OMRON

Machine Automation Controller

NJ/NX-series

CPU Unit

Software User's Manual

NX701-1□□□
NX502-1□□□
NX102-1□□□
NX102-90□□
NX1P2-1□□□□□
NX1P2-9□□□□□
NJ501-□□□□
NJ301-1□□□
NJ101-10□□
NJ101-90□□

CPU Unit

SYSMAC
always in control

W501-E1-44

NOTE

1. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.
2. 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.
3. 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, Visual Basic, and Microsoft Edge 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.

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

Copyrights

- Microsoft product screen shots used with permission from Microsoft.
- 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/.

Introduction

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

Intended Audience

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

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

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

Applicable Products

This manual covers the following products.

- NX-series CPU Units
  - NX701-17□□
  - NX701-16□□
  - NX502-15□□
  - NX502-14□□
  - NX502-13□□
  - NX102-12□□
  - NX102-11□□
  - NX102-10□□
  - NX102-90□□
  - NX1P2-11□□□□
  - NX1P2-11□□□□1
  - NX1P2-10□□□□
  - NX1P2-10□□□□1
  - NX1P2-90□□□□
  - NX1P2-90□□□□1
  - NX1P2-9B□□□□
  - NX1P2-9B□□□□1
- NJ-series CPU Units
  - NJ501-□5□□
  - NJ501-□4□□
  - NJ501-□3□□
  - NJ301-12□□
  - NJ301-11□□
  - NJ101-10□□
  - NJ101-90□□

Part of the specifications and restrictions for the products may be given in other manuals.
Refer to *Relevant Manuals* on page 2 and *Related Manuals* on page 26.

Relevant Manuals

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

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

Purpose of use	Manual																			
	Basic information																			
	NX-series CPU Unit Hardware User's Manual	NX-series NX502 CPU Unit Hardware User's Manual	NX-series NX102 CPU Unit Hardware User's Manual	NX-series NX1P2 CPU Unit Hardware User's Manual	NJ-series CPU Unit Hardware User's Manual	NJ/NX-series CPU Unit Software User's Manual	NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual	NJ/NX-series Instructions Reference Manual	NJ/NX-series CPU Unit Motion Control User's Manual	NJ/NX-series Motion Control Instructions Reference Manual	NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual	NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual	NJ/NX-series CPU Unit User's Manual	NX-series CPU Unit FINS User's Manual	NJ/NX-series Database Connection CPU Units User's Manual	NJ-series SECS/GEM CPU Units User's Manual	NJ-series Robot Integrated CPU Unit User's Manual	NJ-series NJ Robotics CPU Unit User's Manual	NJ/NY-series NC Integrated Controller User's Manual	NJ/NX-series Troubleshooting Manual
Introduction to NX701 CPU Units	○																			
Introduction to NX502 CPU Units		○																		
Introduction to NX102 CPU Units			○																	
Introduction to NX1P2 CPU Units				○																
Introduction to NJ-series Controllers					○															
Setting devices and hardware																				
Using motion control	○	○	○	○	○				○											
Using EtherCAT											○									
Using EtherNet/IP												○								
Using robot control for OMRON robots																	○			

Purpose of use	Manual																							
	Basic information																							
	NX-series CPU Unit Hardware User's Manual	NX-series NX102 CPU Unit Hardware User's Manual	NX-series NX502 CPU Unit Hardware User's Manual	NX-series NX1P2 CPU Unit Hardware User's Manual	NJ-series CPU Unit Hardware User's Manual	NJ/NX-series CPU Unit Software User's Manual	NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual	NJ/NX-series Instructions Reference Manual	NJ/NX-series CPU Unit Motion Control User's Manual	NJ/NX-series Motion Control Instructions Reference Manual	NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual	NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual	NJ/NX-series CPU Unit User's Manual	NX-series CPU Unit FINS User's Manual	NJ/NX-series Database Connection CPU Units User's Manual	NJ-series SECS/GEM CPU Units User's Manual	NJ-series Robot Integrated CPU Unit User's Manual	NJ-series NJ Robotics CPU Unit User's Manual	NJ/NY-series NC Integrated Controller User's Manual	NJ/NX-series Troubleshooting Manual
Software settings						○																		
Using motion control									○															
Using EtherCAT											○													
Using EtherNet/IP												○												
Using OPC UA													○											
Using FINS														○										
Using the database connection service															○									
Using the GEM Services						○										○								
Using robot control for OMRON robots																	○							
Using robot control by NJ Robotics function																		○						
Using numerical control																				○				
Using the NX1P2 CPU Unit functions							○																	
Writing the user program						○		○																
Using motion control									○	○														
Using EtherCAT											○													
Using EtherNet/IP												○												
Using OPC UA													○											
Using FINS														○										
Using the database connection service															○									
Using the GEM Services						○										○								
Using robot control for OMRON robots																	○							
Using robot control by NJ Robotics function																		○						
Using numerical control																				○				
Programming error processing																								○
Using the NX1P2 CPU Unit functions							○																	

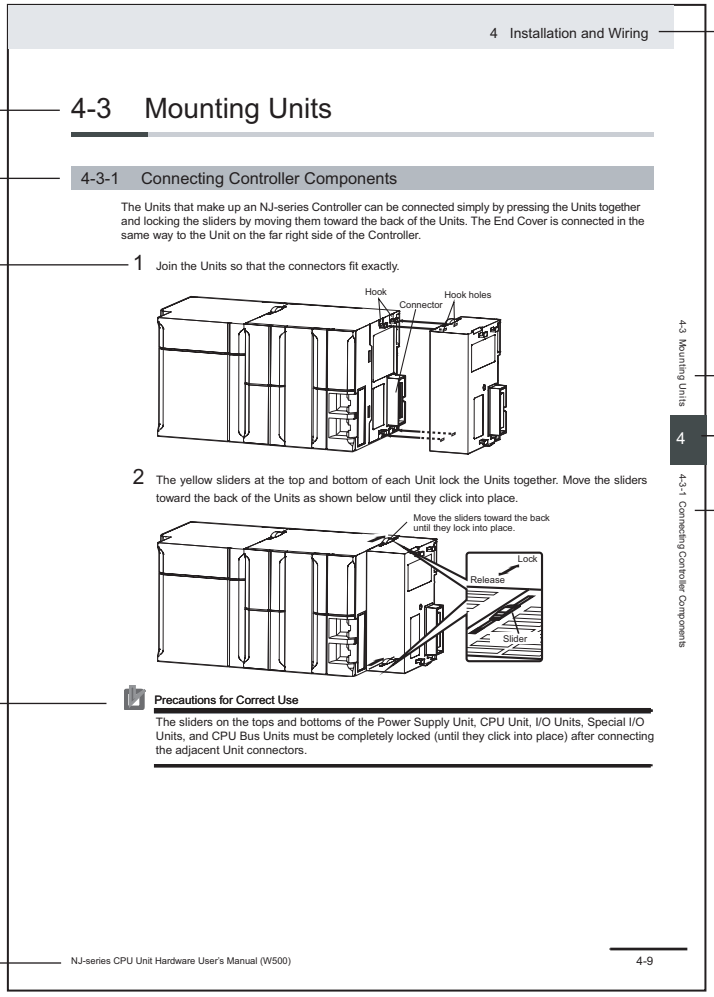
Purpose of use	Manual																					
	Basic information																					
	NX-series CPU Unit Hardware User's Manual	NX-series NX502 CPU Unit Hardware User's Manual	NX-series NX102 CPU Unit Hardware User's Manual	NX-series NX1P2 CPU Unit Hardware User's Manual	NJ-series CPU Unit Hardware User's Manual	NJ/NX-series CPU Unit Software User's Manual	NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual	NJ/NX-series Instructions Reference Manual	NJ/NX-series CPU Unit Motion Control User's Manual	NJ/NX-series Motion Control Instructions Reference Manual	NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual	NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual	NJ/NX-series CPU Unit OPC UA User's Manual	NX-series CPU Unit FINS User's Manual	NJ/NX-series Database Connection CPU Units User's Manual	NJ-series SECS/GEM CPU Units User's Manual	NJ-series Robot Integrated CPU Unit User's Manual	NJ-series NJ Robotics CPU Unit User's Manual	NJ/NY-series NC Integrated Controller User's Manual	NJ/NX-series Troubleshooting Manual
Testing operation and debugging						○																
Using motion control									○													
Using EtherCAT											○											
Using EtherNet/IP												○										
Using OPC UA													○									
Using FINS														○								
Using the database connection service															○							
Using the GEM Services																○						
Using robot control for OMRON robots																	○					
Using robot control by NJ Robotics function																		○				
Using numerical control																			○			
Using the NX1P2 CPU Unit functions							○															
Learning about error management and corrections*1													△	△	△	△	△	△	△	○		
Maintenance	○	○	○	○	○																	
Using motion control									○													
Using EtherCAT											○											
Using EtherNet/IP												○										

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

Manual Structure

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.

Icons indicate precautions, additional information, or reference information.

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.

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

1 Introduction to NJ/NX-series Controllers

2 CPU Unit Operation

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

4 Controller Setup

5 Designing Tasks

6 Programming

7 Checking Operation and Actual Operation

8 CPU Unit Functions

9 Backup Functions

10 Communications Setup

11 Example of Actual Application Procedures

12 Troubleshooting

A Appendices

I Index

CONTENTS

Introduction 1
Intended Audience 1
Applicable Products 1

Relevant Manuals 2

Manual Structure 5
Page Structure 5
Special Information 6
Precaution on Terminology 6

Sections in this Manual 7

Terms and Conditions Agreement 18
Warranty, Limitations of Liability 18
Application Considerations 19
Disclaimers 19
Statement of security responsibilities for assumed use cases and against threats 20

Safety Precautions 21

Precautions for Safe Use 22

Precautions for Correct Use 23

Regulations and Standards 24
Software Licenses and Copyrights 24

Versions 25
Unit Versions of CPU Units and Sysmac Studio Versions 25

Related Manuals 26

Terminology 31

Revision History 36

Section 1 Introduction to NJ/NX-series Controllers

1-1 The NJ/NX-series Controllers 1-2
1-1-1 Features 1-2
1-1-2 Introduction to the System Configurations 1-5

1-2 Main Specifications 1-17

1-3 Overall Operating Procedure for the NJ/NX-series 1-23
1-3-1 Overall Procedure 1-23
1-3-2 Procedure Details 1-24

Section 2 CPU Unit Operation

2-1 Overview of CPU Unit Operation 2-2

2-1-1 Introduction to CPU Unit 2-2
2-1-2 Overview of Operation According to CPU Unit Status 2-3
2-2 Software 2-4
2-2-1 Software Configuration 2-4
2-2-2 Operation of Software 2-4
2-3 Accessing I/O 2-13
2-3-1 Type of Variables 2-13
2-3-2 Accessing I/O with Variables 2-16
2-4 I/O Refreshing of NX Bus Function Module 2-26
2-4-1 I/O Refreshing Methods 2-26
2-4-2 I/O Refreshing Method Operation 2-27
2-5 I/O Refreshing of X Bus Function Module 2-30
2-6 Sequence Control and Motion Control 2-31
2-6-1 Overview of Control 2-31
2-6-2 Sequence Control System 2-32
2-6-3 Motion Control System 2-34
2-6-4 Synchronizing Sequence Control and Motion Control 2-36
2-7 Overview of CPU Unit Data 2-38
2-8 Operation for CPU Unit Status 2-40
2-8-1 CPU Unit Status 2-40
2-8-2 Operation for CPU Unit Status 2-41
2-8-3 Operating Modes 2-43

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

3-1 Procedure to Create the Slave and Unit Configurations 3-2
3-2 Creating and Comparing the Slave and Unit Configurations 3-5
3-2-1 Creating the EtherCAT Slave Configuration 3-5
3-2-2 Creating the Unit Configuration 3-6
3-2-3 Verifying the Unit Configuration 3-6
3-3 I/O Ports and Device Variables 3-8
3-3-1 I/O Ports 3-8
3-3-2 I/O Port Names 3-9
3-3-3 Device Variables 3-11
3-4 Allocating Variables to Units 3-14
3-4-1 Procedure to Assign Variables to Units 3-14
3-4-2 Using Variables Assigned to Units 3-15
3-5 Creating the Axes and Assigning Them to the Servo Drives/Encoder Input Slaves/NX Units 3-19
3-5-1 Introduction 3-19
3-5-2 Axis Variables and Axes Group Variables 3-19
3-5-3 Creating and Using Axes and Axis Variables 3-21

Section 4 Controller Setup

4-1 Overview of the Controller Setup 4-2
4-2 Initial Settings for the PLC Function Module 4-4
4-2-1 Introduction 4-4
4-2-2 Controller Setup 4-4
4-2-3 Task Settings 4-7
4-2-4 Configuration and Setup for NX Units 4-13
4-2-5 Configuration and Setup for X Bus Units 4-16
4-2-6 Unit Configuration and Unit Setup for NJ-series CPU Units 4-19
4-3 Initial Settings for NX Units 4-21

4-3-1 NX Unit Settings 4-21
4-3-2 I/O Allocation Settings 4-22
4-3-3 Unit Operation Settings 4-24
4-4 Initial Settings for X Bus Units 4-25
4-4-1 X Bus Unit Settings 4-25
4-5 Initial Settings for Special Units 4-26
4-6 Initial Settings for the Motion Control Function Module 4-28
4-6-1 Introduction 4-28
4-6-2 Setting Methods 4-28
4-7 Initial Settings for the EtherCAT Master Function Module 4-30
4-8 Initial Settings for the EtherNet/IP Function Module 4-31
4-9 Initial Settings for Built-in I/O 4-32
4-10 Initial Settings for Option Boards 4-33
4-11 Memory Settings for CJ-series Units 4-34
4-11-1 Setting Procedure 4-34
4-11-2 Setting Screen 4-34
4-11-3 Settings 4-35

Section 5 Designing Tasks

5-1 Overview of Task Designing Procedure 5-3
5-2 Overview of Tasks 5-6
5-2-1 Tasks 5-6
5-2-2 Instructions Related to Tasks 5-8
5-2-3 System-defined Variables Related to Tasks 5-8
5-3 Specifications and Basic Operation of Tasks for NX701 5-11
5-3-1 Specifications of Tasks for NX701 CPU Units 5-11
5-3-2 Guidelines for Separating Tasks for NX701 5-11
5-3-3 Basic Operation of Tasks for NX701 CPU Units 5-12
5-3-4 Event Task Execution Conditions for NX701 CPU Units 5-20
5-3-5 Event Task Execution Timing for NX701 CPU Units 5-25
5-3-6 Operation When Execution Condition Is Met Again Before Execution of the Event Task Is Completed 5-29
5-4 Specifications and Basic Operation of Tasks for NX502 CPU Units 5-31
5-4-1 Specifications of Tasks for NX502 CPU Units 5-31
5-4-2 Guidelines for Separating Tasks for NX502 CPU Units 5-31
5-4-3 Basic Operation of Tasks for NX502 CPU Units 5-32
5-4-4 Event Task Execution Conditions for NX502 CPU Units 5-40
5-4-5 Event Task Execution Timing for NX502 CPU Units 5-45
5-4-6 Operation When Execution Condition Is Met Again Before Execution of the Event Task Is Completed 5-49
5-5 Specifications and Basic Operation of Tasks for NX102 CPU Units and NX1P2 CPU Units 5-51
5-5-1 Specifications of Tasks for NX102 CPU Units and NX1P2 CPU Units 5-51
5-5-2 Guidelines for Separating Tasks for NX102 CPU Units and NX1P2 CPU Units 5-51
5-5-3 Basic Operation of Tasks for NX102 CPU Units and NX1P2 CPU Units 5-52
5-5-4 Event Task Execution Conditions for NX102 CPU Units and NX1P2 CPU Units 5-57
5-5-5 Event Task Execution Timing for NX102 CPU Units and NX1P2 CPU Units 5-62
5-5-6 Operation When Execution Condition Is Met Again Before Execution of the Event Task Is Completed 5-66
5-6 Specifications and Basic Operation of Tasks for NJ-series Controllers 5-68
5-6-1 Specifications of Tasks for NJ-series Controllers 5-68
5-6-2 Guidelines for Separating Tasks for NJ-series Controllers 5-68
5-6-3 Basic Operation of Tasks for NJ-series Controllers 5-69
5-6-4 Event Task Execution Conditions for NJ-series Controllers 5-76
5-6-5 Event Task Execution Timing for NJ-series Controllers 5-81

5-6-6 Operation When Execution Condition Is Met Again Before Execution of the Event Task Is Completed 5-85

