownOS • IPC_RebootOS		

Variable name	_OSErrorState				
Meaning	OS Error State Flag			Global/local	Global
Function	TRUE when the Controller determines that an error occurred in OS (Windows).				
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	• IPC_GetOSStatus • IPC_ShutdownOS • IPC_RebootOS		

● 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 Controller 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 Controller 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				
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	---		

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: Version

Variable name	_UnitVersion				
Meaning	Unit Version			Global/local	Global
Function	The unit version of the Controller 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	---		

Variable name	_HardwareRevision				
Meaning	Hardware Revision			Global/local	Global
Function	The hardware revision of the Controller is stored. Contains - if the hardware revision is in blank, and A to Z for other cases.				
Data type	STRING[2]			Range of values	- or A to Z
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: Self-diagnosis

Variable name	_SelfTest_HighTemperature				
Meaning	CPU Unit High Temperature Flag			Global/local	Global
Function	TRUE when the internal temperature of the Controller 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	---		

Variable name	_SelfTest_LowBattery				
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	---		

Variable name	_SelfTest_LowFanRevolution				
Meaning	Low FAN Revolution Flag			Global/local	Global
Function	TRUE when the fan is disconnected or the rotation speed of a fan is decreased.				
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: PLC Built-in

Variable name	_DeviceOutHoldCfg*1				
Meaning	Device Output Hold Configuration			Global/local	Global
Function	It is 16#A5A5 if you retain the target device output when the operating mode is changed or when downloaded. In the case other than 16#A5A5, the target device output is initialized when the operating mode is changed or when downloaded.				
Data type	WORD			Range of values	16#0000 to 16#FFFF
R/W access	RW	Retained	Retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1 This system-defined variable was added for unit version 1.14 of the Controller.

Variable name	_DeviceOutHoldStatus*1				
Meaning	Device Output Hold Status			Global/local	Global
Function	It is TRUE if the target device output is retained when the operating mode is changed or when downloaded. When the device output hold configuration is other than 16#A5A5, or when a major fault level Controller error occurs, the target device output is initialized and changes to FALSE.				
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 1.14 of the Controller.

A-4-2 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 instructions	• 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 instructions	• 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 instructions	• TraceTrig • TraceSamp You can access this variable from the user program only with the following instruction. • GetTraceStatus		

Variable name	_PLC_TraceSta[0..3]			Members	.ParamErr
Meaning	Trace Parameter Error Flag			Global/local	Global
Function	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.				
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 instructions	You can access this variable from the user program only with the following instruction. • GetTraceStatus		

● Functional Classification: Errors

Variable name	_PLC_ErrSta				
Meaning	PLC Function Module Error Status			Global/local	Global
Function	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-3-6 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	• GetPLCError You can use the following instruction to clear this variable. • ResetPLCError		

A-4-3 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-3-6 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-3-6 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 64 axes is shown. You can use this variable directly in the user program. Refer to *A-3-6 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_Reset		

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 32 axes groups is shown. You can use this variable directly in the user program. Refer to *A-3-6 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	ARRAY [0..31] 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		

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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561) 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	NY-series Controller: 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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561) for details on structure members.				
Data type	ARRAY[0..31] 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	NY-series Controller: 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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561) for details on structure members.				
Data type	ARRAY[0..63] OF _sAXIS_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

A-4-4 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-3-6 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	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-3-6 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-3-6 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-3-6 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-3-6 Meanings of Error Status Bits* for the meanings of the error status bits.				
Data type	Array [1..512] 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	Get EtherCAT Error Status • GetECError Reset EtherCAT Controller Error • ResetECError		

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_IndataInvalidErr*1				
Meaning	Input Process Data Invalid Error			Global/local	Global
Function	TRUE if the Input Data Invalid state continued for the following period because the EtherCAT master could not perform process data communications normally when it was in the Operational state. • When the task period is 10 ms or shorter: 100 ms • When the task period is long than 10 ms: 10 periods of the task				
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		

*1 This system-defined variable was added for unit version 1.14 of the Controller.

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			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		

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 Controller cannot establish communications within the set communications period at startup.				
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			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_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			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_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			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		

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			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		

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			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		

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			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_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.				
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.

● Functional Classification: EtherCAT Communications Diagnosis/Statistics Log

Variable name	_EC_StatisticsLogEnable				
Meaning	Diagnosis/Statistics Log Enable			Global/local	Global
Function	Changes to TRUE when the diagnosis/statistics log is started. Changes to FALSE when the diagnosis/statistics log is ended.				
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	_EC_StatisticsLogCycleSec				
Meaning	Diagnosis/Statistics Log Cycle			Global/local	Global
Function	Specifies the interval to write the diagnostic and statistical information of the diagnosis/statistics log in units of seconds. When 0 is specified, the diagnostic and statistical information is written only once when the diagnosis/statistics log is ended. **Note** The write interval does not change even if you change the value of this system-defined variable while the diagnosis/statistics log operation is in progress.				
Data type	UINT			Range of values	0, or 30 to 1800
R/W access	R/W	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EC_StatisticsLogBusy				
Meaning	Diagnosis/Statistics Log Busy			Global/local	Global
Function	TRUE while the diagnosis/statistics log operation is in progress.				
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_StatisticsLogErr				
Meaning	Diagnosis/Statistics Log Error			Global/local	Global
Function	TRUE when the diagnosis/statistics log failed to start or it is impossible to write into the log. The value of this flag is determined when *_EC_StatisticsLogBusy* (Diagnosis/Statistics Log Busy) changes to FALSE after the diagnosis/statistics log operation is started. The error end is caused by the following. • Another records cannot be added in the log file because the capacity of the Virtual SD Memory Card is fully used. • There is no Virtual SD Memory Card. • The function cannot be started because the value specified for *_EC_StatisticsLogCycleSec* (Diagnosis/Statistics Log Cycle) is invalid.				
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	---		

A-4-5 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. NY-series Controllers: Represents the collective status of the following error flags. • *_EIP1_PortErr* (Communications Port1 Error) • *_EIPIn1_PortErr* (Internal Port1 Error) • *_EIP_CipErr* (CIP Communications Error) • *_EIP_TcpAppErr* (TCP Application Communications Error) **Note** Refer to *A-3-6 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. NY-series Controllers: 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-3-6 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-3-6 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	_EIPIn1_PortErr				
Meaning	Internal Port1 Error			Global/local	Global
Function	This is the error status variable for the internal port 1. It represents the collective status of the following error flags. • *_EIPIn1_IPAdrCfgErr* (Internal Port1 IP Address Setting Error) • *_EIP1_IPAdrDupErr* (Internal Port1 IP Address Duplication 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-3-6 Meanings of Error Status Bits* on page A-51 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_CipErr				
Meaning	CIP Communications Error			Global/local	Global
Function	This is the error status variable for CIP communications. NY-series Controller: 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-3-6 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_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 Application Setting Error) • *_EIP_NTPSrvErr* (NTP Server Connection Error) **Note** Refer to *A-3-6 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	NY-series Controller: Indicates that an error occurred when the MAC address was read on the communications port 1 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	---		

Variable name	_EIP1_MacAdrErr				
Meaning	Port1 MAC Address Error			Global/local	Global
Function	Indicates that an error occurred when the MAC address was read on the communications port 1 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	---		