5-7 Services Other Than Tasks 5-86
5-7-1 Execution Priorities and Execution Orders of Services Other Than Tasks 5-88
5-7-2 Processing Performed in and Execution Timing of the Tag Data Link Service 5-92
5-7-3 Processing Performed in and Execution Timing of the Option Board Service 5-96
5-7-4 Processing Performed in and Execution Timing of the Communications Bridge Service 5-97
5-7-5 Processing Performed in and Execution Timing of the System Services 5-98

5-8 Assignment and Settings Related to Tasks 5-102
5-8-1 Assigning I/O Refreshing to Tasks 5-102
5-8-2 Assigning Tasks to Programs 5-109
5-8-3 Parameters for Primary Periodic Task and Periodic Tasks 5-110

5-9 Ensuring Concurrency of Variable Values 5-114
5-9-1 Ensuring Concurrency of Variable Values between Tasks 5-114
5-9-2 Variable Access from Outside the Controller 5-120

5-10 Errors Related to Tasks 5-125

5-11 Monitoring Task Execution Status and Task Execution Times 5-128

5-12 Task Design Methods and I/O Response Times 5-133
5-12-1 Checking the Task Execution Time 5-133
5-12-2 Examples of Task Design 5-136
5-12-3 System Input and Output Response Times 5-137

Section 6 Programming

6-1 Overview of Programming Procedures 6-3

6-2 POUs (Program Organization Units) 6-5
6-2-1 What Are POUs? 6-5
6-2-2 Overview of the Three Types of POUs 6-5
6-2-3 Differences between Programs, Functions, and Function Blocks 6-6
6-2-4 Details on Programs 6-7
6-2-5 Details on Function Blocks 6-8
6-2-6 Details on Functions 6-16
6-2-7 Operation That Applies to Both Functions and Function Blocks 6-22
6-2-8 POU Restrictions 6-25

6-3 Variables 6-28
6-3-1 Variables 6-28
6-3-2 Types of Variables 6-28
6-3-3 Types of User-defined Variables in Respect to POUs 6-28
6-3-4 Attributes of Variables 6-30
6-3-5 Data Types 6-32
6-3-6 Derivative Data Types 6-41
6-3-7 Array Specifications and Range Specifications for Data Types 6-51
6-3-8 Variable Attributes 6-60
6-3-9 Changes to Variables for Status Changes 6-69
6-3-10 Function Block Instances 6-83
6-3-11 Monitoring Variable Values 6-83
6-3-12 Restrictions on Variable Names and Other Program-related Names 6-84

6-4 Constants (Literals) 6-86
6-4-1 Constants 6-86
6-4-2 Notation for Different Data Types 6-86

6-5 Programming Languages 6-91
6-5-1 Programming Languages 6-91
6-5-2 Ladder Diagram Language 6-91
6-5-3 Structured Text Language 6-98

6-6 Instructions 6-134
6-6-1 Instructions 6-134
6-6-2 Basic Understanding of Instructions 6-134

6-6-3 Instruction Errors 6-136

6-7 Namespaces 6-142
6-7-1 Namespaces 6-142
6-7-2 Namespace Specifications 6-143
6-7-3 Procedure for Using Namespaces 6-146

6-8 Libraries 6-147
6-8-1 Introduction to Libraries 6-147
6-8-2 Specifications of Libraries 6-147
6-8-3 Library Object Specifications 6-148
6-8-4 Procedure to Use Libraries 6-149

6-9 Programming Precautions 6-151
6-9-1 Array Specifications for Input Variables, Output Variables, In-Out Variables 6-151
6-9-2 Structure Variables for Input Variables, Output Variables, In-Out Variables 6-151
6-9-3 Master Control 6-152

Section 7 Checking Operation and Actual Operation

7-1 Overview of Steps in Checking Operation and Actual Operation 7-2

7-2 Offline Debugging 7-3
7-2-1 Features of Simulation 7-3
7-2-2 Simulation Execution 7-3
7-2-3 Setting Up Simulations 7-6

7-3 Checking Operation on the Actual System and Actual Operation 7-8
7-3-1 Checking Operation on the Actual System and Actual Operation 7-8
7-3-2 Downloading the Project 7-8
7-3-3 Checking I/O Wiring 7-9
7-3-4 MC Test Run 7-9
7-3-5 Checking the Operation of the User Program 7-10
7-3-6 Starting Actual Operation 7-11

Section 8 CPU Unit Functions

8-1 Data Management, Clock, and Operating Functions 8-3
8-1-1 Clearing All Memory 8-3
8-1-2 Clock 8-3
8-1-3 RUN Output 8-6

8-2 Management Functions for NX Units 8-7
8-2-1 NX Bus Function Module 8-7
8-2-2 Mounting Settings of NX Units on the CPU Unit 8-11
8-2-3 Restarting NX Units on the CPU Unit 8-14
8-2-4 Checking Wiring for NX Units on the CPU Unit 8-15
8-2-5 Fail-soft Operation for NX Units on the CPU Unit 8-16
8-2-6 Monitoring Total Power-ON Time for NX Units on the CPU Unit 8-19
8-2-7 Clearing All Memory for Each NX Unit on the CPU Unit 8-20

8-3 Management Function for X Bus Units 8-22
8-3-1 X Bus Function Module 8-22
8-3-2 X Bus Unit Common Function Module 8-23
8-3-3 X Bus Unit Mounting Settings 8-24
8-3-4 X Bus Unit Setting Synchronization Function 8-26
8-3-5 Clearing All Memory for Each X Bus Unit 8-26

8-4 Management Functions for CJ-series Units 8-27
8-4-1 Basic I/O Units 8-27
8-4-2 Special Units 8-28

8-5 SD Memory Card Operations 8-30
8-5-1 SD Memory Card Operations 8-30
8-5-2 Specifications of Supported SD Memory Cards, Folders, and Files 8-31

8-5-3 SD Memory Card Operation Instructions 8-34
8-5-4 FTP Client Communications Instructions 8-34
8-5-5 FTP Server 8-35
8-5-6 File Operations from the Sysmac Studio 8-35
8-5-7 SD Memory Card Life Expiration Detection 8-35
8-5-8 List of System-defined Variables Related to SD Memory Cards 8-36
8-5-9 SD Memory Card Self-diagnostic Functions 8-37
8-5-10 Exclusive Control of File Access in SD Memory Cards 8-39

8-6 Security 8-40
8-6-1 Authentication of User Program Execution IDs 8-41
8-6-2 User Program Transfer with No Restoration Information 8-44
8-6-3 Overall Project File Protection 8-45
8-6-4 Data Protection 8-46
8-6-5 Secure Communication Function 8-48
8-6-6 Secure Socket Services 8-48
8-6-7 TCP/UDP Port Close Function 8-49
8-6-8 Packet Filter 8-49
8-6-9 Complete Data Erasure Function 8-49
8-6-10 Operation Authority Verification 8-51
8-6-11 User Authentication 8-53
8-6-12 CPU Unit Write Protection 8-57
8-6-13 Firmware Update Prohibition 8-58
8-6-14 CPU Unit Names and Serial IDs 8-59
8-6-15 Access log 8-60

8-7 Debugging 8-61
8-7-1 Forced Refreshing 8-61
8-7-2 Changing Present Values 8-65
8-7-3 Online Editing 8-67
8-7-4 Data Tracing 8-68
8-7-5 Differential Monitoring 8-75
8-7-6 Automation Playback 8-80

8-8 Event Logs 8-81
8-8-1 Introduction 8-81
8-8-2 Detailed Information on Event Logs 8-83
8-8-3 Controller Events (Controller Errors and Information) 8-87
8-8-4 User-defined Events (User-defined Errors and Information) 8-88

8-9 Changing Event Levels 8-95
8-9-1 Applications of Changing Event Levels 8-95
8-9-2 Events for Which the Event Level Can Be Changed 8-95
8-9-3 Procedure to Change an Event Level 8-95

8-10 Displaying Firmware Update Log 8-98

Section 9 Backup Functions

9-1 The Backup Functions 9-3
9-1-1 Applications of Backup Functions 9-4
9-1-2 Examples of Operating Procedures for the Backup Functions 9-4
9-1-3 Data that Is Backed Up 9-6
9-1-4 Types of Backup Functions 9-8
9-1-5 Relation between the Different Types of Backup Functions and Data Groups 9-11
9-1-6 Applicable Range of the Backup Functions 9-12

9-2 SD Memory Card Backups 9-15
9-2-1 Backup (Controller to SD Memory Card) 9-16
9-2-2 Restore (SD Memory Card to Controller) 9-21
9-2-3 Verify (between Controller and SD Memory Card) 9-29

9-3 Disabling Backups to SD Memory Cards 9-35

9-4 Automatic Transfers from SD Memory Cards 9-37

9-5 Program Transfer from SD Memory Card 9-39

9-6 Sysmac Studio Controller Backups 9-46
9-6-1 Backup (Controller to Computer) 9-47
9-6-2 Restore (Computer to Controller) 9-48
9-6-3 Verify (between Controller and Computer) 9-49

9-7 Importing and Exporting Sysmac Studio Backup File Data 9-51

9-8 Sysmac Studio Variable and Memory Backup Functions 9-53
9-8-1 Applicable Data for Sysmac Studio Variable and Memory Backup Functions 9-53
9-8-2 Using Sysmac Studio Variable and Memory Backup Functions 9-53
9-8-3 Compatibility between CPU Unit Models 9-54

9-9 Backup Functions When EtherCAT Slaves Are Connected 9-57
9-9-1 Backed Up EtherCAT Slave Data 9-57
9-9-2 Backup Support Depending on the Controller Status 9-57
9-9-3 Conditions for Restoring EtherCAT Slave Data 9-58
9-9-4 EtherCAT Slaves for Which You Can Back Up Data 9-59

9-10 Backup Functions When EtherCAT Slave Terminals Are Connected 9-62
9-10-1 Backing Up Data in an EtherCAT Slave Terminal 9-62
9-10-2 Backup Support Depending on the EtherCAT Slave Terminal Status 9-63
9-10-3 Conditions for Restoring EtherCAT Slave Terminal Data 9-63

9-11 Backup Functions When NX Units Are Connected 9-65
9-11-1 Backing Up Data in NX Units on the CPU Unit 9-65
9-11-2 Backup Support Depending on the Controller Status 9-65
9-11-3 Conditions for Restoring NX Unit Data on the CPU Unit 9-66

9-12 Backup Functions When X Bus Units Are Connected 9-67
9-12-1 Backing Up Data in X Bus Units on the CPU Unit 9-67
9-12-2 Backup Support Depending on the Controller Status 9-67
9-12-3 Conditions for Restoring X Bus Unit Data on the CPU Unit 9-67

9-13 Backup Functions When CJ-series Units Are Connected 9-68
9-13-1 Backed Up CJ-series Unit Data 9-68
9-13-2 Backup Support Depending on the Controller Status 9-68
9-13-3 Conditions for Restoring CJ-series Unit Data 9-68

9-14 Backup-related Files 9-70
9-14-1 Types of Backup-related Files 9-70
9-14-2 Specifications of a Backup File 9-71
9-14-3 Specifications of a Restore Command File 9-72
9-14-4 Specifications of an Automatic Transfer Command File 9-74
9-14-5 Specifications of a Controller Verification Results File 9-76
9-14-6 Specifications of an EtherCAT Verification Results File 9-77
9-14-7 Specifications of an EtherCAT Slave Terminal Verification Results File 9-78
9-14-8 Specifications of an NX Unit Verification Results File 9-79
9-14-9 Specifications of an X Bus Unit Verification Results File 9-80
9-14-10 Specifications of a CJ-series Unit Verification Results File 9-82

9-15 Compatibility between Backup-related Files 9-84
9-15-1 Compatibility between Backup Functions 9-84
9-15-2 Compatibility between CPU Unit Models 9-85
9-15-3 Compatibility between Unit Versions of CPU Units 9-86

9-16 Functions that cannot be Executed during Backup Functions 9-88

Section 10 Communications Setup

10-1 Communications System Overview 10-2
10-1-1 Introduction 10-3

10-2 Connection with Sysmac Studio 10-9
10-2-1 Configurations That Allow Online Connections 10-9
10-2-2 Configurations That Do Not Allow Online Connections 10-11
10-2-3 Secure Communication 10-12

10-3 Connection with Other Controllers or Slaves 10-15

10-3-1 Connection Configurations between Controllers 10-15
10-3-2 Connection Configuration between Controllers and Slaves 10-18

10-4 Connection with HMIs or Serial Communications Devices 10-19
10-4-1 Connection with HMIs 10-19
10-4-2 Connection with Serial Communications Devices 10-20

Section 11 Example of Actual Application Procedures

11-1 Example Application 11-2
11-1-1 System Configuration 11-2
11-1-2 Operation 11-2

11-2 Overview of the Example Procedure 11-3
11-2-1 Wiring and Settings 11-3
11-2-2 Software Design 11-3
11-2-3 Software Settings from the Sysmac Studio 11-4
11-2-4 Programming with the Sysmac Studio 11-7
11-2-5 Simulation with the Sysmac Studio 11-8
11-2-6 Checking Operation and Starting Operation on the Actual System 11-9

Section 12 Troubleshooting

12-1 Overview of Troubleshooting 12-2

Appendices

A-1 Specifications A-4
A-1-1 General Specifications A-4
A-1-2 Performance Specifications A-4
A-1-3 Function Specifications A-18

A-2 Calculating Guidelines for the Real Processing Times of Tasks for the NX701 System A-30
A-2-1 Calculating the Average Real Processing Times of Tasks A-31
A-2-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-39

A-3 Calculating Guidelines for the Real Processing Times of Tasks for the NX502 System A-42
A-3-1 Calculating the Average Real Processing Times of Tasks A-42
A-3-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-49

A-4 Calculating Guidelines for the Real Processing Times of Tasks for the NX102 System A-53
A-4-1 Calculating the Average Real Processing Times of Tasks A-54
A-4-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-61

A-5 Calculating Guidelines for the Real Processing Times of Tasks for the NX1P2 System A-64
A-5-1 Calculating the Average Real Processing Times of Tasks A-65
A-5-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-72

A-6 Calculating Guidelines for the Real Processing Times of Tasks for the NJ-series System A-75
A-6-1 Calculating the Average Real Processing Times of Tasks A-76
A-6-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-87

A-7 System-defined Variables A-91
A-7-1 System-defined Variables for the Overall NJ/NX-series Controller (No Category) A-92
A-7-2 PLC Function Module, Category Name: _PLC A-102
A-7-3 PLC Function Module, Category Name: _CJB A-107

A-7-4 PLC Function Module, Category Name: _APB A-110
A-7-5 NX Bus Function Module, Category Name: _NXB A-111
A-7-6 X Bus Function Module, Category Name: _XBU A-114
A-7-7 Motion Control Function Module, Category Name: _MC A-117
A-7-8 EtherCAT Master Function Module, Category Name: _EC A-119
A-7-9 EtherNet/IP Function Module, Category Name: _EIP A-126
A-7-10 Meanings of Error Status Bits A-161

A-8 Specifications for Individual System-defined Variables A-163
A-8-1 System-defined Variables for the Overall NJ/NX-series Controller (No Category) A-163
A-8-2 PLC Function Module, Category Name: _PLC A-183
A-8-3 PLC Function Module, Category Name: _CJB A-189
A-8-4 PLC Function Module, Category Name: _APB A-193
A-8-5 NX Bus Function Module, Category Name: _NXB A-195
A-8-6 X Bus Function Module, Category Name: _XBU A-199
A-8-7 Motion Control Function Module, Category Name: _MC A-201
A-8-8 EtherCAT Master Function Module, Category Name: _EC A-204
A-8-9 EtherNet/IP Function Module, Category Name: _EIP A-214

A-9 Attributes of CPU Unit Data A-241

A-10 Contents of Memory Used for CJ-series Units A-247
A-10-1 CIO Area A-247
A-10-2 Internal I/O Area A-250
A-10-3 Holding Area A-250
A-10-4 DM Area A-250
A-10-5 EM Area A-251

A-11 Variable Memory Allocation Methods A-252
A-11-1 Variable Memory Allocation Rules A-252
A-11-2 Important Case Examples A-261

A-12 Registering a Symbol Table on the CX-Designer A-265

A-13 Enable/Disable EtherCAT Slave and Axes A-268
A-13-1 Project Settings When Using EtherCAT Slaves and Axes A-268
A-13-2 Using Instructions to Enable/Disable EtherCAT Slaves and Axes A-268
A-13-3 System-defined Variables That Indicate EtherCAT Slave or Axis Status A-269
A-13-4 Enabling/Disabling Execution of Program A-270
A-13-5 Checking Enabled/Disabled Program A-270
A-13-6 Settings with the Sysmac Studio A-271
A-13-7 Examples of Applications of Enabling/Disabling EtherCAT Slaves and Axes A-272

A-14 Size Restrictions for the User Program A-275
A-14-1 User Program Object Restrictions A-275
A-14-2 Counting User Program Objects A-279

A-15 Replacing CPU Units with Unit Version 1.02 or Earlier A-281
A-15-1 Uploading the Data from the CPU Unit A-281
A-15-2 Connecting the New CPU Unit A-284
A-15-3 Downloading the Data to the CPU Unit A-285

A-16 Possible Combination of Windows Versions and CPU Unit Versions A-288

A-17 Device Variables A-289
A-17-1 X Bus Unit Common Function Module A-289

A-18 Specifications for Individual Device Variables A-291
A-18-1 X Bus Unit Common Function Module A-291

A-19 Version Information for NX-series Controllers A-293
A-19-1 Relationship between Unit Versions of CPU Units and Sysmac Studio Versions A-293
A-19-2 Functions That Were Added or Changed for Each Unit Version A-296

A-20 Version Information for NJ-series Controllers A-298
A-20-1 Relationship between Unit Versions of CPU Units and Sysmac Studio Versions A-298
A-20-2 Functions That Were Added or Changed for Each Unit Version A-300
A-20-3 Performance Improvements for Unit Version Upgrades A-303

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 OR IN LARGE QUANTITIES WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Programmable Products

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

Disclaimers

Performance Data

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

Change in Specifications

Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may

be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions

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

Statement of security responsibilities for assumed use cases and against threats

OMRON SHALL NOT BE RESPONSIBLE AND/OR LIABLE FOR ANY LOSS, DAMAGE, OR EXPENSES DIRECTLY OR INDIRECTLY RESULTING FROM THE INFECTION OF OMRON PRODUCTS, ANY SOFTWARE INSTALLED THEREON OR ANY COMPUTER EQUIPMENT, COMPUTER PROGRAMS, NETWORKS, DATABASES OR OTHER PROPRIETARY MATERIAL CONNECTED THERETO BY DISTRIBUTED DENIAL OF SERVICE ATTACK, COMPUTER VIRUSES, OTHER TECHNOLOGICALLY HARMFUL MATERIAL AND/OR UNAUTHORIZED ACCESS.

It shall be the users sole responsibility to determine and use adequate measures and checkpoints to satisfy the users particular requirements for (i) antivirus protection, (ii) data input and output, (iii) maintaining a means for reconstruction of lost data, (iv) preventing Omron Products and/or software installed thereon from being infected with computer viruses and (v) protecting Omron Products from unauthorized access.

Safety Precautions

Refer to the following manuals for safety precautions.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User’s Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User’s Manual (Cat No. W500)*

Precautions for Safe Use

Refer to the following manuals for precautions for safe use.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User’s Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User’s Manual (Cat No. W500)*

Precautions for Correct Use

Refer to the following manuals for precautions for correct use.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User’s Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User’s Manual (Cat No. W500)*

Regulations and Standards

Refer to the following manuals for regulations and standards.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User's Manual (Cat No. W500)*

Software Licenses and Copyrights

Products that support the secure socket service or secure communication include the following third-party software. The license and copyright information associated with this software is available at http://www.fa.omron.co.jp/nj_info_e/.

- OpenSSL
  This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).
  Copyright (C) 1998-2019 The OpenSSL Project. All rights reserved.
  Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved.
  This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

For the products that support secure socket service, refer to *8-6-6 Secure Socket Services* on page 8-48.
For the products that support secure communication, refer to *8-6-5 Secure Communication Function* on page 8-48.

Versions

Hardware revisions and unit versions are used to manage the hardware and software in NJ/NX-series Units 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.

Refer to the following manuals for versions.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User's Manual (Cat No. W500)*

Unit Versions of CPU Units and Sysmac Studio Versions

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

Refer to *A-19 Version Information for NX-series Controllers* on page A-293 and *A-20 Version Information for NJ-series Controllers* on page A-298 for the relationship between the unit versions of the CPU Units and the Sysmac Studio versions, and for the functions that are supported by each unit version.

Related Manuals

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

Manual name	Cat. No.	Model numbers	Application	Description
NX-series CPU Unit Hardware User's Manual	W535	NX701-□□□□	Learning the basic specifications of the NX701 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX701 system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NX-series NX502 CPU Unit Hardware User's Manual	W629	NX502-□□□□	Learning the basic specifications of the NX502 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX502 system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NX-series NX102 CPU Unit Hardware User's Manual	W593	NX102-□□□□	Learning the basic specifications of the NX102 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX102 system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NX-series NX1P2 CPU Unit Hardware User's Manual	W578	NX1P2-□□□□	Learning the basic specifications of the NX1P2 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX1P2 system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NJ-series CPU Unit Hardware User's Manual	W500	NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning the basic specifications of the NJ-series CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NJ-series system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection

Manual name	Cat. No.	Model numbers	Application	Description
NJ/NX-series CPU Unit Software User's Manual	W501	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning how to program and set up an NJ/NX-series CPU Unit. Mainly software information is provided.	The following information is provided on a Controller built with an NJ/NX-series CPU Unit. • CPU Unit operation • CPU Unit features • Initial settings • Programming based on IEC 61131-3 language specifications
NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual	W579	NX1P2-□□□□	Learning about the details of functions only for an NX-series NX1P2 CPU Unit and an introduction of functions for an NJ/NX-series CPU Unit.	Of the functions for an NX1P2 CPU Unit, the following information is provided. • Built-in I/O • Serial Communications Option Boards • Analog I/O Option Boards An introduction of following functions for an NJ/NX-series CPU Unit is also provided. • Motion control functions • EtherNet/IP communications functions • EtherCAT communications functions
NJ/NX-series Instructions Reference Manual	W502	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning detailed specifications on the basic instructions of an NJ/NX-series CPU Unit.	The instructions in the instruction set (IEC 61131-3 specifications) are described.
NJ/NX-series CPU Unit Motion Control User's Manual	W507	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about motion control settings and programming concepts.	The settings and operation of the CPU Unit and programming concepts for motion control are described.
NJ/NX-series Motion Control Instructions Reference Manual	W508	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the specifications of the motion control instructions.	The motion control instructions are described.
NJ/NX-series CPU Unit Built-in EtherCAT® Port User's Manual	W505	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Using the built-in EtherCAT port on an NJ/NX-series CPU Unit.	Information on the built-in EtherCAT port is provided. This manual provides an introduction and provides information on the configuration, features, and setup.
NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual	W506	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Using the built-in EtherNet/IP port on an NJ/NX-series CPU Unit.	Information on the built-in EtherNet/IP port is provided. Information is provided on the basic setup, tag data links, and other features.
NJ/NX-series CPU Unit OPC UA User's Manual	W588	NX701-□□□□ NX502-□□□□ NX102-□□□□ NJ501-1□00	Using the OPC UA.	Describes the OPC UA.

Manual name	Cat. No.	Model numbers	Application	Description
NX-series CPU Unit FINS Function User's Manual	W596	NX701-□□20 NX502-□□□□ NX102-□□□□	Using the FINS function of an NX-series CPU Unit.	Describes the FINS function of an NX-series CPU Unit.
NJ/NX-series Database Connection CPU Units User's Manual	W527	NX701-□□20 NX502-□□□□ NX102-□□20 NJ501-□□20 NJ101-□□20	Using the database connection service with NJ/NX-series Controllers.	Describes the database connection service.
NJ-series SECS/GEM CPU Units User's Manual	W528	NJ501-1340	Using the GEM Services with NJ-series Controllers.	Provides information on the GEM Services.
NJ-series Robot Integrated CPU Unit User's Manual	O037	NJ501-R□□□	Using the NJ-series Robot Integrated CPU Unit.	Describes the settings and operation of the CPU Unit and programming concepts for OMRON robot control.
NJ-series NJ Robotics CPU Unit User's Manual	W539	NJ501-4□□□ NJ501-R□□□	Controlling robots with NJ-series CPU Units.	Describes the functionality to control robots.
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.
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.
NJ/NX-series Troubleshooting Manual	W503	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the errors that may be detected in an NJ/NX-series Controller.	Concepts on managing errors that may be detected in an NJ/NX-series Controller and information on individual errors are described.
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.
Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual	W595	SYSMAC-SE2□□□ SYSMAC-SE200D-64	Learning about the operating procedures and functions of the Sysmac Studio to configure Robot Integrated System using Robot Integrated CPU Unit.	Describes the operating procedures of the Sysmac Studio for Robot Integrated CPU Unit.
Sysmac Studio Robot Integrated System Building Function with IPC Application Controller Operation Manual	W621	SYSMAC-SE2□□□ SYSMAC-SE200D-64	Learning about the operating procedures and functions of the Sysmac Studio to configure Robot Integrated System using IPC Application Controller.	Describes the operating procedures of the Sysmac Studio for IPC Application Controller.
Sysmac Studio 3D Simulation Function Operation Manual	W618	SYSMAC-SE2□□□ SYSMAC-SA4□□ □-64	Learning about an outline of the 3D simulation function of the Sysmac Studio and how to use the function.	Describes an outline, execution procedures, and operating procedures for the 3D simulation function of the Sysmac Studio.

Manual name	Cat. No.	Model numbers	Application	Description
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 the 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 Data Reference Manual	W525	NX-□□□□□□	Referencing lists of the data that is required to configure systems with NX-series Units.	Lists of the power consumptions, weights, and other NX Unit data that is required to configure systems with NX-series Units are provided.
NX-series NX Units User's Manual	W521	NX-ID□□□□ NX-IA□□□□ NX-OC□□□□ NX-OD□□□□ NX-MD□□□□	Learning how to use NX Units.	Describes the hardware, setup methods, and functions of the NX Units. Manuals are available for the following Units. Digital I/O Units, Analog I/O Units, System Units, Position Interface Units, Communications Interface Units, Load Cell Input Unit, and IO-Link Master Units.
	W522	NX-AD□□□□ NX-DA□□□□		
	W592	NX-HAD□□□		
	W566	NX-TS□□□□ NX-HB□□□□		
	W523	NX-PD1□□□ NX-PF0□□□ NX-PC0□□□ NX-TBX01		
	W524	NX-EC0□□□ NX-ECS□□□ NX-PG0□□□		
	W540	NX-CIF□□□		
	W565	NX-RS□□□□		
	W567	NX-ILM□□□		
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 Sensors 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.

Manual name	Cat. No.	Model numbers	Application	Description
CJ-series Special Unit Manuals For NJ-series CPU Unit	W490	CJ1W-AD□□□ CJ1W-DA□□□ CJ1W-MAD42	Learning how to use CJ-series Units with an NJ-series CPU Unit.	The methods and precautions for using CJ-series Units with an NJ-series CPU Unit are described, including access methods and programming interfaces. Manuals are available for the following Units. Analog I/O Units, Insulated-type Analog I/O Units, Temperature Control Units, ID Sensor Units, High-speed Counter Units, Serial Communications Units, DeviceNet Units, EtherNet/IP Units and CompoNet Master Units.
	W491	CJ1W-TC□□□		
	W492	CJ1W-CT021		
	W498	CJ1W-PDC15 CJ1W-PH41U CJ1W-AD04U		
	W493	CJ1W-CRM21		
	W494	CJ1W-SCU□□		
	W495	CJ1W-EIP21		
	W497	CJ1W-DRM21		
	Z317	CJ1W-V680□□□		
NA-series Programmable Terminal Software User's Manual	V118	NA5-□W□□□□	Learning about NA-series PT pages and object functions.	Describes the pages and object functions of the NA-series Programmable Terminals.
NS-series Programmable Terminals Programming Manual	V073	NS15-□□□□□ NS12-□□□□□ NS10-□□□□□ NS8-□□□□□ NS5-□□□□□	Learning how to use the NS-series Programmable Terminals.	Describes the setup methods, functions, etc. of the NS-series Programmable Terminals.
CX-Designer User's Manual	V099	---	Learning to create screen data for NS-series Programmable Terminals.	Describes operating procedures for the CX-Designer.

Terminology

Term	Description
AT	One of the attributes of a variable. This attribute allows the user to specify what is assigned to a variable. An I/O port or an address in memory used for CJ-series Units can be specified.
CJ-series Unit	Any of the CJ-series Units that can be used with an NJ-series Controller.
memory used for CJ-series Units	One type of I/O memory that contains addresses to which variables can be assigned, for example, when accessing a CJ-series Unit or CJ-series network. It can be accessed only with variables with an AT attribute.
CPU Unit	The Unit that serves as the center of control for a Machine Automation Controller. The CPU Unit executes tasks, refreshes I/O for other Units and slaves, etc. The NJ/NX-series CPU Units include the NX701-□□□□, NX102-□□□□, NX1P2-□□□□, NJ501-□□□□.
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.
FB	An acronym for "function block."
FUN	An abbreviation for "function."
I/O port	A logical interface that is used by the CPU Unit to exchange data with an external device (slave or Unit).
I/O map settings	Settings that assign variables to I/O ports. Assignment information between I/O ports and variables.
I/O refreshing	Cyclic data exchange with external devices that is performed with predetermined memory addresses.
MC Test Run	A function to check motor operation and wiring from the Sysmac Studio.
NX bus	An internal bus of the NX-series Controller.
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.
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.)
PLC Function Module	One of the function modules. This function module executes the user program, sends commands to the Motion Control Function Module, and provides an interface to the USB and SD Memory Card.
POU	An acronym for "program organization unit". A POU is a unit in a program execution model that is defined in IEC 61131-3. A POU contains an algorithm and a local variable table and forms the basic unit used to build a user program. There are three types of POUs: programs, functions, and function blocks.
SDO communications	One type of EtherCAT communications in which service data objects (SDOs) are used to transmit information whenever required.
Sysmac Studio	A computer software application for setting, programming, debugging, and troubleshooting NJ/NX-series Controllers. It also provides operations for motion control and a Simulator.
X Bus	An internal bus of the NX-series Controller. A bus that can transfer data with a higher speed and larger capacity than the NX bus.

Term	Description
X Bus Function Module	One of the function modules. This function module controls X Bus Units mounted to the left side of the CPU Unit.
X Bus Units	Units that can connect to the X Bus.
upload	To transfer data from the Controller to the Sysmac Studio with the synchronization operation of the Sysmac Studio.
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.
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.
event log	A function that recognizes and records errors and other events.
inline ST	ST programming that is included within a ladder diagram.
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.
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.
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.
function module	One of the functional units of the software configuration of the CPU Unit.
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.
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.
union	One of the derivative data types. It allows you to handle the same data as different data types.
global variable	A variable that can be read or written from all POUs (programs, functions, and function blocks).
minor fault level Controller error	An error for which some of the control operations for one of the function modules in the NJ/NX-series Controller stop. An NJ/NX-series CPU Unit continues operation even after a minor fault level Controller error occurs.
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.
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.

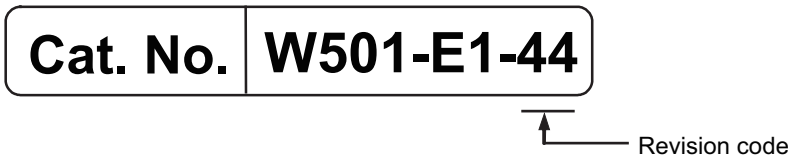
Term	Description
Controller	The range of devices that are directly controlled by the CPU Unit. In the NX-series System, the Controller includes the CPU Rack and EtherCAT slaves (including general-purpose slaves and Servo Drives). In the NJ-series System, the Controller includes the CPU Rack, Expansion Racks, and EtherCAT slaves (including general-purpose slaves and Servo Drives).
Controller error	Errors that are defined by the NJ/NX-series System. “Controller error” is a collective term for major fault level, partial fault level, minor fault level, and observation Controller events.
Controller event	One of the events in the NJ/NX-series System. Controller events are errors and information that are defined by the system for user notification. A Controller event occurs when the system detects a factor that is defined as a Controller event.
Controller information	Information that is defined by the NJ/NX-series System that is not an error. It represents an information Controller event.
Servo Drive/encoder input slave	Any of the EtherCAT slaves that is assigned to an axis. In the NJ/NX-series System, it would be a Servo Drive or Encoder Input Slave Unit.
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.
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 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.
system common processing	System processing that is performed by the CPU Unit to perform I/O refreshing and the user program execution within a task. Exclusive control of variables between tasks, data trace processing, and other processing is performed.
system service	Processing that is performed by the CPU Unit in unused time between task processing. The system service includes communications processing, SD Memory Card access processing, self-diagnosis processing, and other processing.
system-defined variable	A variable for which all attributes are defined by the system and cannot be changed by the user.
Initial Value	One of the attributes of a variable. The variable is set to the initial value in the following situations. • When power is turned ON • When the CPU Unit changes to RUN mode • When you specify to initialize the values when the user program is transferred • When a major fault level Controller error occurs
slave	A device that performs remote I/O for a master.
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.
slave and Unit configurations	A generic term for the EtherCAT configuration and Unit configuration.