Variable name	_EIP_LanHwErr				
Meaning	Communications Controller Error			Global/local	Global
Function	NY-series Controller: Indicates that a communications controller failure occurred on the communications port 1. TRUE: Failure 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_LanHwErr				
Meaning	Port1 Communications Controller Error			Global/local	Global
Function	Indicates that a communications controller failure occurred on the communications port 1. TRUE: Failure 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_EtnCfgErr				
Meaning	Basic Ethernet Setting Error			Global/local	Global
Function	NY-series Controller: 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				
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				
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	NY-series Controller: 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. 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. 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	_EIPIn1_IPAdrCfgErr				
Meaning	Internal Port1 IP Address Setting Error			Global/local	Global
Function	Indicates the IP address setting errors for the internal port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. 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_IPAdrDupErr				
Meaning	IP Address Duplication Error			Global/local	Global
Function	NY-series Controller: 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.				
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.				
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	_EIPIn1_IPAdrDupErr				
Meaning	Internal Port1 IP Address Duplication Error			Global/local	Global
Function	Indicates that the same IP address is assigned to more than one node for the internal port 1. 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	_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				
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	NY-series Controller: 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.				
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.				
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_IPRTblErr				
Meaning	IP Route Table Error			Global/local	Global
Function	NY-series Controller: 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				
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_IdentityErr				
Meaning	Identity Error			Global/local	Global
Function	NY-series Controller: 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				
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	NY-series Controller: 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				
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	NY-series Controller: 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.				
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	NY-series Controller: Indicates that a timeout occurred in a tag data link connection for CIP communications 1. 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	_EIP_TagAdrErr				
Meaning	Tag Name Resolution Error			Global/local	Global
Function	NY-series Controller: 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 Controller. •There is no network variable in the Controller that corresponds to the tag setting. 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	_EIP_MultiSwONErr				
Meaning	Multiple Switches ON Error			Global/local	Global
Function	NY-series Controller: 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.				
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 Application Setting Error			Global/local	Global
Function	TRUE: At least one of the set values for a TCP 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	Always FALSE for an NY-series Controller.				
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	NY-series Controller: 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.				
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.				
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	_EIPIn1_EtnOnlineSta				
Meaning	Internal Port1 Online			Global/local	Global
Function	Indicates that the built-in EtherNet/IP port's communications can be used via the internal 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 communications of the built-in EtherNet/IP port's internal port 1 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	_EIP_TDLinkRunSta				
Meaning	Tag Data Link Communications Status			Global/local	Global
Function	NY-series Controller: Indicates that at least one connection is in normal operation in CIP communications 1. 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	_EIP_TDLinkAllRunSta				
Meaning	All Tag Data Link Communications Status			Global/local	Global
Function	NY-series Controller: 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.				
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	NY-series Controller: 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.				
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	NY-series Controller: 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.				
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	NY-series Controller: 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.				
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	NY-series Controller: 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.				
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_TargetNodeErr [255]				
Meaning	Target Node Error Information			Global/local	Global
Function	NY-series Controller: 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).				
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_NTPResult			Member name	.ExecTime
Meaning	NTP Last Operation Time			Global/local	Global
Function	NY-series Controller: No change from the initial value.				
Data type	Structure: _sNTP_RESULT Members: DATE_AND_TIME			Range of values	Depends on data type.
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	You can read the contents of this variable with the GetNTPStatus instruction.		

Variable name	_EIP_NTPResult			Member name	.ExecNormal
Meaning	NTP Operation Result			Global/local	Global
Function	NY-series Controller: No change from the initial value.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	You can read the contents of this variable with the GetNTPStatus instruction.		

● Functional Classification: EtherNet/IP Communications Switches

Variable name	_EIP_TDLinkStartCmd				
Meaning	Tag Data Link Communications Start Switch			Global/local	Global
Function	NY-series Controller: 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.				
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	NY-series Controller: 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.				
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-5 Attributes of Controller Data

The following table shows the attributes of the Controller data including the Retain/Non-retain attribute in the following cases: power interruption, power on, operating mode change, and major fault level Controller error.

Controller 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.

Controller 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 Virtual SD Memory Card settings Shutdown wait time settings Event log 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.
	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.
	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.
		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.
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.
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.

Controller 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 execution ID in Controller			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 non-volatile memory).	Same as before power interruption.	Retained.	Retained.	Supported.	Not retained.	RUN/PROGRAM mode	Supported.
Event logs	Logs	System log User event log	Retained (with non-volatile memory).	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 non-volatile memory).	Same as before power interruption.	Retained (continued).	Retained (continued).	Supported.	Not retained.		Not supported.

A-6 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.

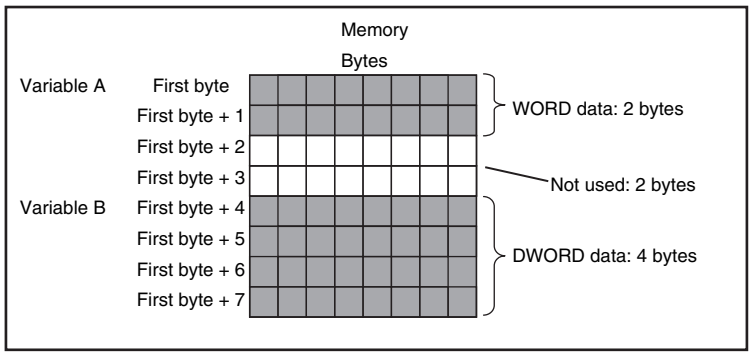
- When using EtherNet/IP tag data links or CIP messages to access variables between NY-series Controller and other Controllers
- When using structure variables to exchange data with devices other than Controllers, such as ID Tags

A-6-1 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.

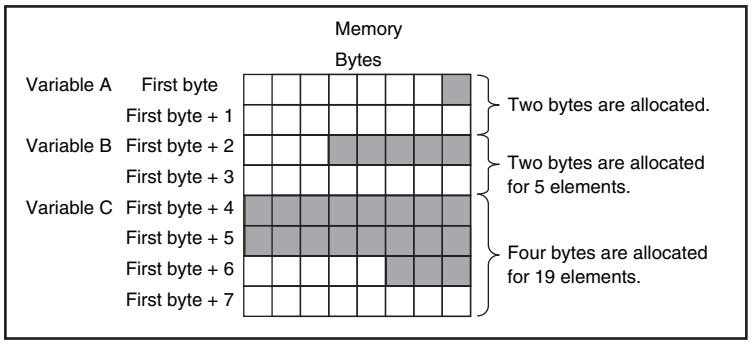
Data type	Alignment	Amount of memory that is allocated
Array	Same as alignment of the data type of the elements	(Amount of memory that is allocated for the data type of the elements) × Number of elements*
Structure	The largest alignment of all of the members	The integral multiple of the alignment that 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.

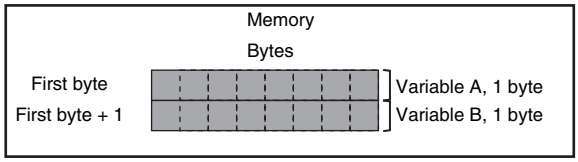
Specific examples of the rules for memory allocation for variables of each data type are given below.