Term	Description
absolute encoder home offset	This data is used to restore in the CPU Unit the actual position of a Servo Drive with an absolute encoder. The offset is the difference between the command position after homing and the absolute data that is read from the absolute encoder.
project unit version	A unit version to be set for the project. It is set for the project in the Select Device Area of the Project Properties Dialog Box on the Sysmac Studio.
major fault level Controller error	An error for which all NJ/NX-series Controller control operations stop. The CPU Unit immediately stops user program execution and turns OFF the loads for all slaves and Units (including remote I/O).
download	To transfer data from the Sysmac Studio to the Controller with the synchronization operation of the Sysmac Studio.
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.
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.
periodic task	A tasks for which user program execution and I/O refreshing are performed each period.
device	A general term for any Unit or slave that is refreshed by the I/O refreshing that is performed by the CPU Unit. Specifically, it refers to EtherCAT slaves, NX Units on the CPU Unit, built-in I/O, Option Boards, and CJ-series Units.
device output	An output for any Unit or slave that is refreshed by the I/O refreshing that is performed by the CPU Unit.
device variable	A variable that is used to access a specific device through an I/O port.
synchronization	A function that automatically compares the information in the NJ/NX-series Controller with the information in the Sysmac Studio, displays any differences and locations in a hierarchical form, and can be used to synchronize the information.
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.
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.
derivative data type	A data type that is defined by the user. Structures, unions, and enumerations are derivative data types.
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.
general-purpose slave	Any of the EtherCAT slaves that cannot be assigned to an axis.
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.
partial fault level Controller error	An error for which all of the control operations for one of the function modules in the NJ/NX-series Controller stop. An NJ/NX-series CPU Unit continues operation even after a partial fault level Controller error.
primary periodic task	The task with the highest priority.
program	Along with functions and function blocks, one of the three types of POUs. Programs are assigned to tasks to execute them.

Term	Description
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.
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.
Retain	One of the attributes of a variable. The values of variables with a Retain attribute are held at the following times. (Variables without a Retain attribute are set to their initial values.) • When power is turned ON after a power interruption • When the CPU Unit changes to RUN mode • When you specify to not initialize the values when the user program is transferred
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.
main memory	The memory inside the CPU Unit that is used by the CPU Unit to execute the OS and user program.
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.
user-defined event	One of the events in the NJ/NX-series System. These events are defined by the user. “User-defined events” is a generic term for user-defined errors and user-defined information.
user-defined variable	A variable for which all of the attributes are defined by the user and can be changed by the user.
user program	All of the programs in one project.
Unit	A device that mounts to the CPU Rack or an Expansion Rack.
Unit configuration	The configuration information for the Units that are set on the Sysmac Studio. This information tells what Unit models are connected to the CPU Unit and where they are connected.
literal	A constant expression that is used in a user program.
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.
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.

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

Revision code	Date	Revised content
14	April 2016	• Added information on the functions supported by unit version 1.11 of the CPU Units. • Corrected mistakes.
15	July 2016	• Added information on the functions supported by unit version 1.12 of the CPU Units.
16	October 2016	• Added information on the NX1P2-□□□□□□. • Added information on the functions supported by unit version 1.13 of the CPU Units.
17	January 2017	• Corrected mistakes.
18	April 2017	• Added information on the functions supported by unit version 1.14 of the CPU Units. • Corrected mistakes.
19	June 2017	• Corrected mistakes.
20	June 2017	• Added information on the functions supported by unit version 1.15 of the CPU Units.
21	October 2017	• Added information on the functions supported by unit version 1.16 of the CPU Units. • Corrected mistakes.
22	January 2018	• Added information on the functions supported by unit version 1.17 of the CPU Units.
23	April 2018	• Added information on the functions supported by unit version 1.18 of the CPU Units. • Corrected mistakes.
24	April 2018	• Added information on the NX102-□□□□. • Added information on the functions supported by unit version 1.30 of the CPU Units. • Made changes accompanying the transfer of explanation for event codes and errors to the*NJ/NX-series Troubleshooting Manual*.
25	July 2018	• Added information on the hardware revision. • Added information on the functions supported by unit version 1.19 of the NJ-series CPU Units. • Added information on the functions supported by unit version 1.31 of the NX102-□□□□. • Corrected mistakes.
26	January 2019	• Added information on the functions supported by unit version 1.20 of the NJ-series CPU Units. • Corrected mistakes.
27	April 2019	• Added information on the functions supported by unit version 1.32 of the NX102-□□□□. • Added information on the functions supported by unit version 1.21 of the NX1P2-□□□□□□, NJ501-1□00, NJ301-□□□□, and NJ101-□□00. • Corrected mistakes.
28	July 2019	• Added information on the functions supported by unit version 1.40 of the NX102-□□00, NX1P2-□□□□□□, NJ501-1□00, NJ301-□□□□, and NJ101-□□00. • Added information on the functions supported by unit version 1.21 of the NX701-□□□□, NJ501-4□00, NJ501-4□10, NJ501-1340, and NJ501-5300. • Corrected mistakes.

Revision code	Date	Revised content
29	October 2019	• Added information on the NX1P2-9B□□□□. • Corrected mistakes.
30	July 2020	• Added information on the functions supported by unit version 1.35 of the NX102-□□20. • Added information on the functions supported by unit version 1.23 of the NX701-□□□□, NJ501-1□20, NJ501-4320, and NJ101-□□20.
31	August 2020	• Added information on the NJ501-R□00.
32	October 2020	• Added information on the functions supported by unit version 1.42 of the NX102-□□00, NX1P2-□□□□□□, NJ501-1□00, NJ301-□□□□, and NJ101-□□00.
33	January 2021	• Added information on the functions supported by unit version 1.43 of the NX102-□□00 and NJ501-1□00. • Added information on the functions supported by unit version 1.36 of the NX102-□□20. • Added information on the functions supported by unit version 1.24 of the NX701-1□□□.
34	July 2021	• Added information on the HMC-SD292, HMC-SD492, and HMC-SD1A2 SD Memory Cards. • Added information on the functions supported by unit version 1.46 of the NX102-□□00, NX1P2-□□□□□, NJ501-1□00, NJ501-R□□□, NJ301-□□□□, and NJ101-□□00. • Added information on the functions supported by unit version 1.26 of the NJ501-□□20, NJ501-1340, NJ501-5300, NJ501-4□□□, NJ101-1□20, and NX701-□□□□. • Added information on the functions supported by unit version 1.37 of the NX102-□□20.
35	October 2021	• Added information related to the hardware revision A of the NX701-□□□□.
36	November 2021	• Added information related to the hardware revision D of the NJ-series CPU Unit.
37	April 2022	• Added information on the functions supported by unit version 1.48 of the NX102-□□□□ and NJ501-1□00. • Added information on the functions supported by unit version 1.47 of the NJ501-1□20, J501-4□□□, NJ501-1340, NJ501-5300, and NJ101-□□20. • Added information on the functions supported by unit version 1.28 of NX701-□□□□.
38	April 2022	• Added information to Terms and Conditions Agreement.
39	June 2022	• Added information on the hardware revision B of NX701-□□□□. • Added information on the hardware revision E of NJ501-□□□□. • Changed descriptions on the accuracy of the built-in clock. • Corrected mistakes.
40	October 2022	• Made changes accompanying the release of unit version 1.50 of the NJ-series, NX102, and NX1P2 CPU Units. • Corrected mistakes.
41	November 2022	• Added information on the functions supported by unit version 1.60 of the NJ-series, NX102, and NX1P2 CPU Units. • Added information on the functions supported by unit version 1.32 of the NX701 CPU Units.
42	January 2023	• Corrected mistakes.

Revision code	Date	Revised content
43	April 2023	• Added information on the NX502-1□□□.
44	July 2023	• Added information on the functions supported by unit version 1.63 of the NX502 CPU Unit. • Corrected mistakes.

1

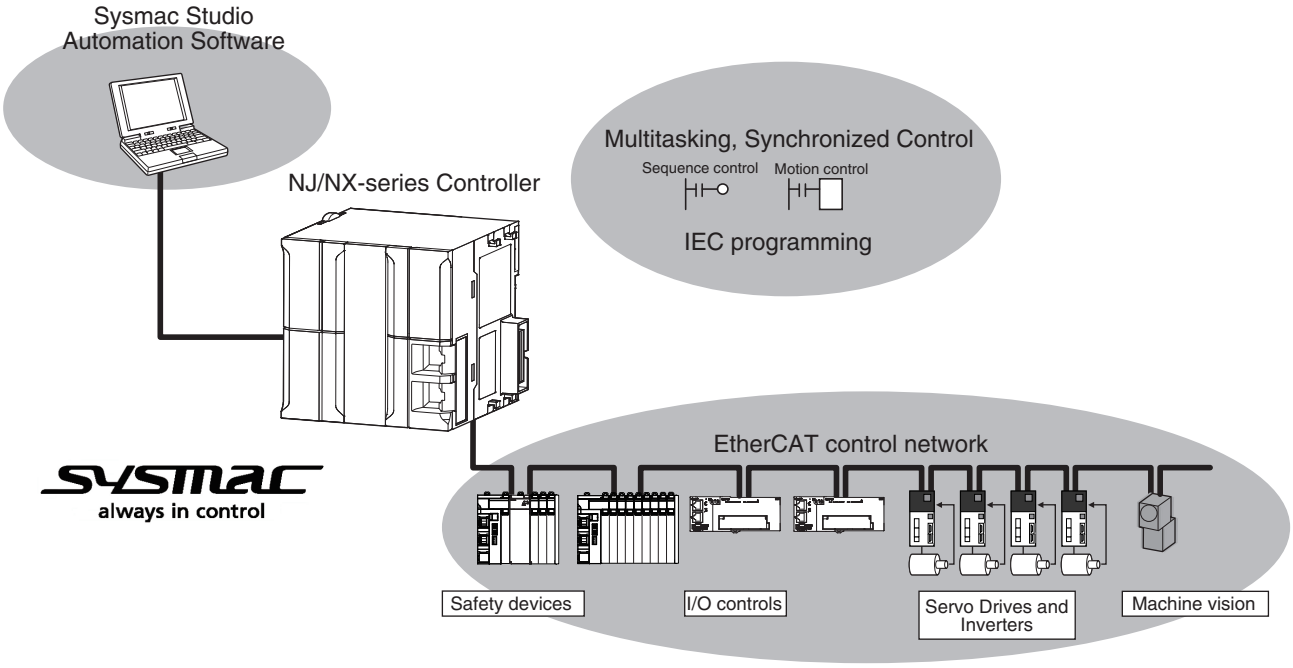
Introduction to NJ/NX-series Controllers

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

1-1 The NJ/NX-series Controllers 1-2
1-1-1 Features 1-2
1-1-2 Introduction to the System Configurations 1-5
1-2 Main Specifications 1-17
1-3 Overall Operating Procedure for the NJ/NX-series 1-23
1-3-1 Overall Procedure 1-23
1-3-2 Procedure Details 1-24

1-1 The NJ/NX-series Controllers

The SYSMAC NJ/NX-series Controllers are next-generation machine automation controllers that provide the functionality and high-speed performance that are required for machine control. They provide the safety, reliability, and maintainability that are required of industrial controllers.
The NJ/NX-series Controllers provide the functionality of previous OMRON PLCs, and they also provide the functionality that is required for motion control. Synchronized control of I/O devices on high-speed EtherCAT can be applied to safety devices, vision systems, motion equipment, discrete I/O, and more.
OMRON offers the new Sysmac Series of control devices designed with unified communications specifications and user interface specifications. The NJ/NX-series Machine Automation Controllers are part of the Sysmac Series. You can use them together with EtherCAT slaves, other Sysmac products, and the Sysmac Studio Automation Software to achieve optimum functionality and ease of operation.
With a system that is created from Sysmac products, you can connect components and commission the system through unified concepts and usability.



1-1-1 Features

Hardware Features

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

  All CPU Units provide an EtherCAT master port for EtherCAT communications.
  EtherCAT is an advanced industrial network system that achieves faster, more-efficient communications. It is based on Ethernet. Each node achieves a short fixed communications cycle time by transmitting Ethernet frames at high speed.

The standard-feature EtherCAT control network allows you to connect all of the devices required for machine control (e.g., I/O systems, Servo Drives, Inverters, and machine vision) to the same network.

● Support for EtherCAT Slave Terminals

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

● Achieving a Safety Subsystem

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

Version Information

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

● NX Units (Only for the NX502, NX102, and NX1P2)

The NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units allow you to mount NX-series Digital I/O Units, Analog I/O Units and other Units to the CPU Unit, in addition to EtherCAT network slaves.

Precautions for Correct Use

- You cannot mount NX-series Safety Control Units on the NX1P2 CPU Unit and use them. Use NX-series Safety Control Units as a subsystem on EtherCAT.

● X Bus Units (Only for the NX502)

The NX502 CPU Units allow you to mount EtherNet/IP Units and other X Bus Units to the CPU Unit, in addition to EtherCAT network slaves and NX Units.

● CJ-series Units (Only for the NJ□01)

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

● Standard-feature EtherNet/IP Communications Port

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

● Standard-feature USB Port (Only for the NJ□01)

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

Precautions for Correct Use

The NX701 CPU Units with hardware revision A or later, NX502, NX102, and NX1P2 CPU Units do not provide a USB port.

● Standard-feature SD Memory Card Slot

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

● Highly Reliable Hardware

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

● Parallel Execution of Tasks with a Multi-core Processor (NX701, NX502, NX102, and NX1P2)

The NX701-□□□□ CPU Unit has a multi-core processor that can execute more than one task in parallel. This enables high-speed control of even large-scale devices including an improvement of performance for large capacity data communications.
The NX502-□□□□ CPU Unit, NX102-□□□□ CPU Unit, and NX1P2-□□□□ CPU Unit have a multi-core processor that can execute the tasks, tag data link service, and system services in parallel. This maintains communications performance with the Sysmac Studio, an HMI, or other devices.

Software Features

● Integrated Sequence Control and Motion Control

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

● Multitasking

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

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

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

● Programming with Variables to Eliminate Worrying about the Memory Map

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

● A Wealth of Security Features

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

● Complete Controller Monitoring

The CPU Unit monitors events in all parts of the Controller, including mounted Units and EtherCAT slaves.

Troubleshooting information for errors is displayed on the Sysmac Studio or on an NS-series PT. Events are also recorded in logs.

● Sysmac Studio Automation Software

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

● A Wealth of Simulation Features

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

1-1-2 Introduction to the System Configurations

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

Introduction to the System Configurations of the NX701 CPU Units

The NX701 CPU Unit supports the following system configurations.

● Basic System Configurations

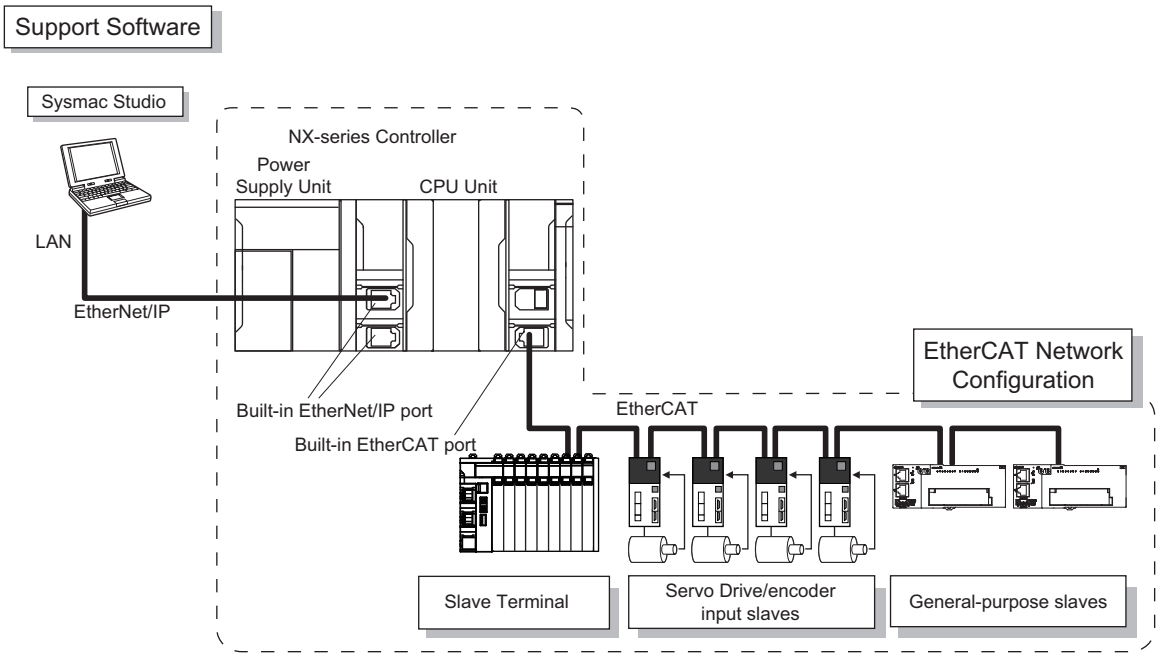
The NX701 basic configurations include the EtherCAT network configuration and the Support Software.

- 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

  You can connect the Support Software through an Ethernet cable that is connected to the built-in EtherNet/IP port.

  Refer to *10-2 Connection with Sysmac Studio* on page 10-9 for details on the connection configuration of the Support Software.



Precautions for Correct Use

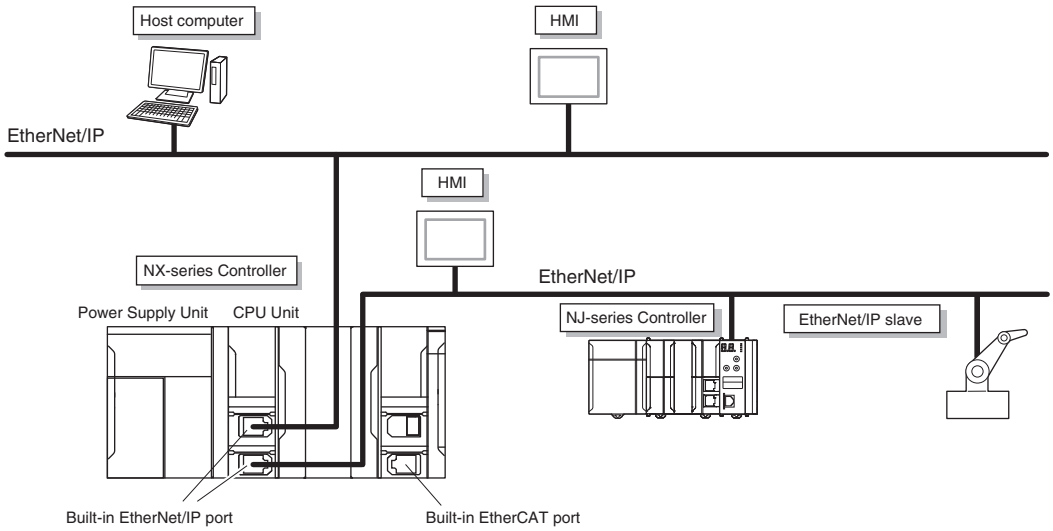
- NX Units should be connected to Slave Terminals. The NX bus connector of the CPU Unit is provided for future expansion so that it cannot be used to connect any NX Unit.
- The NX701 CPU Unit of hardware revision A or later does not provide a USB port.

Additional Information

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

● Network Configurations

- Host computers, HMIs, and other NJ/NX-series Controllers are connected to the built-in EtherNet/IP port.
- An NX701 CPU Unit has two built-in EtherNet/IP ports.



Refer to *Section 10 Communications Setup* on page 10-1 for details on the network configuration.

● Support Software

You can use the following Support Software to set up, monitor, and debug an NX701 CPU Unit.

- Sysmac Studio

  The Sysmac Studio is the main Support Software that you use for an NX701 CPU Unit. 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
Network Configurator	The Network Configurator is used for tag data links on EtherNet/IP ports.*1

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

Precautions for Correct Use

For the NX701 CPU Unit with unit version 1.29 or later, use Sysmac Studio on Windows 10 version 1803 or higher OS on the computer. Sysmac Studio on Windows 10 version 1709 or lower cannot connect online to the Controller.

Introduction to the System Configurations of the NX502 CPU Units

The NX502 CPU Unit supports the following system configurations.

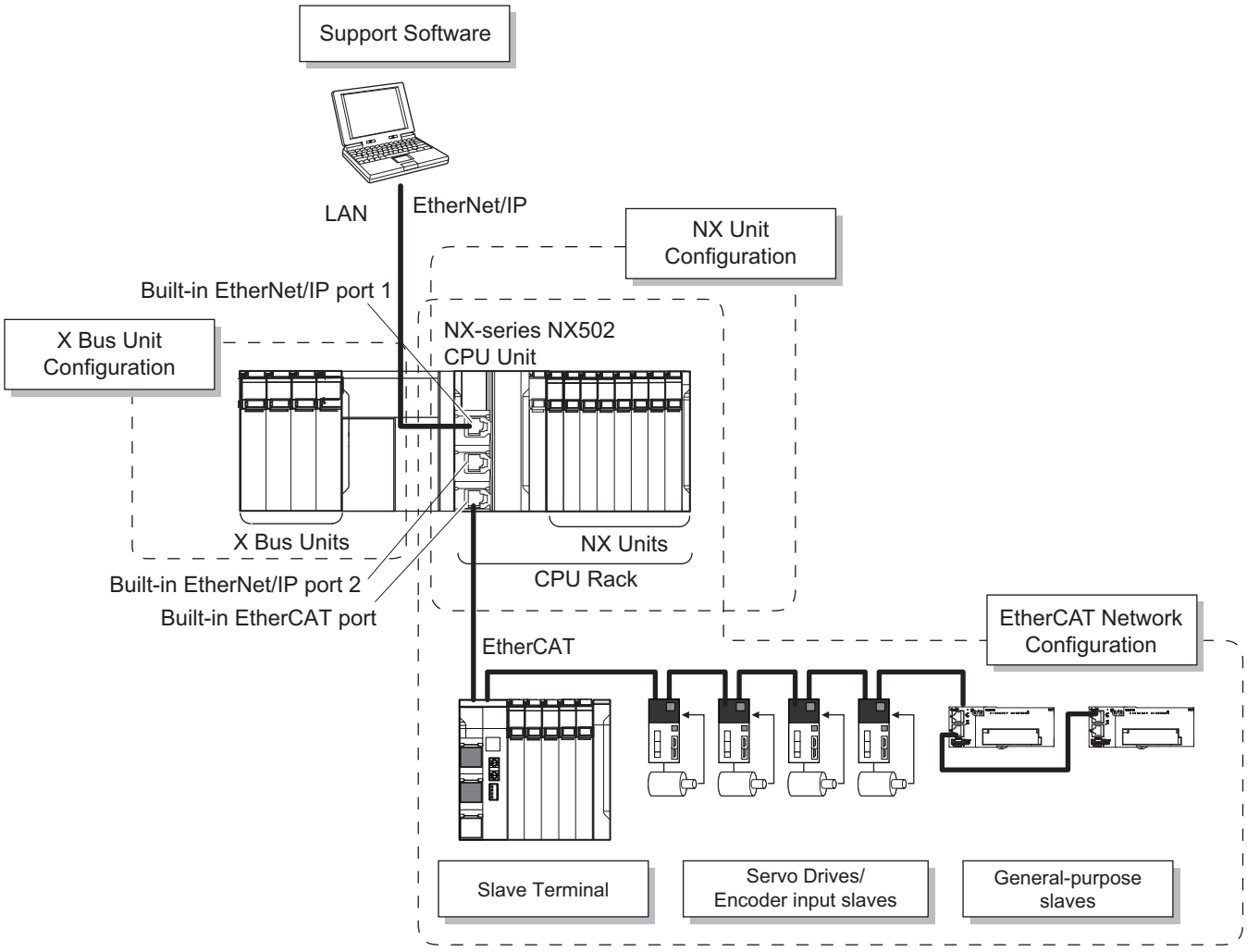
● Basic System Configurations

The NX502 basic configurations include the EtherCAT network configuration, NX Unit configuration, X Bus Unit configuration, and the Support Software.

- EtherCAT Network Configuration

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

- NX Unit Configuration
  You can mount NX-series Digital I/O Units, Analog I/O Units and other Units.
- X Bus Unit Configuration
  You can mount EtherNet/IP Units and other X Bus Units.
- Support Software
  You can connect the Support Software through an Ethernet cable that is connected to the built-in EtherNet/IP port.
  Refer to *10-2 Connection with Sysmac Studio* on page 10-9 for details on the connection configuration of the Support Software.



Precautions for Correct Use

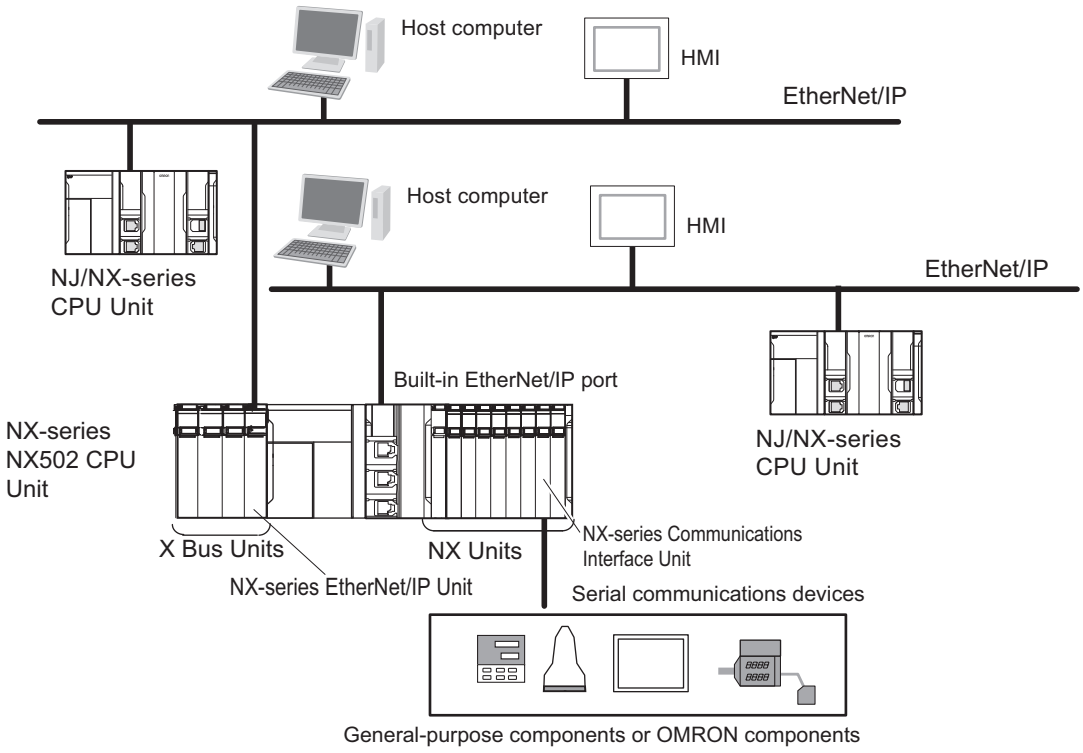
An NX502 CPU Unit does not provide a USB port.

Additional Information

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

● Network Configurations

- Host computers, HMIs, and other NJ/NX-series Controllers are connected to the built-in EtherNet/IP port or the EtherNet/IP port on the NX-series EtherNet/IP Unit.



Refer to *Section 10 Communications Setup* on page 10-1 for details on the network configuration.

● Support Software

You can use the following Support Software to set up, monitor, and debug an NX502 CPU Unit.

- Sysmac Studio

  The Sysmac Studio is the main Support Software that you use for an NX502 CPU Unit. 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
Network Configurator	The Network Configurator is used for tag data links on the built-in EtherNet/IP port and EtherNet/IP port of the NX-series EtherNet/IP Unit. *1

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

Precautions for Correct Use

For the NX502 CPU Unit, use Sysmac Studio on Windows 10 version 1803 or higher OS on the computer. Sysmac Studio on Windows 10 version 1709 or lower cannot connect online to the Controller.

Introduction to the System Configurations of the NX102 CPU Units

The NX102 CPU Unit supports the following system configurations.

● Basic System Configurations

The NX102 basic configurations include the EtherCAT network configuration, NX Unit configuration, and the Support Software.

- EtherCAT Network Configuration

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

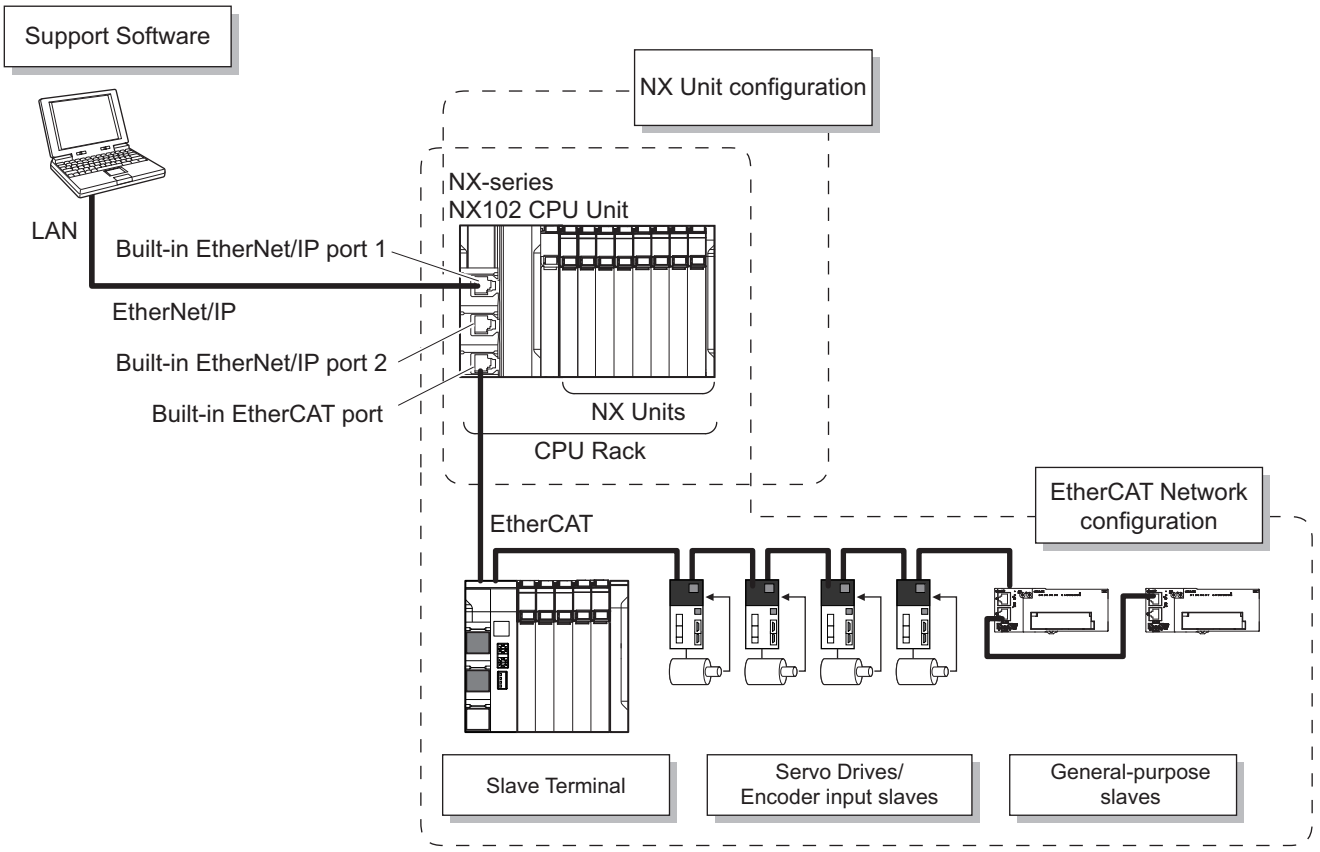
- NX Unit Configuration

  In addition to the EtherCAT network, you can mount NX-series Digital I/O Units, Analog I/O Units and other Units.

- Support Software

  You can connect the Support Software through an Ethernet cable that is connected to the built-in EtherNet/IP port.

  Refer to *10-2 Connection with Sysmac Studio* on page 10-9 for details on the connection configuration of the Support Software.



Precautions for Correct Use

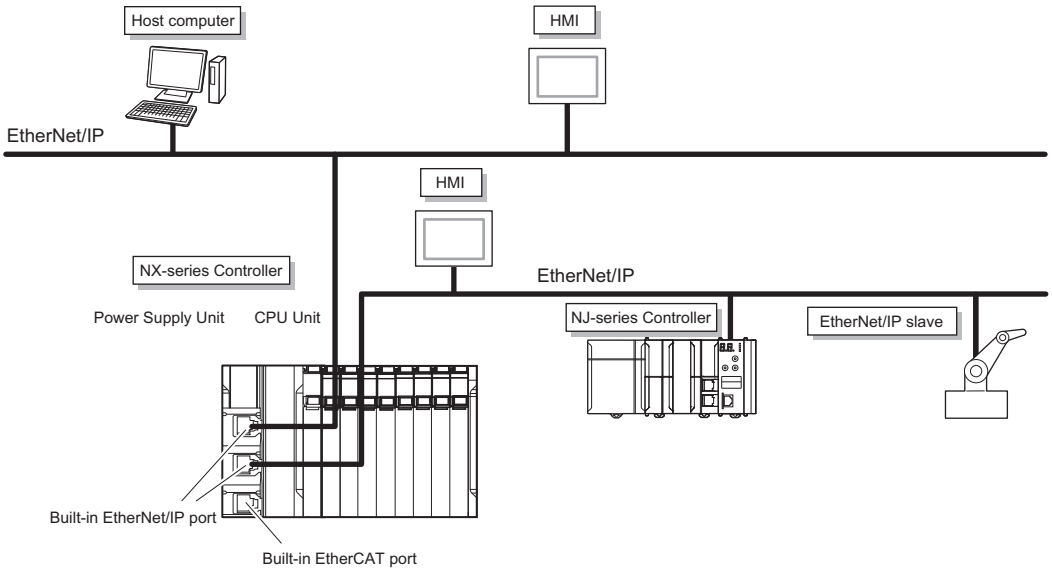
An NX102 CPU Unit does not provide a USB port.