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



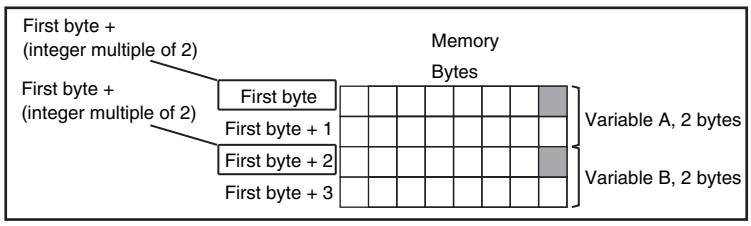
Variable Table

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



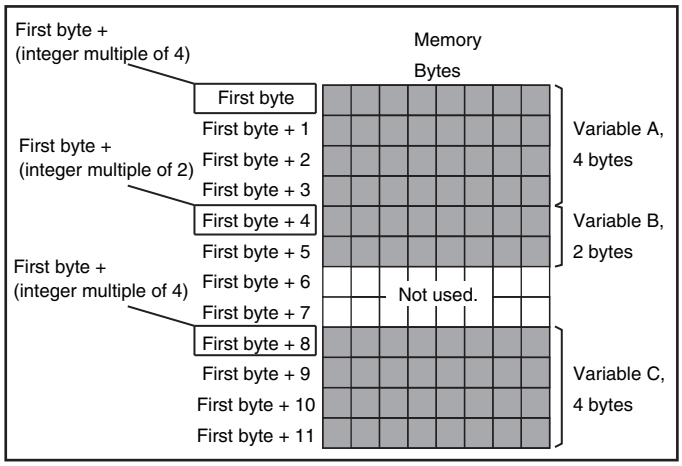
Variable Table

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



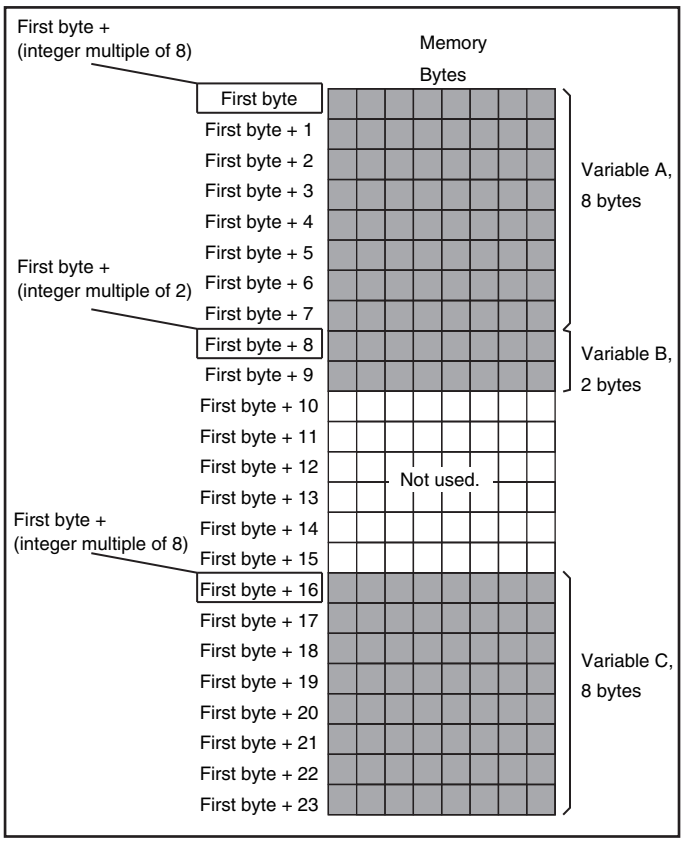
Variable Table

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



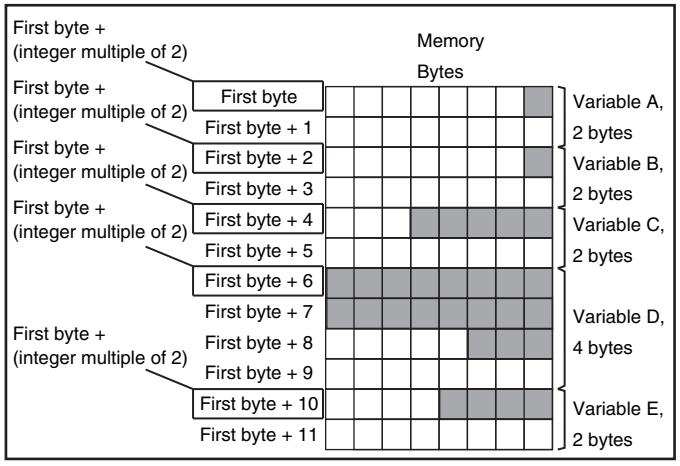
Variable Table

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.

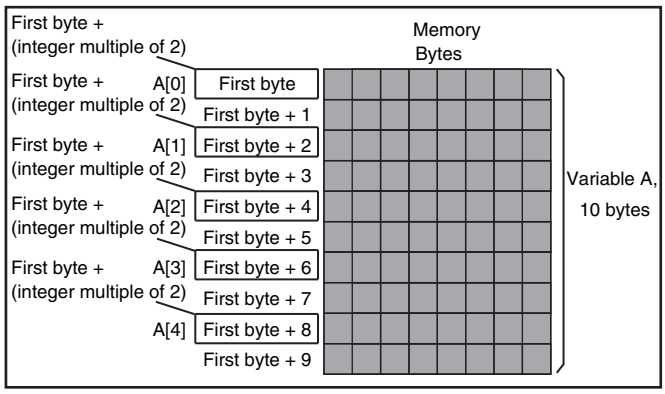
Example: Continuous variables in the following order: two BOOL variable, one BOOL array with five elements, one BOOL array with 19 elements, and one BOOL array with four elements



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

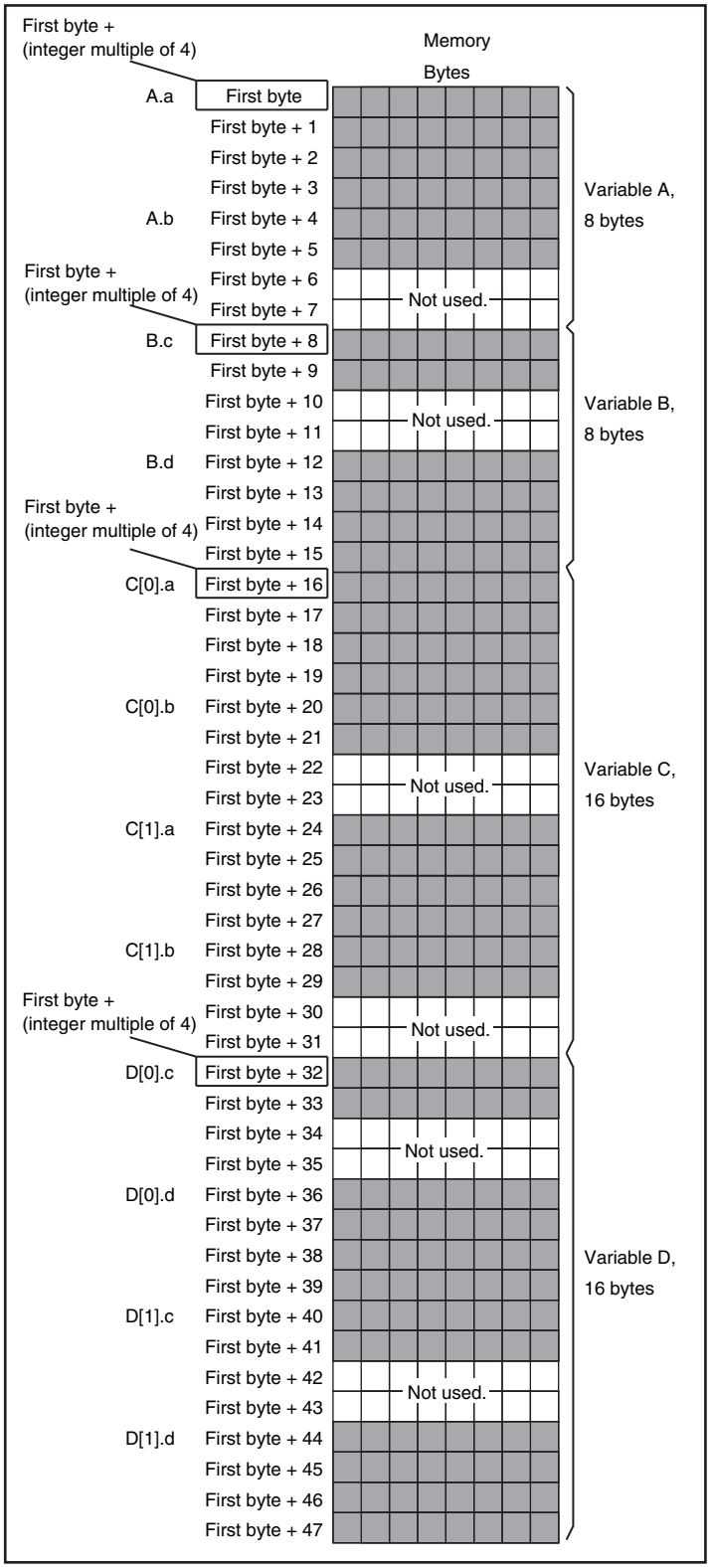
Name	Data type
Variable A	ARRAY[0..4] OF INT

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

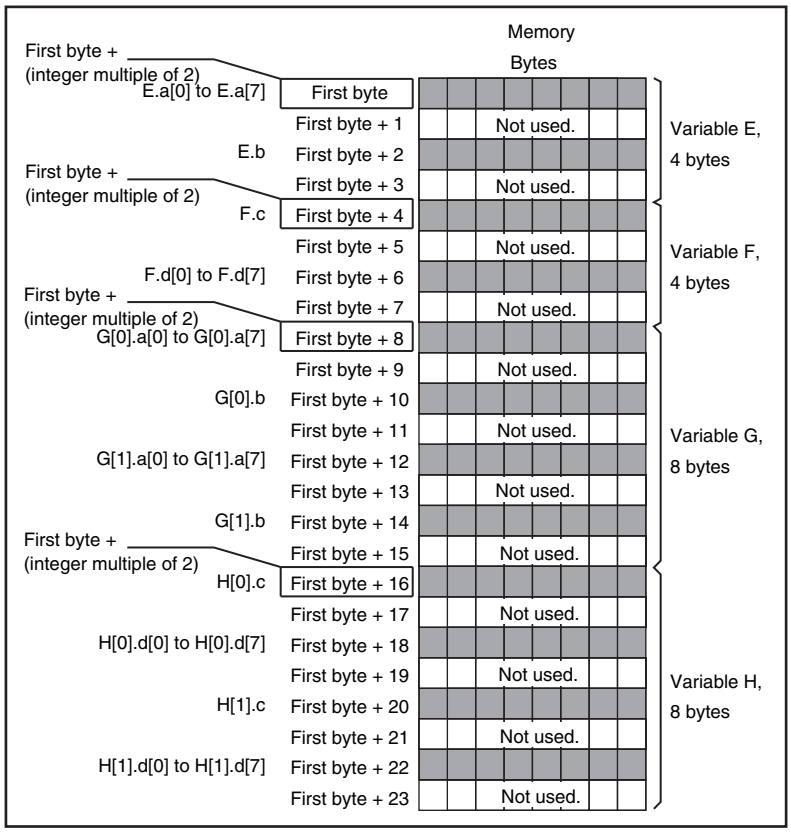
Name	Data type
Structure *STR_B*	STRUCT
c	INT
d	DINT

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

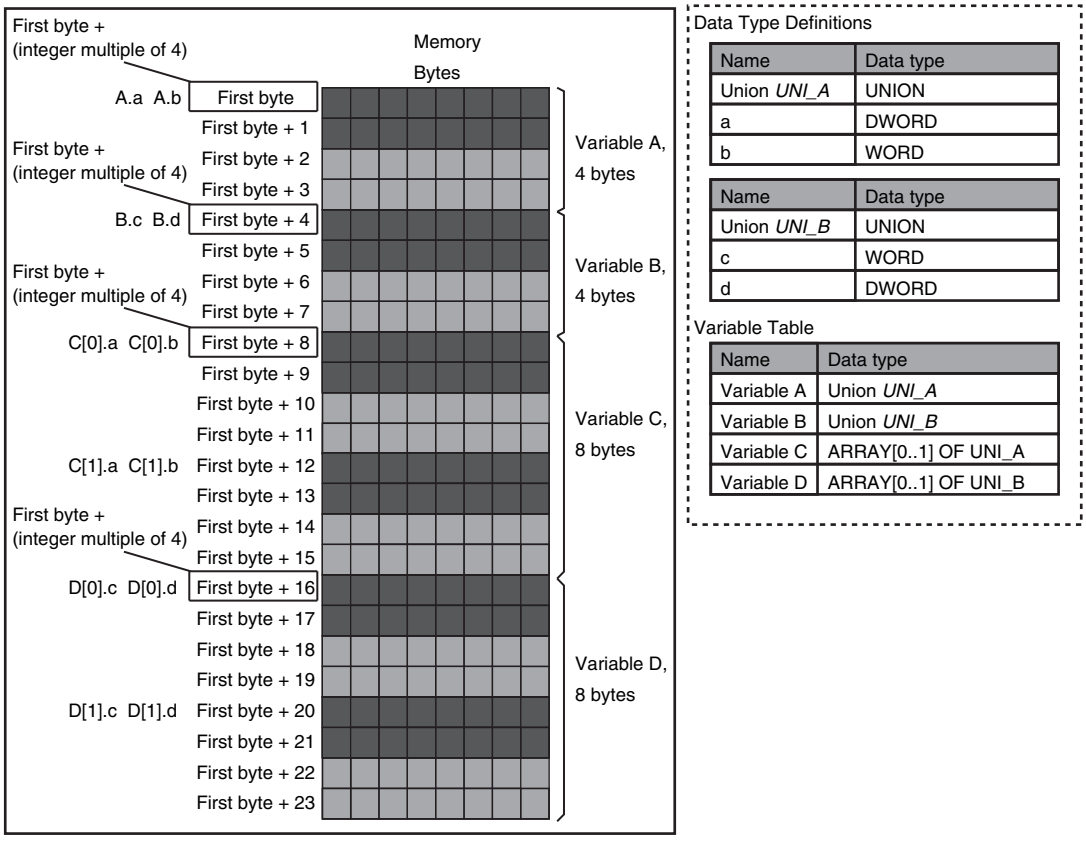
Name	Data type
Variable E	Structure *STR_C*
Variable F	Structure *STR_D*
Variable G	ARRAY[0..1] OF STR_C
Variable H	ARRAY[0..1] OF STR_D

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-6-2 Important Case Examples

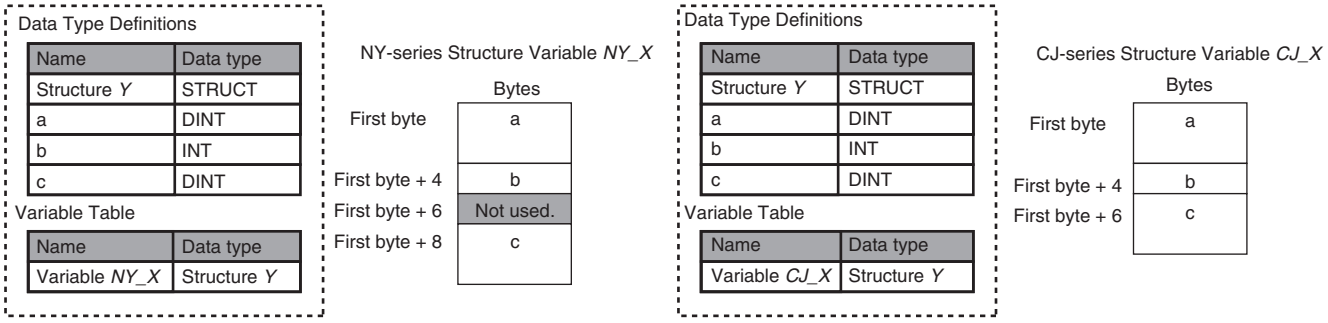
When you exchange structure variable data between an NY-series Controller and a remote device, you must align the memory configuration of the structure variable members with those of the remote device. This section describes what to do in either the NY-series Controller or in the remote device.