Additional Information

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

● Network Configurations

- Host computers, HMIs, and other NJ/NX-series Controllers are connected to the built-in EtherNet/IP port.
- An NX102 CPU Unit has two built-in EtherNet/IP ports.



Refer to *Section 10 Communications Setup* on page 10-1 for details on the network configuration.

● Support Software

You can use the following Support Software to set up, monitor, and debug an NX102 CPU Unit.

- Sysmac Studio

  The Sysmac Studio is the main Support Software that you use for an NX102 CPU Unit. On it, you can set up the Controller configurations, parameters, and programs, and you can debug and simulate operation.

- Other Support Soft

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

Configuration software	Application
Network Configurator	The Network Configurator is used for tag data links on EtherNet/IP ports.*1

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

Precautions for Correct Use

For the NX102 CPU Unit with unit version 1.49 or later, use Sysmac Studio on Windows 10 version 1803 or higher OS on the computer. Sysmac Studio on Windows 10 version 1709 or lower cannot connect online to the Controller.

Introduction to the System Configurations of the NX1P2 CPU Units

The NX1P2 CPU Unit supports the following system configurations.

● Basic System Configurations

The NX1P2 basic configurations include the EtherCAT network configuration, NX Unit configuration, and the Support Software.

- EtherCAT network configuration

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

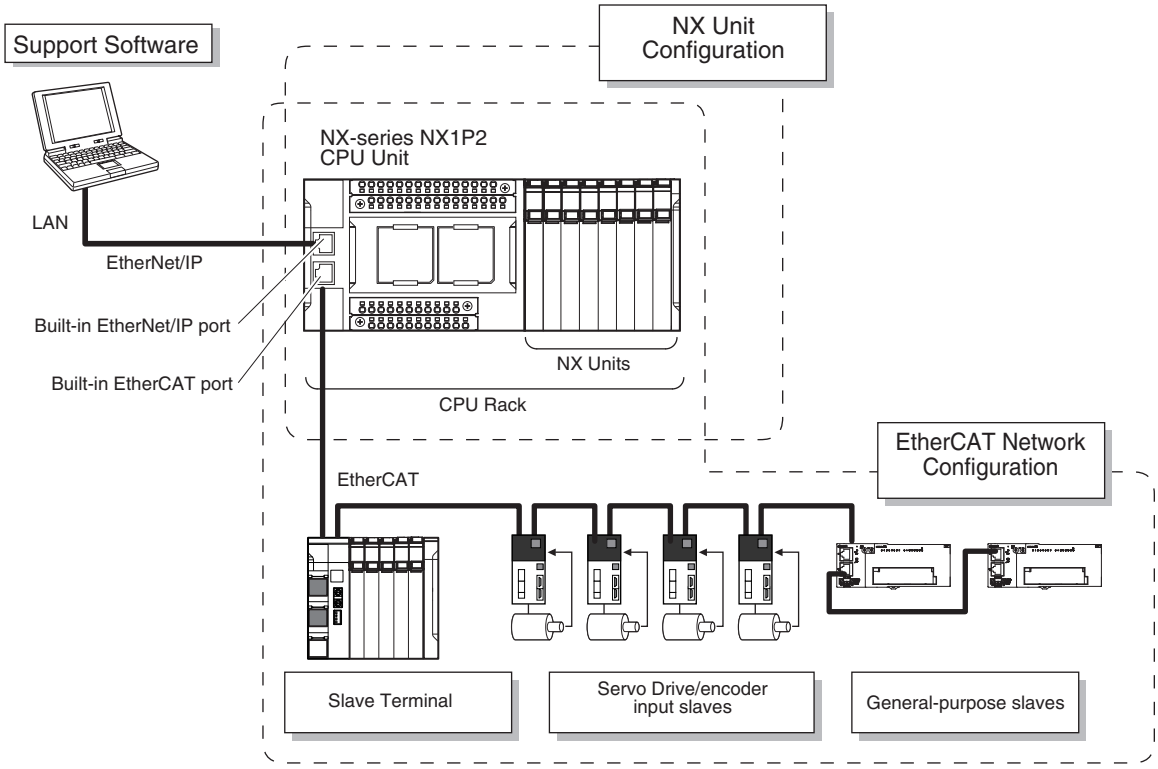
- NX Unit configuration

  In addition to the EtherCAT network, you can mount NX-series Digital I/O Units, Analog I/O Units and other Units.

- Support Software

  You can connect the Support Software through an Ethernet cable that is connected to the built-in EtherNet/IP port.

  Refer to *10-2 Connection with Sysmac Studio* on page 10-9 for details on the connection configuration of the Support Software.



Precautions for Correct Use

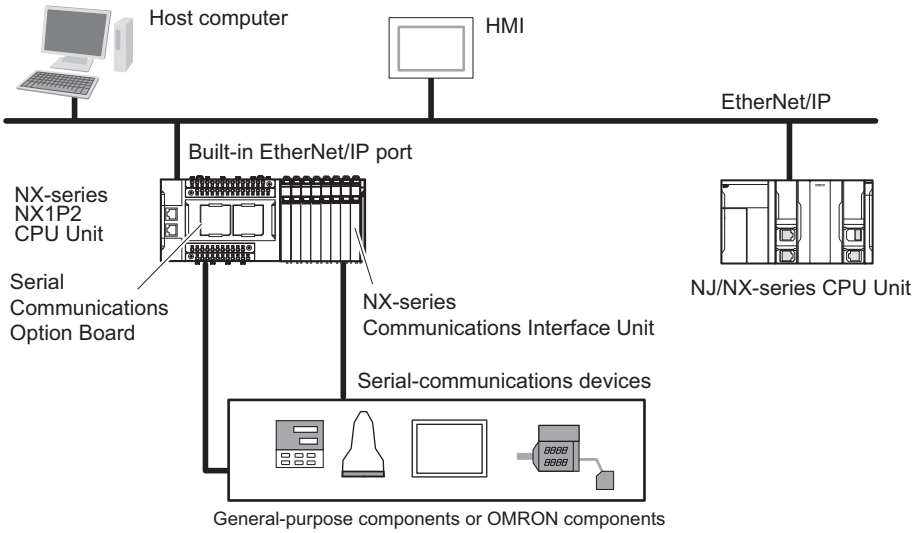
An NX1P2 CPU Unit does not provide a USB port.

Additional Information

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

● Network Configurations

- Host computers, HMIs, and other NJ/NX-series Controllers are connected to the built-in EtherNet/IP port.



Refer to *Section 10 Communications Setup* on page 10-1 for details on the network configuration.

● Support Software

You can use the following Support Software to set up, monitor, and debug an NX1P2 CPU Unit.

- Sysmac Studio

  The Sysmac Studio is the main Support Software that you use for an NX1P2 CPU Unit. 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
Network Configurator	The Network Configurator is used for tag data links on EtherNet/IP ports.*1

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

Precautions for Correct Use

For the NX1P2 CPU Unit with unit version 1.49 or later, use Sysmac Studio on Windows 10 version 1803 or higher OS on the computer. Sysmac Studio on Windows 10 version 1709 or lower cannot connect online to the Controller.

Introduction to the System Configurations of the NJ-series Controllers

The NJ-series supports the following system configurations.

● Basic System Configurations

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

- EtherCAT network configuration

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

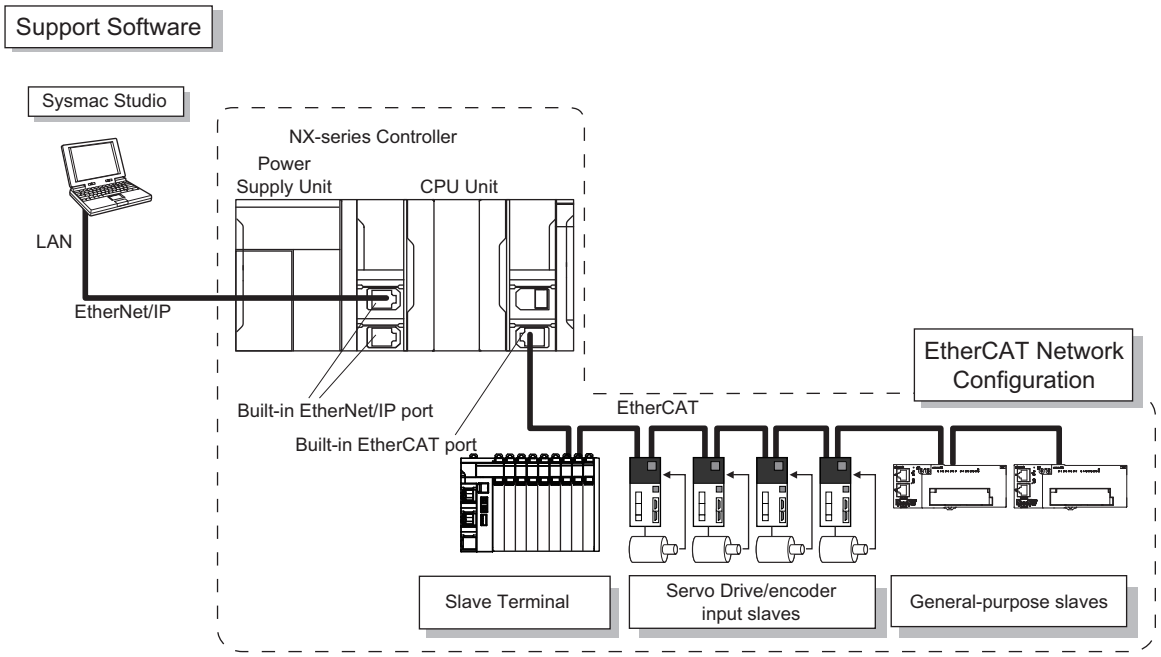
- CJ-series Unit configuration

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

- Support Software

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

  Refer to *10-2 Connection with Sysmac Studio* on page 10-9 for details on the connection configuration of the Support Software.

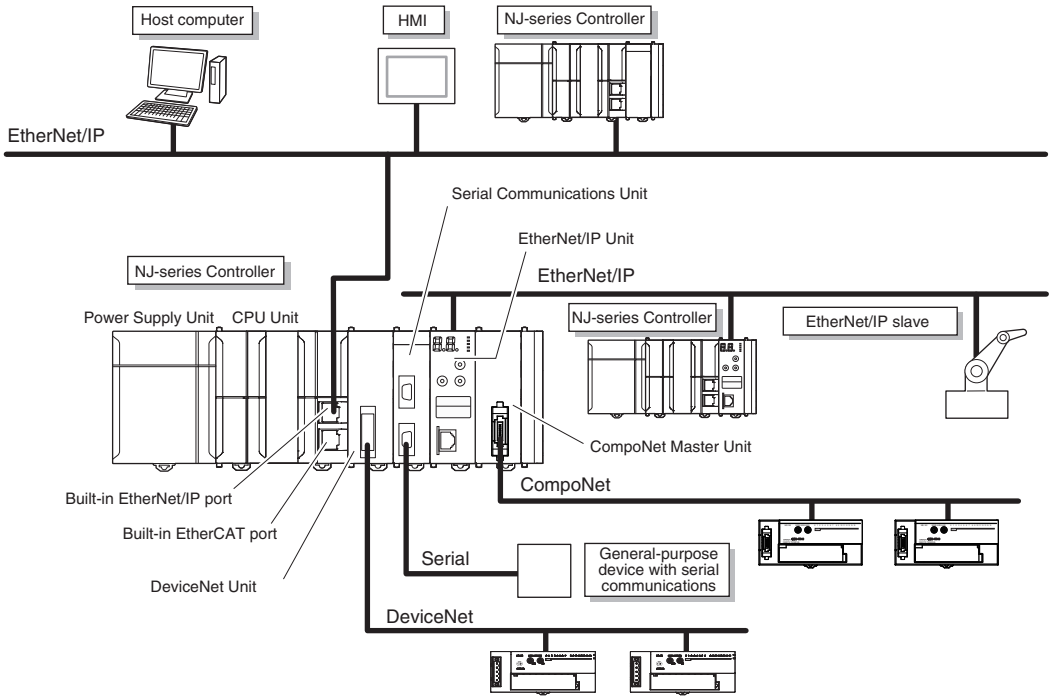


Additional Information

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

● Network Configurations

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



Refer to *Section 10 Communications Setup* on page 10-1 for details on the network configuration.

● Support Software

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

- Sysmac Studio

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

- Other Support Software

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

Configuration software	Application
Network Configurator	The Network Configurator is used for tag data links on EtherNet/IP ports or Units. *1
CX-Integrator	The CX-Integrator is used for remote I/O communications with a DeviceNet Unit or CompoNet Master Unit.
CX-Protocol	The CX-Protocol is used for protocol macros with Serial Communications Units.
CX-Designer	The CX-Designer is used to create screens for NS-series PTs.

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

Precautions for Correct Use

For the NJ-series CPU Unit with unit version 1.49 or later, use Sysmac Studio on Windows 10 version 1803 or higher OS on the computer. Sysmac Studio on Windows 10 version 1709 or lower cannot connect online to the Controller.

1-2 Main Specifications

This section gives the main specifications of the NJ/NX-series Controllers. Refer to *A-1 Specifications* on page A-4 for general specifications, performance specifications, and function specifications.

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Programming	Program capacity[*1]	Size		80 MB	80 MB	5 MB	1.5 MB[*2]	20 MB	5 MB	3 MB
		Quantity	Number of POU definitions	6,000		3,000	450	3,000	750	450
			Number of POU instances	48,000		9,000	1,800	9,000 (*)	3,000 (*)	1,800
	Memory capacity for variables	Retain attributes[*3]	Size	4 MB	4MB[*4]	1.5MB[*4]	32 KB[*4]	2 MB	0.5 MB	
			Number of variables	40,000		10,000	5,000	10,000	5,000 (*)	
		No Retain attributes[*5]	Size	256 MB	256MB[*4]	32MB[*4]	2MB[*4]	4 MB	2 MB	
			Number of variables	360,000		90,000	90,000	180,000 (*)	90,000 (*)	22,500
	Data types	Number of data types		8,000		1,000	1,000	2,000	1,000	
	Memory for CJ-series Units (Can be specified with AT specifications for variables.)	CIO Area		---	6,144 words (CIO 0 to CIO 6143)[*6]			6,144 words (CIO 0 to CIO 6143)		
		Work Area		---	512 words (W0 to W511)[*6]			512 words (W0 to W511)		
		Holding Area		---	1,536 words (H0 to H1535)[*7]			1,536 words (H0 to H1535)		

Item			NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
		DM Area	---	32,768 words (D0 to D32767)*7		16,000 words (D0 to D15999)*7	32,768 words (D0 to D32767)		
		EM Area	---	32,768 words × 25 banks (E0_00000 to E18_32767) *7	32,768 words × 25 banks (E0_00000 to E18_32767) *7*8	---	32,768 words × 25 banks (E0_00000 to E18_32767)	32,768 words × 4 banks (E0_00000 to E3_32767)	
Motion control			Refer to *Performance Specifications of Motion Control for Each Type of CPU Units* on page A-13.						
Peripheral (USB) port	Supported services		---*9	---	---	---	Sysmac Studio connection		
	Physical layer		---*9	---	---	---	USB 2.0-compliant B-type connector		
	Transmission distance		---*9	---	---	---	5 m max.		
Built-in EtherNet/IP port	Number of ports		2	2	2	1			
	Physical layer		10BASE-T/100BASE-TX/1000BASE-T		10Base-T or 100Base-TX				
	Frame length		1,514 bytes max.						
	Media access method		CSMA/CD						
	Modulation		Baseband						
	Topology		Star						
	Baud rate		1 Gbps (1000BASE-T)		100 Mbps (100BASE-TX)				
	Transmission media		Shielded twisted-pair (STP) cable, Category 5, 5e, or higher						
	Maximum transmission distance between Ethernet switch and node		100 m						
	Maximum number of cascade connections		There is no limitation when an Ethernet switch is used.						
	CIP service: Tag data links (cyclic communications)	Maximum number of connections	256 per port 512 total	64 per port 128 total	32 per port 64 total	32			
		Packet interval*10	Can be set for each connection. 0.5 to 10,000 ms in 0.5-ms increments	Can be set for each connection. 1 to 10,000 ms in 1-ms increments	Can be set for each connection. 1 to 10,000 ms in 1-ms increments	Can be set for each connection. 2 to 10,000 ms in 1-ms increments	Can be set for each connection. 1 to 10,000 ms in 1-ms increments (*)		