Additional Information

This is not necessary when you exchange data between NY-series Controllers.

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 NY-series Controller and a CJ-series CPU Unit are shown below.
This section describes how to align the memory configuration for these Units.



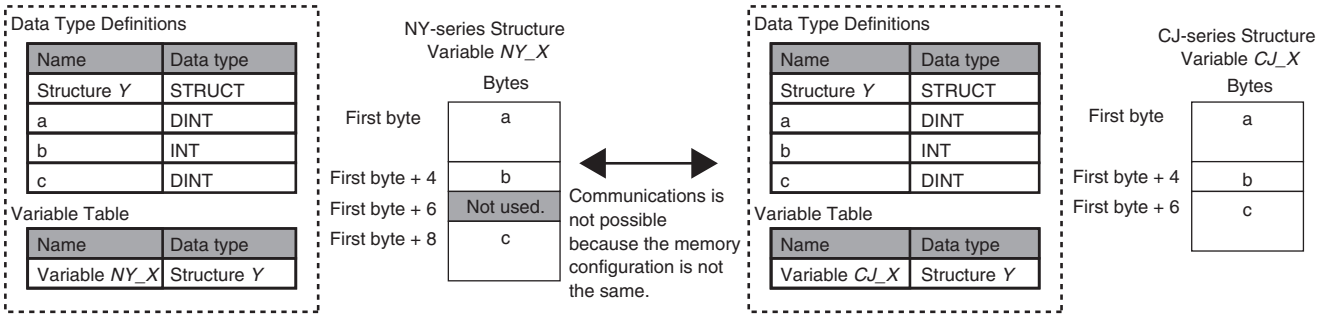
● Method 1: Changing the Memory Configuration of the Structure Variable in the NY-series Controller

With an NY-series Controller, 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 NY-series Controller so that it is the same as the memory configuration in a remote device that the NY-series Controller 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.

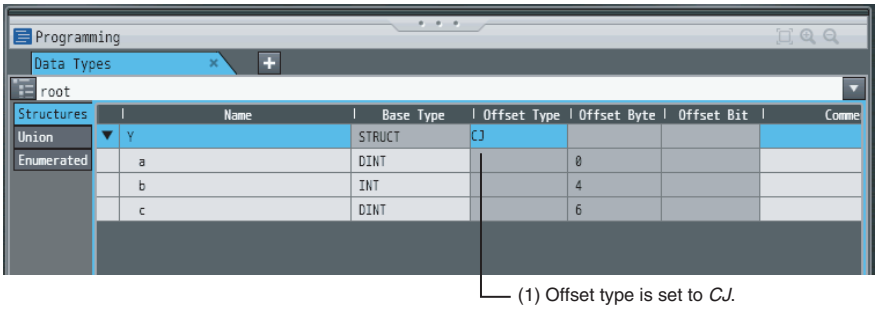
If you change the memory configuration of a structure variable by setting offsets, you must make the same changes for the same structure variable in other NY-series Controllers on the network.
Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No W504-E1-03 or later) for the procedure to change the memory configuration of a structure variable.

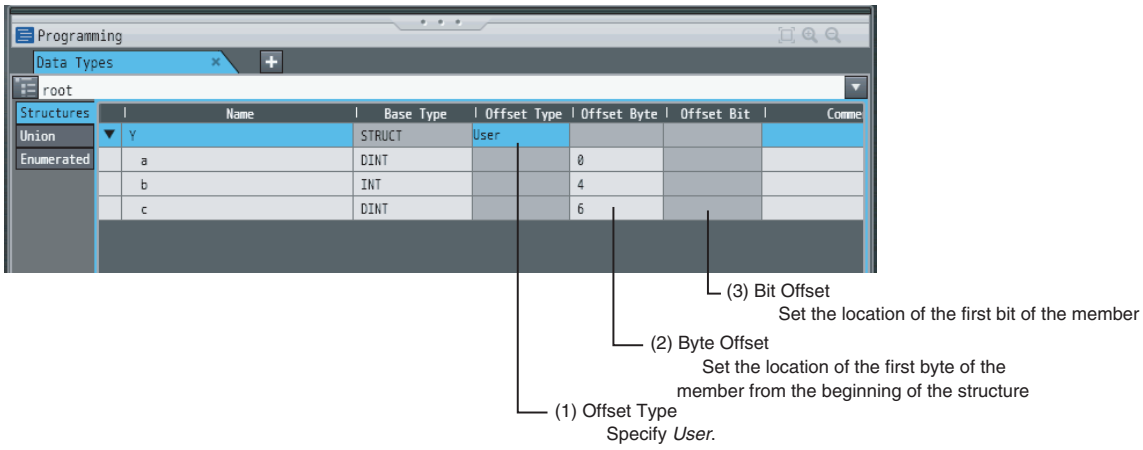
Example: The following example shows how the memory configuration of the structure variable in the CJ-series CPU Unit is changed to match the memory configuration of the structure variable in the NY-series Controller.

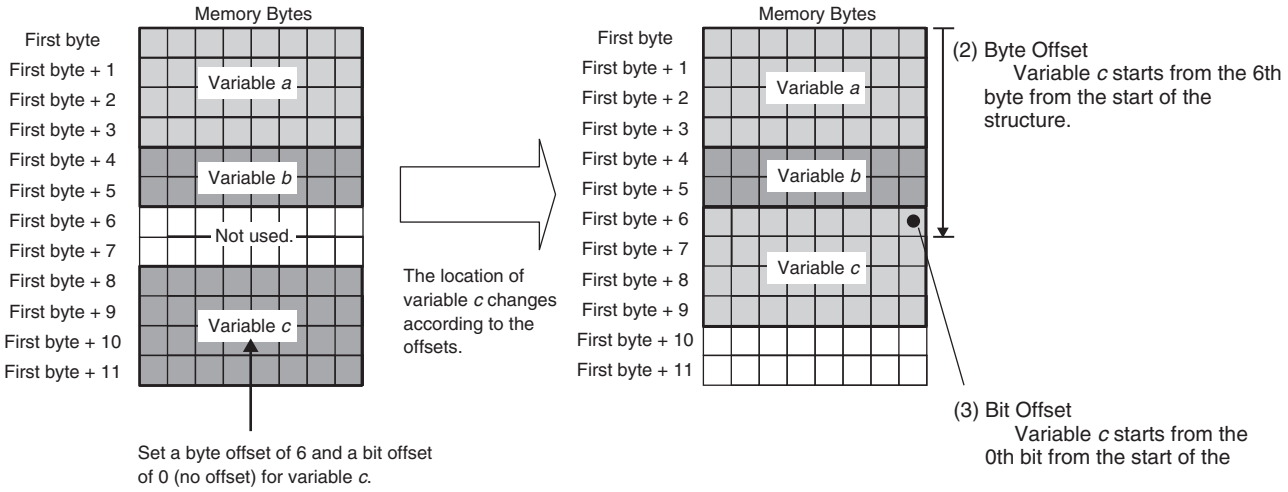


To align the memory configurations in the NY-series Controller and CJ-series CPU Unit, offsets are set in the Sysmac Studio.