Item			NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
		Permissible communications band	40,000 pps*11 (including heartbeat)	20,000 pps*11 (including heartbeat)	12,000 pps*11 (including heartbeat, CIP Safety routing)	3,000 pps*11 (including heartbeat) (*)			
		Maximum number of tag sets	256 per port 512 total	64 per port 128 total	32 per port 40 total*12	32			
		Tag types	Network variables	Network variables, CIO, Work, Holding, DM, and EM Areas		Network variables, CIO, Work, Holding, and DM Areas	Network variables, CIO, Work, Holding, DM, and EM Areas		
		Number of tags per connection (i.e., per tag set)	8 (7 tags if Controller status is included in the tag set.)	64 (63 tags if Controller status is included in the tag set.)	8 (7 tags if Controller status is included in the tag set.)				
		Maximum number of tags	256 per port 512 total	256 per port 512 total	256 per port 512 total	256			
		Maximum link data size per node (total size for all tags)	369,664 bytes per port 739,328 bytes total	92,416 bytes per port 184,832 bytes total	19,200 bytes per port 38,400 bytes total	19,200 bytes			
		Maximum data size per connection	1,444 bytes		600 bytes				
		Maximum number of registrable tag sets	256 per port 512 total (1 connection = 1 tag set)	64 per port 128 total (1 connection = 1 tag set)	32 per port 40 total*12 (1 connection = 1 tag set)	32 (1 connection = 1 tag set)			
		Maximum tag set size	1,444 bytes (Two bytes are used if Controller status is included in the tag set.)		600 bytes (Two bytes are used if Controller status is included in the tag set.)				
		Multi-cast packet filter*13	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)						

Item			NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	Duplex mode		Auto						
	Topology		Line, daisy chain, and branching	Line, daisy chain, branching, and ring*14					
	Transmission media		Twisted-pair cable of category 5 or higher (double-shielded straight cable with aluminum tape and braiding)						
	Maximum transmission distance between nodes		100 m						
	Maximum number of slaves		512	256	64	16*15	192		64
Serial communications	Communications method		---			Half duplex (when connected to the Serial Communications Option Board)	---		
	Synchronization method		---			Start-stop synchronization (when connected to the Serial Communications Option Board)	---		
	Baud rate		---			1.2/2.4/4.8/9.6/19.2/38.4/57.6/115.2 kbps (when connected to the Serial Communications Option Board)	---		
Unit configuration	Maximum number of connectable Units	Maximum number of CJ Units per CPU Rack or Expansion Rack	---				10		
		Maximum number of NX Units per CPU Rack or Expansion Rack	---	63	32	8	---		

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
		Maximum number of CJ Units for entire controller		---				40		
		Maximum number of NX Units for entire controller		4,096		432	24	4,096		400
		Maximum number of X Bus Units for entire controller		---	4	---				
	Maximum number of Expansion Racks			0				3		
	I/O capacity	Maximum number of I/O points on CJ-series Units		---				2,560		
	Power Supply Unit for CPU Rack and Expansion Racks	Model		NX-PA9001 NX-PD7001	A non-isolated power supply for DC input is built into the CPU Unit.			NJ-P□3001		
		Power OFF detection time	AC power supply	30 to 45 ms	---			30 to 45 ms		
			DC power supply	5 to 20 ms	2 to 4 ms	2 to 8 ms		22 to 25 ms		
	Maximum NX Bus I/O data size			---	Inputs: 8,192 bytes Outputs: 8,192 bytes			---		

*1. Execution objects and variable tables (including variable names)
*2. The size of program capacity is 1.0 MB for an NX1P2-9B□□□□□ CPU Unit.
*3. Does not include Holding, DM, and EM Area memory for CJ-series Units.
*4. Memory for CJ-series Units is included.
*5. Does not include CIO and Work Area memory for CJ-series Units.
*6. Variables without a Retain attribute are used. The value can be set for each word.
*7. Variables with a Retain attribute are used. The value can be set for each word.
*8. For the NX102 CPU Unit, creating all banks simultaneously at the maximum number of words is not possible, because the capacity of the retain variable memory is limited to 1.5 MB.
*9. An NX701 CPU Unit without hardware revision provides a peripheral (USB) port of the following specifications.
- Supported service: Sysmac Studio connection
- Physical layer: USB 2.0-compliant B-type connector
- Transmission distance: 5 m max.

*10. Data will be refreshed at the set interval, regardless of the number of nodes.
*11. “pps” means packets per second, i.e., the number of communications packets that can be sent or received in one second.
*12. If more than 40 tag sets are set in total, a Number of Tag Sets for Tag Data Links Exceeded (840E0000 hex) event will occur.

*13. 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.
*14. A ring topology can be used for project unit version 1.40 or later.
*15. The maximum number of slaves is 8 for an NX1P2-9B□□□□□ CPU Unit.

Note Items that are marked with asterisks in the table are improvements that were made during version upgrades. Refer to *A-19 Version Information for NX-series Controllers* on page A-293 and *A-20 Version Information for NJ-series Controllers* on page A-298 for information on version upgrades.

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

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

1-3-1 Overall Procedure

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

Step 1. Software Design
Design the overall system configuration, task configuration, programs, and variables.
Step 1-1 Designing I/O and Processing
Step 1-2 Designing Tasks
Step 1-3 Designing Programs

Step 2. Software Setups and Programming
Create the system configurations that you designed in step 1 on the Support Software and assign the variables. Create the tasks and programs, and debug them, e.g., with simulations.
Step 2-1 Slave and Unit Configurations
Step 2-2 Controller Setup
Step 2-3 Programming
Step 2-4 Offline Debugging

Step 3. Mounting and Setting Hardware
Mount the Units and make the required hardware settings.

Step 4. Wiring
Connect the network cables and wire the I/O.

Step 5. Checking Operation and Starting Operation on the Actual System
Connect the Support Software to the physical system and download the project. Check operation on the physical system and then start actual system operation.

1-3-2 Procedure Details

Step 1. Software Design		
Step	**Description**	**Reference**
STEP1-1 Designing I/O and Processing	• External I/O devices and Unit configuration • Refresh periods for external devices • Program contents	*NX-series CPU Unit Hardware User's Manual (Cat. No. W535)* *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)* *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)* *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)* *NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)*
STEP1-2 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* on page 4-7
STEP1-3 Designing Programs		
POU (Program Organization Unit) Design	• Programs • Functions and function blocks • Determining the algorithm languages	*Section 6 Programming* on page 6-1
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* on page 6-28

Step 2. Software Setup 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.

STEP2-1 **Slave and Unit configurations**			
1) Creating the Slave and Unit Configurations	1. Creating the slave configuration and Unit configuration either offline or online. (For online configuration, make the online connection that is described in step 5.) 2. Setting up any Slave Terminals that are used.	EtherCAT Slave Setting Editor Unit Editor	*3-2-1 Creating the EtherCAT Slave Configuration* on page 3-5 *3-2-2 Creating the Unit Configuration* on page 3-6 *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* on page 3-8

(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/NX Units* on page 3-19

STEP2-2 Controller Setup	Setting the following parameters from the Sysmac Studio			*Section 4 Controller Setup* on page 4-1
		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* on page 4-4
		For NX502 CPU Units • Setting the clock data with clock function when a Battery is mounted • Configuration and Setup for NX Units • Initial Settings for NX Units • Configuration and Setup for X Bus Units • Initial Settings for X Bus Units • Memory settings for CJ-series Units For NX1P2 CPU Units • Setting the clock data with clock function when a Battery is mounted • Configuration and Setup for NX Units • Initial Settings for NX Units • Initial settings for built-in I/O • Initial settings for Option Boards • Memory settings for CJ-series Units	Each setting is given below. • **Controller** - **Controller Clock** • **Configurations and Setup** - **CPU/Expansion Racks** - **CPU Rack: NX Bus** • **Configurations and Setup** - **CPU/Expansion Racks** - **CPU Rack: X Bus** • **Configurations and Setup** - **Controller Setup** - **Built-in I/O Settings** • **Configurations and Setup** - **Controller Setup** - **Option Board Settings** • **Configurations and Setup** - **Controller Setup** - **Memory Settings for CJ-series Units**	*8-1-2 Clock* on page 8-3 *4-2-4 Configuration and Setup for NX Units* on page 4-13 *4-3 Initial Settings for NX Units* on page 4-21 *4-4 Initial Settings for X Bus Units* on page 4-25 *4-2-5 Configuration and Setup for X Bus Units* on page 4-16 *NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual (Cat. No. W579)*
		For NJ-series CPU Units Initial Settings for Special Units	**Configurations and Setup** - **CPU/Expansion Racks**	*4-5 Initial Settings for Special Units* on page 4-26
		To use motion control Setting the initial settings for the Motion Control Function Module	**Configurations and Setup** - **Motion Control Setup**	*4-6 Initial Settings for the Motion Control Function Module* on page 4-28
		Setting the initial values for the EtherCAT Function Module	**Configurations and Setup** - **EtherCAT**	*4-7 Initial Settings for the EtherCAT Master Function Module* on page 4-30
		Setting the initial values for the EtherNet/IP Function Module	**Configurations and Setup** - **Controller Setup** - **Built-in EtherNet/IP Port Settings**	*4-8 Initial Settings for the EtherNet/IP Function Module* on page 4-31

STEP2-3 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* on page 6-28
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* on page 6-1 *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* and *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*
3) Setting the Tasks	Making task settings	Task Settings	*4-2-3 Task Settings* on page 4-7
STEP2-4 Offline Debugging	Checking the algorithms and task execution times on the Simulator (virtual controller)		*Section 7 Checking Operation and Actual Operation* on page 7-1

Step 3. Mounting and Setting Hardware		
Step	**Description**	**Reference**
1. Mounting	• Connecting adjacent Units • Mounting to DIN Track	*NX-series CPU Unit Hardware User's Manual (Cat. No. W535)* *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)* *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)* *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)* *NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)*

2. Setting Hardware	• Setting the node addresses of the EtherCAT slaves • Setting unit numbers on the rotary switches on the front of the Special Units	Operation manuals for the EtherCAT slaves and Special Units

Step 4. Wiring		
Step	**Description**	**Reference**
1. Connecting Ethernet Cable	• Connecting the built-in EtherCAT port • Connecting the built-in EtherNet/IP port	*NX-series CPU Unit Hardware User's Manual (Cat. No. W535)* *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)* *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)* *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)* *NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)*
2. Wiring I/O	• Wiring I/O to EtherCAT slaves • Wiring Basic I/O Units and Special Units	Operation manuals for the EtherCAT slaves *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)* *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)* *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)* *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)* *NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)*
	• Checking wiring	*Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*
3. Connecting the Computer That Runs the Sysmac Studio	• Connecting the USB cable*1 • Connecting the built-in EtherNet/IP port	*Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*

*1. The NX701 CPU Units with hardware revision A or later, NX502, NX102, and NX1P2 CPU Units do not provide a USB port.

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. *1 Perform this before you create the slave configuration from the mounted Units, which is explained in STEP 2-1: *Creating the Slave and Unit Configurations*.	**Controller** - **Communications Setup** **Controller**– **Synchronization**	*Section 7 Checking Operation and Actual Operation* on page 7-1
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* on page 7-1
3. Actual Controller Operation	Start actual operation.	---	---

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

2

CPU Unit Operation

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

2-1 Overview of CPU Unit Operation 2-2
2-1-1 Introduction to CPU Unit 2-2
2-1-2 Overview of Operation According to CPU Unit Status 2-3

2-2 Software 2-4
2-2-1 Software Configuration 2-4
2-2-2 Operation of Software 2-4

2-3 Accessing I/O 2-13
2-3-1 Type of Variables 2-13
2-3-2 Accessing I/O with Variables 2-16

2-4 I/O Refreshing of NX Bus Function Module 2-26
2-4-1 I/O Refreshing Methods 2-26
2-4-2 I/O Refreshing Method Operation 2-27

2-5 I/O Refreshing of X Bus Function Module 2-30

2-6 Sequence Control and Motion Control 2-31
2-6-1 Overview of Control 2-31
2-6-2 Sequence Control System 2-32
2-6-3 Motion Control System 2-34
2-6-4 Synchronizing Sequence Control and Motion Control 2-36

2-7 Overview of CPU Unit Data 2-38

2-8 Operation for CPU Unit Status 2-40
2-8-1 CPU Unit Status 2-40
2-8-2 Operation for CPU Unit Status 2-41
2-8-3 Operating Modes 2-43

2-1 Overview of CPU Unit Operation

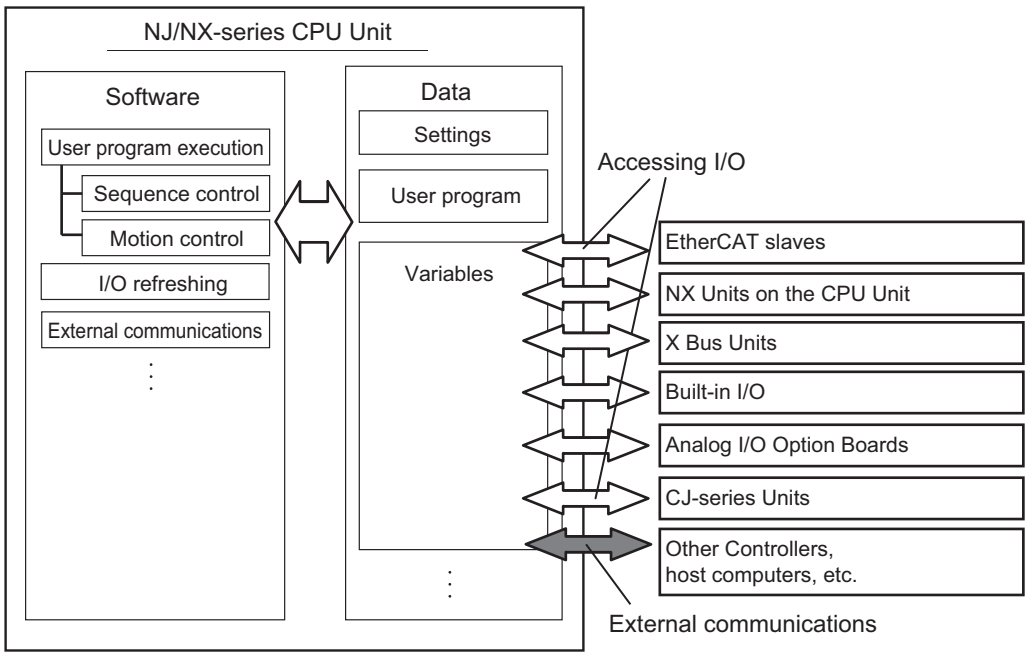
This section describes the operation of the CPU Unit and gives an overview of how it operates depending on the status of the CPU Unit.

2-1-1 Introduction to CPU Unit

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

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

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



Note You can use CJ-series Units only on the NJ-series CPU Units.
Note You can use NX Units on the CPU Unit only with the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units. You can use the built-in I/Os and Analog I/O Option Boards only with the NX1P2 CPU Units.
Note You can use X Bus Units only on the NX502 CPU Units.

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

Description	Reference
Software	page 2-4
Accessing I/O	page 2-13
Sequence control and motion control	page 2-31
Overview of CPU Unit data	page 2-38

Description	Reference
Operation for CPU Unit status	page 2-40

Additional Information

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

Communication	Manual
EtherNet/IP	• For information on using variables with the built-in EtherNet/IP port, refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)*. • For information on using variables with a CJ-series EtherNet/IP Unit, refer to the *CJ-series EtherNet/IP Units Operation Manual for NJ-series CPU Unit (Cat. No. W495)*. • For information on using variables with an NX-series EtherNet/IP Unit, refer to the *NX-series EtherNet/IP Unit User's Manual (Cat. No. W627)*.
Serial communications	*CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit (Cat. No. W494)*.

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

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

The CPU Unit operation according to the status of the CPU Unit is described in *2-8 Operation for CPU Unit Status* on page 2-40.

2-2 Software

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

2-2-1 Software Configuration

The software in the CPU Unit is divided into six modules. These functional units are called *function modules*.
The function modules and the processing that they perform are described in the following table.