Here, the following offsets are set for member c of data type Y of the structure variable NY_X.



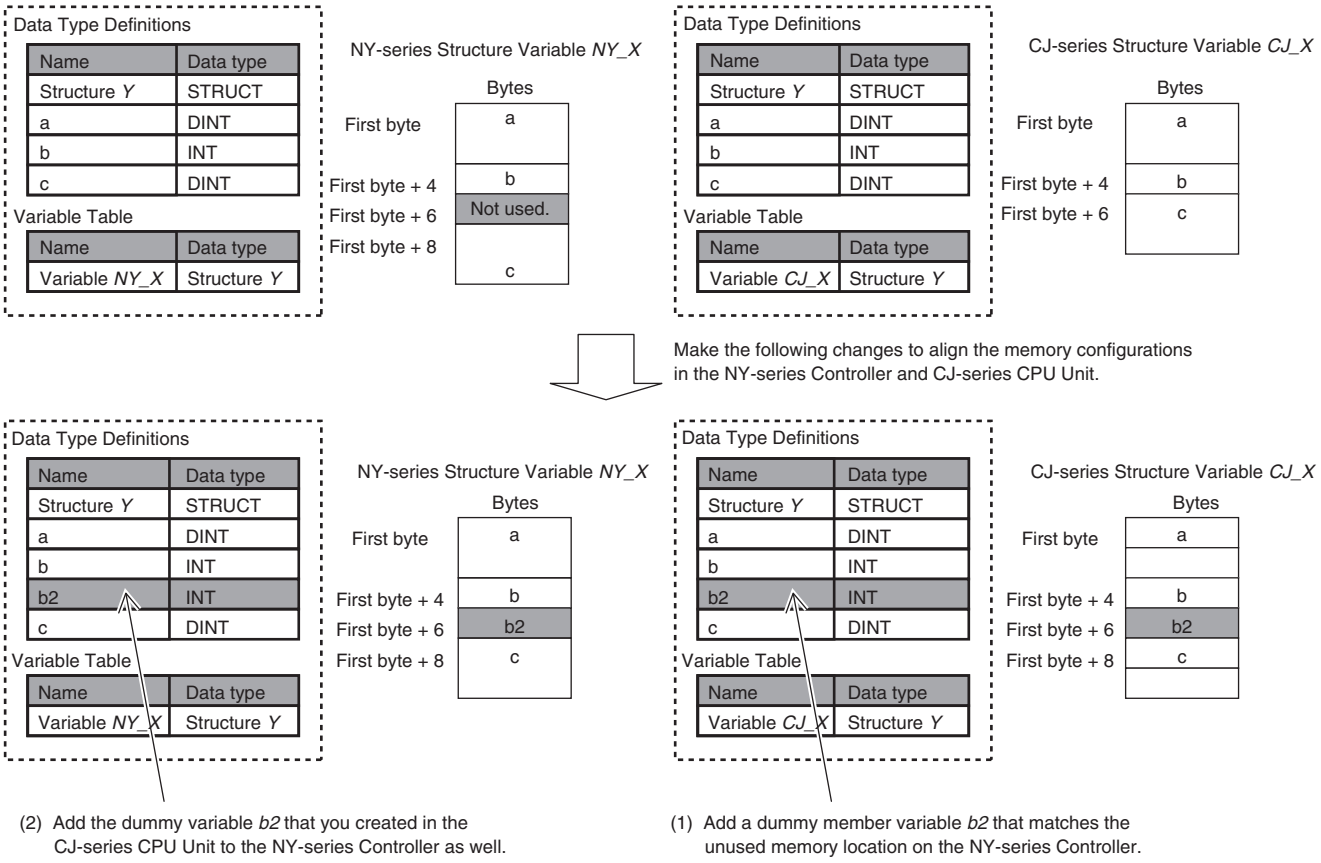




● Method 2: Changing the Memory Configuration of the Structure Variable in the Remote Device

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 NY-series Controllers. 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 NY-series Controllers.

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 NY-series Controllers.



A-7 Registering a Symbol Table on the CX-Designer

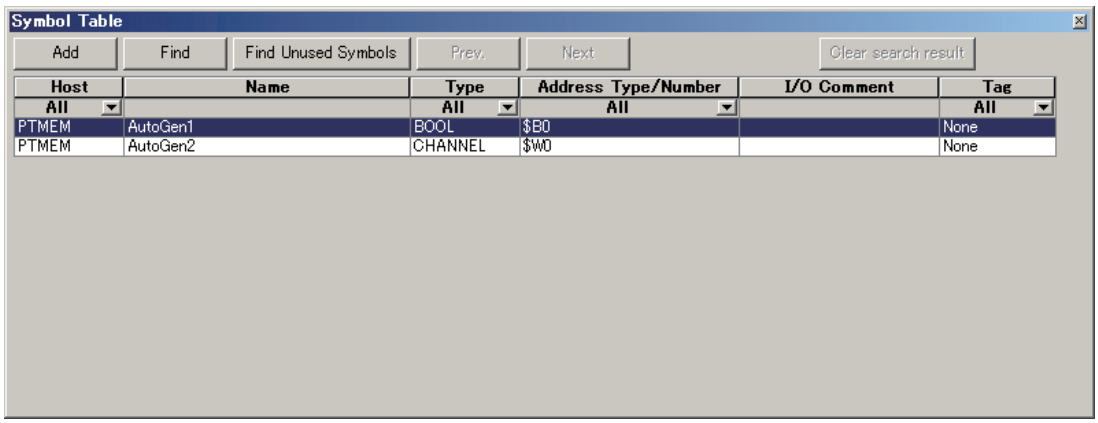
When you connect the NY-series Controller to an NS-series PT, you can use variables on the CX-Designer to set addresses for the functional objects. The variables are managed in a symbol table. This section shows how to copy a table of variables from a Microsoft Excel spreadsheet to register them all 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	_Card1BkupCmd.ExecBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.ExecVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.DirName	STRING(64)			TRUE
HOST3	_Card1BkupSta.Done	BOOL			TRUE
HOST3	_Card1BkupSta.Active	BOOL			TRUE
HOST3	_Card1BkupSta.Err	BOOL			TRUE
HOST3	_Card1VefySta.Done	BOOL			TRUE
HOST3	_Card1VefySta.Active	BOOL			TRUE
HOST3	_Card1VefySta.VefyRslt	BOOL			TRUE
HOST3	_Card1VefySta.Err	BOOL			TRUE
HOST3	_BackupBusy	BOOL			TRUE

2 Start the CX-Designer and open the Symbol Table Dialog Box.

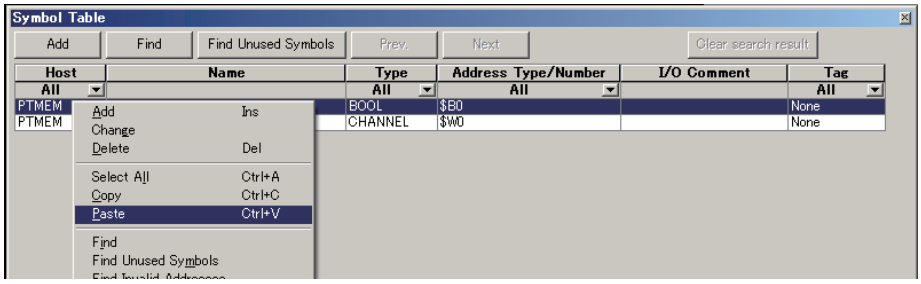


3 **Copy the shaded portion of the Microsoft Excel spreadsheet.**

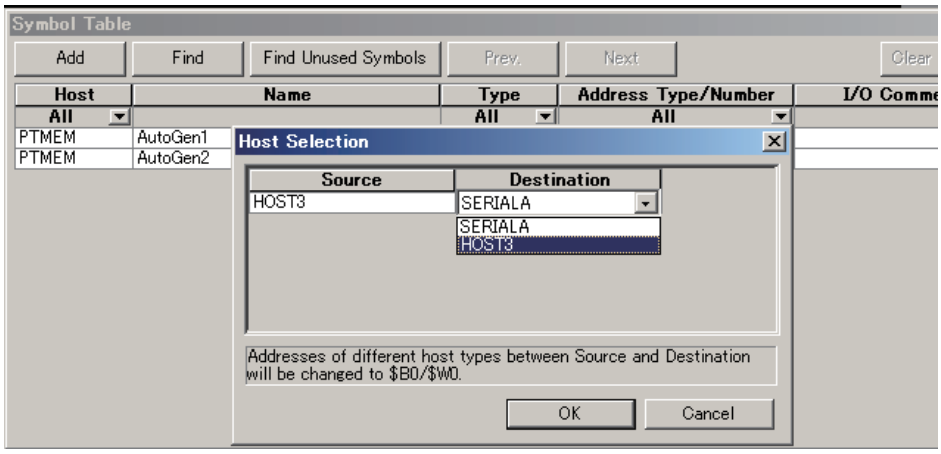
Always copy all of the columns that are shown below.

Host	Name	Type	Address type/address	I/O comment	Tag
HOST3	_Card1BkupCmd.ExecBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.ExecVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.DirName	STRING(64)			TRUE
HOST3	_Card1BkupSta.Done	BOOL			TRUE
HOST3	_Card1BkupSta.Active	BOOL			TRUE
HOST3	_Card1BkupSta.Err	BOOL			TRUE
HOST3	_Card1VefySta.Done	BOOL			TRUE
HOST3	_Card1VefySta.Active	BOOL			TRUE
HOST3	_Card1VefySta.VefyRslt	BOOL			TRUE
HOST3	_Card1VefySta.Err	BOOL			TRUE
HOST3	_BackupBusy	BOOL			TRUE

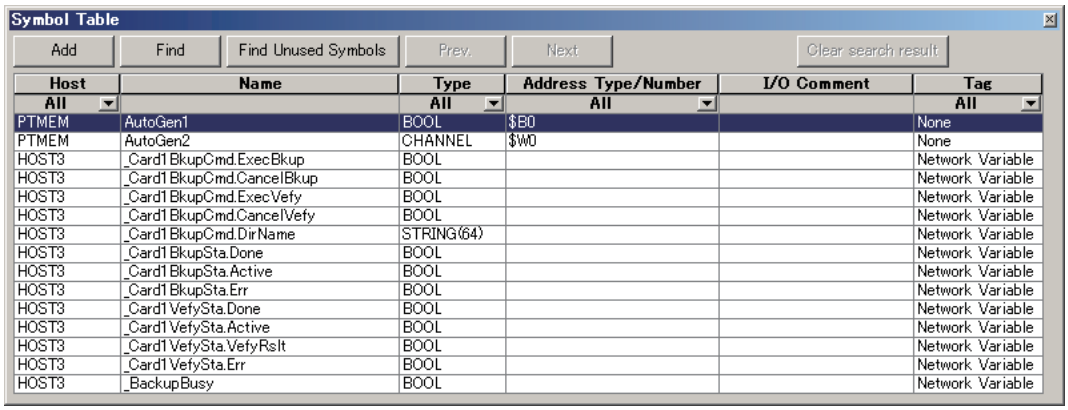
4 **Right-click in the Symbol Table Dialog Box in the CX-Designer and select Paste from the menu.**



5 **In the Host Selection Dialog Box on the CX-Designer, select the NY-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	_Card1 BkupCmd.ExecBkup	BOOL			Network Variable
HOST3	_Card1 BkupCmd.CancelBkup	BOOL			Network Variable
HOST3	_Card1 BkupCmd.ExecVefy	BOOL			Network Variable
HOST3	_Card1 BkupCmd.CancelVefy	BOOL			Network Variable
HOST3	_Card1 BkupCmd.DirName	STRING(64)			Network Variable
HOST3	_Card1 BkupSta.Done	BOOL			Network Variable
HOST3	_Card1 BkupSta.Active	BOOL			Network Variable
HOST3	_Card1 BkupSta.Err	BOOL			Network Variable
HOST3	_Card1 VefySta.Done	BOOL			Network Variable
HOST3	_Card1 VefySta.Active	BOOL			Network Variable
HOST3	_Card1 VefySta.VefyRslt	BOOL			Network Variable
HOST3	_Card1 VefySta.Err	BOOL			Network Variable
HOST3	_BackupBusy	BOOL			Network Variable

A-8 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.

A-8-1 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-110.
- You can disable an EtherCAT slave to enable removing it or installing it on the EtherCAT network.

A-8-2 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

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 *NY-series Instructions Reference Manual* (Cat. No. W560) 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 *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561) for the detailed specifications of the MC_ChangeAxisUse instruction.

A-8-3 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)

_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-8-4 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 *NY-series Instructions Reference Manual* (Cat. No. W560) 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.

A-8-5 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 *NY-series Instructions Reference Manual* (Cat. No. W560) for the detailed specifications of the PrgStatus instruction.

A-8-6 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.

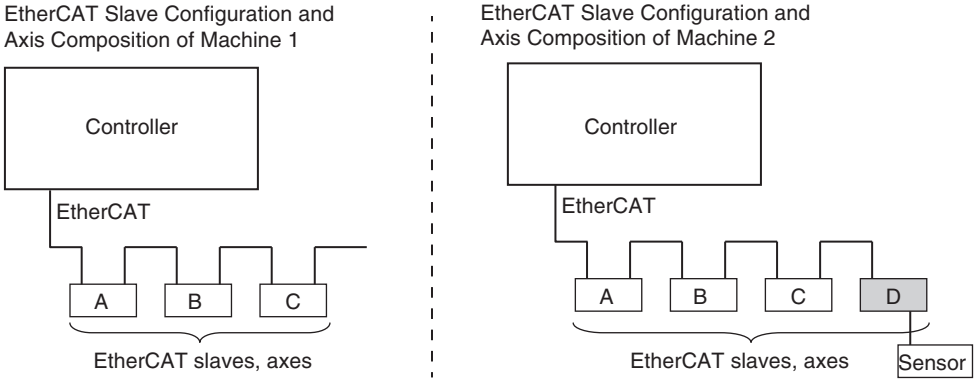
1. Right-click **Task Settings** under **Configurations and Setup** and select **Edit** from the menu.
   The Task Settings Tab Page is displayed.
2. Click the **Program Assignment Settings** Button.
   The Program Assignment Settings Display appears.
3. Set **Initial Status** of the program to **Run** on the Program Assignment Settings Display.

A-8-7 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 NY-series Controllers are different for machines 1 and 2 as shown below. These two machines are centrally managed using one Sysmac Studio project.



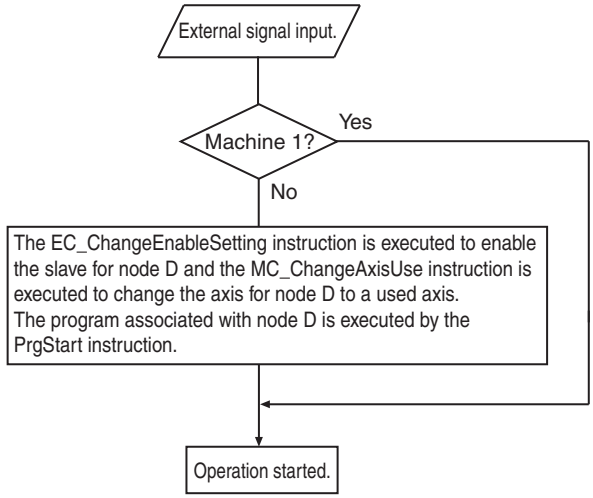
In the Sysmac Studio project, an EtherCAT Slave Configuration is created for all four EtherCAT slaves and axes in A, B, C, and D in the figure. Then, on the Sysmac Studio, you set the EtherCAT slave enable/disable settings, Axis Use parameter settings, and the associated program run/stop status at the 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.

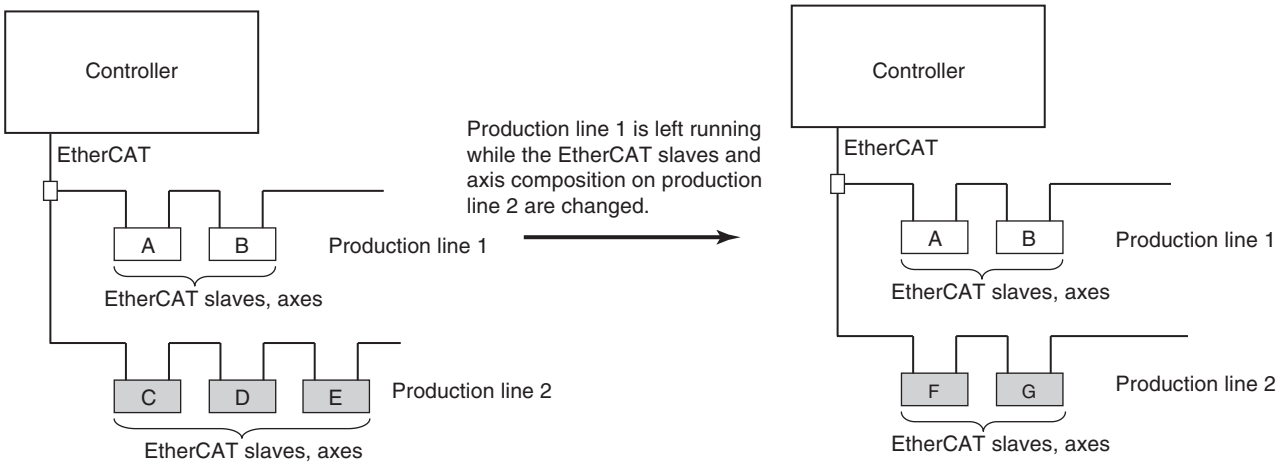
EtherCAT slave	Installed in EtherCAT network	Enable/disable setting	Axis Use parameter setting	Associated programs
A, B, and C	Installed.	Enabled.	Used Axis	Executed.
D	Installed.	Enabled.	Used Axis	Executed.

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-9 Size Restrictions for the User Program

There are size restrictions for the user program due to the limitations of the memory capacity in the Controller 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 Controller.

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 Controller 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-9-1 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	NY5□2-□□□□
Maximum number of programs	500
Maximum value of the following: Number of function block definitions + Number of function definitions + Number of ladder diagram sections	3,000
Maximum total number of input, output, and in-out variables in function block and function definitions	64

● POU Instance Restrictions

POU instances are subject to the following restrictions.

Restriction	NY5□2-□□□□
Maximum number of POU instances	24,000

Refer to *Number of POU Instances* on page A-115 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	NY5□2-□□□□
Maximum total size in Mbytes of variables without a Retain attribute*1	64
Maximum total size*1 in Mbytes of variables with a Retain attribute	4
Maximum number of variables*2 without a Retain attribute	180,000
Maximum number of variables*3 with a Retain attribute	40,000
Maximum number of network variables	40,000

*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-115 for information on counting variables.

*3 Refer to *Number of Variables with a Retain Attribute* on page A-116 for information on counting variables.

● Restrictions to Variable Definitions

Variable definitions are subject to the following restrictions.

Restriction	NY5□2-□□□□
Maximum number of elements per array	65,535
Maximum number of dimensions in an array	3
Maximum size in MB of an array	8
Maximum value of a subscript (element number) for an array	65,535
Maximum size in bytes*1 of a string variable	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	NY5□2-□□□□
Maximum number*1 of data type definitions	4,000
Maximum number of levels in a structure definition	8
Maximum number of members in a structure definition	2,048
Maximum size in MB of a structure variable	8
Maximum number of members in a union definition	4
Maximum number of enumerators in an enumeration definition	2,048

*1 Refer to *Number of Data Type Definitions* on page A-116 for information on counting data types.

Restrictions to Constants (Literals)

The constants (literals) are subject to the following restrictions.

Restriction	NY5□2-□□□□
Maximum size in bytes of a constant (literal)	1,985

A-9-2 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 NY-series Controller. 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-10 Version Information for NY-series Controllers

This section describes the relationship between the unit versions of NY-series Controllers and the Sysmac Studio versions, and the functions that are supported for each unit version.

A-10-1 Relationship between Unit Versions of Controllers and Sysmac Studio Versions

This section also describes how the unit versions of NY-series Controllers 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 NY-series Controllers and the corresponding Sysmac Studio versions.

Unit version of NY-series Controller	Corresponding version of Sysmac Studio
Ver. 1.16*1	Ver. 1.20
Ver. 1.14*2	Ver. 1.18
Ver. 1.12	Ver. 1.17

*1 There is no NY-series Controller with unit version 1.15.
*2 There is no NY-series Controller with unit version 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 NY-series Controllers are given in this section.

● Using Sysmac Studio Version 1.16 or Lower

You cannot use the NY-series Controller with Sysmac Studio version 1.16 or lower.

A-10-2 Functions That Were Added or Changed for Each Unit Version

This section describes the functions that were added or changed for each unit version of NY-series Controller.

● 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 Controller. For details, refer to the *NY-series Instructions Reference Manual* (Cat. No. W560) and *NY-series Motion Control Instructions Reference Manual* (Cat. No. W561).

● Additions and Changes to Controller Events

The events that can occur have increased or changed for the new unit version of the Controller. There are also changes in the recovery methods to use when some errors occur.
For details, refer to the *NY-series Troubleshooting Manual* (Cat. No. W564).

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 Controller. Refer to *A-3 System-defined Variables* for details.

Index

Index

Numerics

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

OMRON Corporation **Industrial Automation Company**
Kyoto, 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:

Cat. No. W558-E1-03 1017