Function module name	Processing
PLC Function Module	Performs the following services: task scheduling, commands for other function modules, event logging, execution of the user program, I/O refreshing*1 for the CJ-series Units, USB port services*2, SD Memory Card services, and data trace processing.
NX Bus Function Module*3	Performs processing such as event logging and I/O refreshing for the NX Units that are connected to the NX bus of the CPU Unit.
X Bus Function Module*4	Performs processing such as I/O refreshing for the X Bus Units that are connected to the X Bus of the CPU Unit.
Motion Control Function Module	Performs motion control processing. *5
EtherCAT Master Function Module	Performs communications with EtherCAT slaves as the EtherCAT master, including I/O refreshing*6 for the EtherCAT slaves, EtherCAT message communications*7, etc.
EtherNet/IP Function Module	Performs processing for EtherNet/IP communications, including tag data link processing, built-in EtherNet/IP port servicing, etc.

*1. Some CJ-series Units can also be connected to an NJ-series CPU Unit.
*2. The NX701 CPU Units with hardware revision A or later, NX502, NX102, and NX1P2 CPU Units do not provide a USB port.
*3. Only the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units have the NX Bus Function Module.
*4. Only the NX502 CPU Units have the X Bus Function Module.
*5. 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.
*6. 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.
*7. This module communicates with the EtherCAT slaves as the EtherCAT master.

2-2-2 Operation of Software

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

Type of CPU Unit processing	Status of CPU Unit	Execution conditions	Processing example
Initialization	Startup state	Initialization is performed only when the power supply is turned ON.	Self diagnosis at startup

Type of CPU Unit processing	Status of CPU Unit	Execution conditions	Processing example
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 for the built-in EtherNet/IP port
Option board service*1			I/O refreshing for Analog Option Boards
System services			USB port service*2, SD Memory Card service, and communications processing for Serial Communications Option Boards*3

*1. The option board service is executed only by an NX1P2 CPU Unit.
*2. The NX701 CPU Units with hardware revision A or later, NX502, NX102, and NX1P2 CPU Units do not provide a USB port.
*3. Only the NX1P2 CPU Units support Serial Communications Option Boards.

Refer to *2-8-1 CPU Unit Status* on page 2-40 for information on the CPU Unit status.

This section describes the operation of the processes.

Initialization

Initialization is performed only when the power supply is turned ON.
The following processing is performed for initialization.

Processing	Description
Self diagnosis at startup	Operation is monitored for the following errors: *Power Supply Error*, *CPU Unit Reset*, *CPU Unit Watchdog Timer Error*, and *Incorrect Power Supply Unit Connected*. *1
Data check	The *_RetainFail* (Retention Failure Flag) system-defined variable changes to TRUE at the following time: when the values of variables for which the Retain attribute was set to retain the values and the values in DM, EM, and HR Areas in the memory used for CJ-series Units*2 were not retained after a power interruption.
Detecting X Bus Units*5	The X Bus Units mounted in the Controller are detected.
Detecting NX Units*3	The NX Units mounted in the Controller are detected.
Detecting CJ-series Units*4	The CJ-series Units mounted in the Controller are detected.
Recording Power Turned ON and Power Interrupted events	The Power Turned ON and Power Interrupted events are recorded.

*1. Refer to *Types of Fatal Errors* in the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for information on the following errors: *Power Supply Error*, *CPU Unit Reset*, *CPU Unit Watchdog Timer Error*, and *Incorrect Power Supply Unit Connected*.

*2. You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*3. You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*4. You can use CJ-series Units only on the NJ-series CPU Units.
*5. You can use X Bus Units only on the NX502 CPU Units.

Processing Executed with Tasks

● Types of Processing That Are Executed with Tasks

The following processing is performed with tasks.

Processing	Description
I/O refreshing	Data I/O for EtherCAT slaves, NX Units on the CPU Unit*1, X Bus Units*2*3, I/O built in the CPU Unit*4, Analog I/O Option Boards in the CPU Unit*5, CJ-series*6 Basic I/O Units, and CJ-series Special Units is input and output.
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 for the built-in EtherNet/IP port, is performed.

*1. You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*2. You can use X Bus Units only on the NX502 CPU Units.
*3. For NX-series EtherNet/IP Units, tag data link processing is performed.
*4. You can use built-in I/O only with the NX1P2 CPU Units.
*5. You can use Analog Option Boards only with the NX1P2 CPU Units.
*6. You can use CJ-series Units only on the NJ-series CPU Units.

Refer to *5-3-3 Basic Operation of Tasks for NX701 CPU Units* on page 5-12, *5-4-3 Basic Operation of Tasks for NX502 CPU Units* on page 5-32, *5-5-3 Basic Operation of Tasks for NX102 CPU Units and NX1P2 CPU Units* on page 5-52, and *5-6-3 Basic Operation of Tasks for NJ-series Controllers* on page 5-69 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.

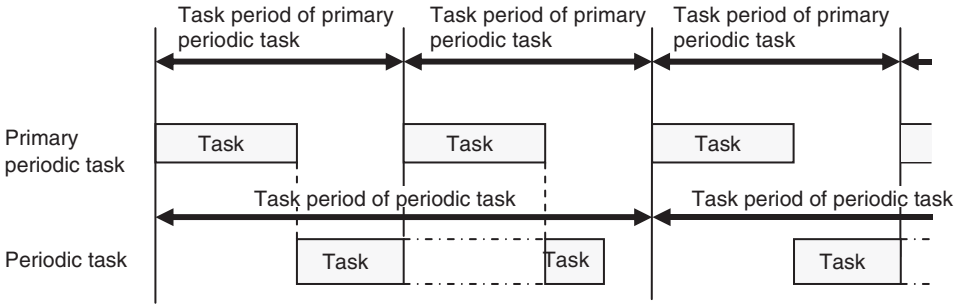
Type of periodic task	Execution priority (smaller values indicate higher priority)	Execution conditions	Main processing content
Primary periodic task	4 The task has the highest execution priority.	The periodic task is periodically executed.	I/O refreshing, user program execution, and motion control

Type of periodic task	Execution priority (smaller values indicate higher priority)	Execution conditions	Main processing content
Periodic task	5*1 This task has the next highest execution priority after the primary periodic task.	The periodic task is periodically executed. The task period is an integer multiple of the task period of the primary periodic task.	I/O refreshing, user program execution, and motion control
	16*2,17, or 18	The periodic task is periodically executed. The task period is an integer multiple of the task period of the primary periodic task.	The processing that can be performed depends on the task execution priority. Execution priority 16: I/O refreshing and user program execution Execution priority 17 or 18: User program execution
Event task	8 or 48	An event task is executed only once when the specified condition is met.	User program execution

*1. You can use the priority-5 periodic task only with the NX701 CPU Units.

*2. You cannot use the priority-16 periodic task with the NX102 CPU Unit or NX1P2 CPU Unit.

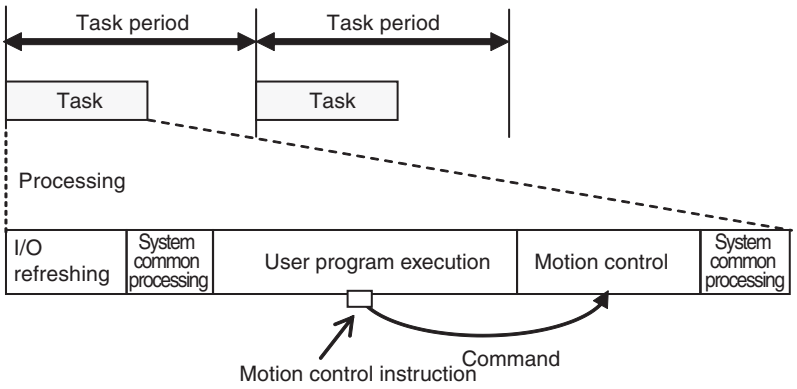
The CPU Unit executes the task with the highest execution priority first. The following operation example is for the primary periodic task and a periodic task. If the primary periodic task is executed while a periodic task is in execution, execution of the primary periodic task is prioritized.



The operation of tasks with lower priority during the execution of higher priority tasks differs between the NX701 CPU Units, NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units. Refer to *5-2 Overview of Tasks* on page 5-6 for details on task operation.

● Operation of Processing with Tasks

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



Note The CPU Unit executes motion control in the primary periodic task and in the priority-5 periodic task. Refer to *5-12-3 System Input and Output Response Times* on page 5-137 for details.

Additional Information

With an NX701 CPU Unit, you can execute motion control in the primary periodic task and in the priority-5 periodic task. If these two motion controls need to be identified, the motion control in the primary periodic task is called motion control 1, while the motion control in the priority-5 periodic task is called motion control 2.

Tag Data Link Service

● Processing Performed by the Tag Data Link Service

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

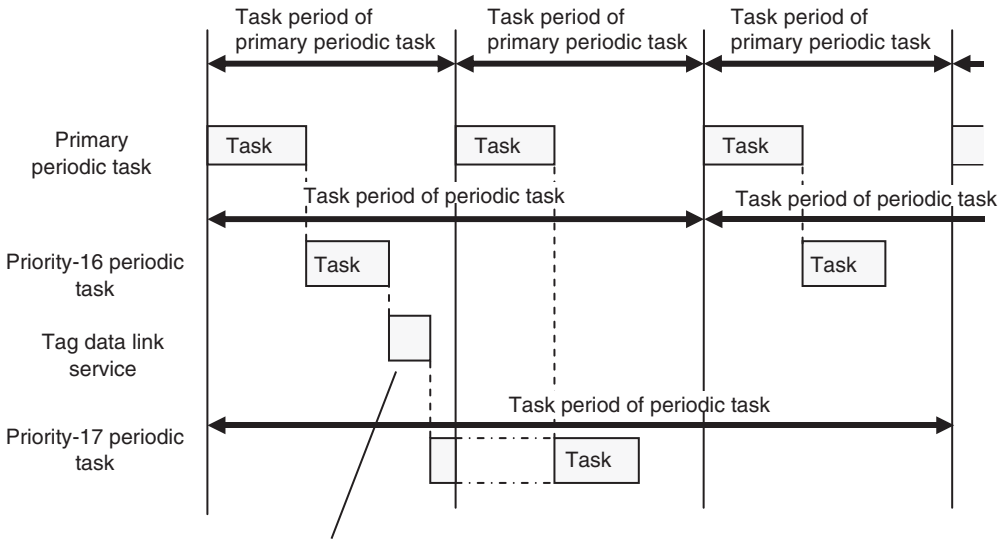
Note 1. You can use the CJ1W-EIP21 EtherNet/IP Unit only with NJ-series CPU Units.

Note 2. For the NX-series NX-EIP201 EtherNet/IP Unit, tag data link processing is performed by the I/O refreshing in the task, not by the tag data link service. Data exchange of network variables for tag data link communications is performed between the X Bus Units and the CPU Unit.

● Operation of the Tag Data Link Service

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

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



The tag data link service is executed with a priority that is between the execution priorities of the priority-16 periodic task and the priority-17 periodic task.

The tag data link service for an NX-series CPU Unit is executed in parallel with the execution of tasks.

For details on the tag data link service, refer to *5-7 Services Other Than Tasks* on page 5-86.

Option Board Service

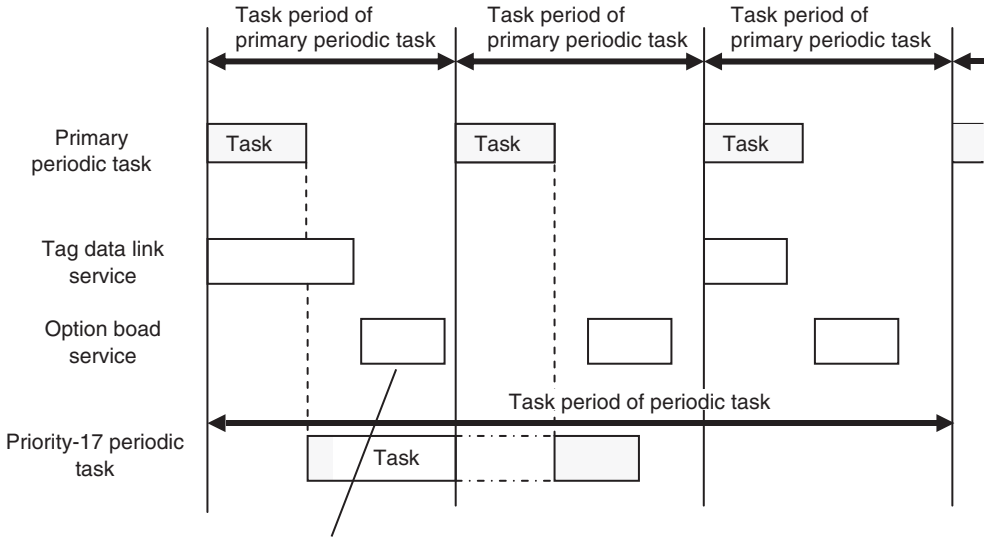
● Processing Performed by the Option Board Service

The option board service processes the I/O refreshing for an Analog Option Board that is mounted on the NX1P2 CPU Unit.

Note The option board service is executed only by an NX1P2 CPU Unit.

● Operation of the Option Board Service

The option board service for an NX1P2 CPU Unit is executed in parallel with the execution of tasks. However, during execution of the tag data link service, the option board service is not executed.



The option board service is executed in parallel with the execution of tasks.

Refer to *5-7 Services Other Than Tasks* on page 5-86 for details on the option board service.

Communications Bridge Service

● Processing Performed by the Communications Bridge Service

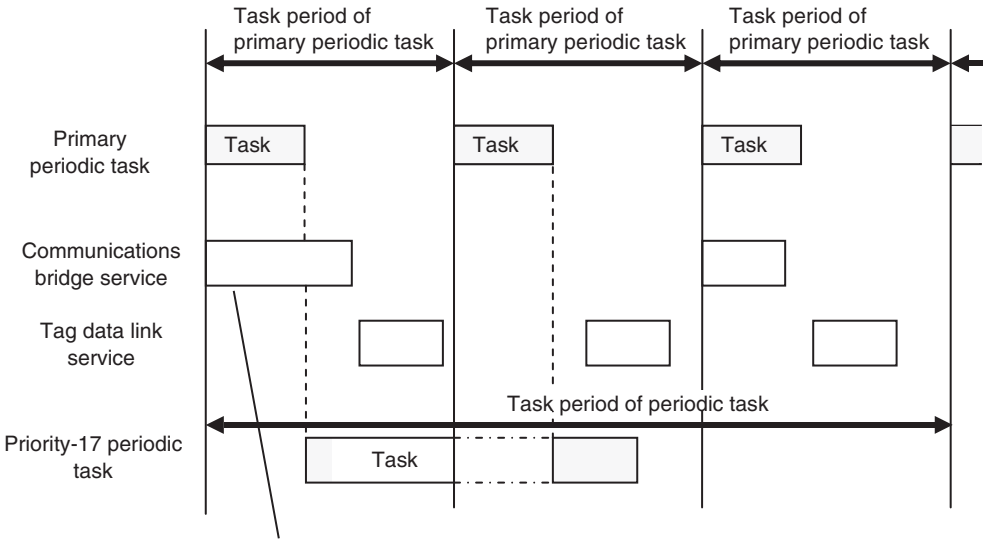
The communications bridge service performs processing to relay CIP Safety communications between CIP Safety on EtherNet/IP devices and the NX-SL5□□□ Safety CPU Unit that is mounted to the NX102 CPU Unit.

Refer to the *NX-series Safety Control Unit User's Manual (Cat. No. Z930-E1-12 or later)* for information on the CIP Safety communications.

Note The communications bridge service is executed by an NX102 CPU Unit with unit version 1.31 or later.

● Operation of the Communications Bridge Service

The communications bridge service for an NX102 CPU Unit is executed in parallel with the execution of tasks. The tag data link service is not executed during execution of the communications bridge service.



The communications bridge service is executed in parallel with the execution of tasks.

Refer to *5-7 Services Other Than Tasks* on page 5-86 for details on the communications bridge service.

System Services

● System Services

System services include the following processing.

Processing	Contents
USB port service*1	• Processing of service requests from the Sysmac Studio or host computers

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
Communications processing for a Serial Communications Option Board*2	• Execution of communications processing for a Serial Communications Option Board
CJ-series Special Unit service*3	• Event servicing for CJ-series Special Units • Execution of communications instructions (CIP)
SD Memory Card service	• Access from FTP client • SD Memory Card operations from the Sysmac Studio • Execution of SD Memory Card instructions
Self-diagnosis	• Hardware error detection

*1. The NX701 CPU Units with hardware revision A or later, NX502, NX102, and NX1P2 CPU Units do not provide a USB port.

*2. The communications processing for a Serial Communications Option Board is executed only by an NX1P2 CPU Unit.

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

● System Service Operations

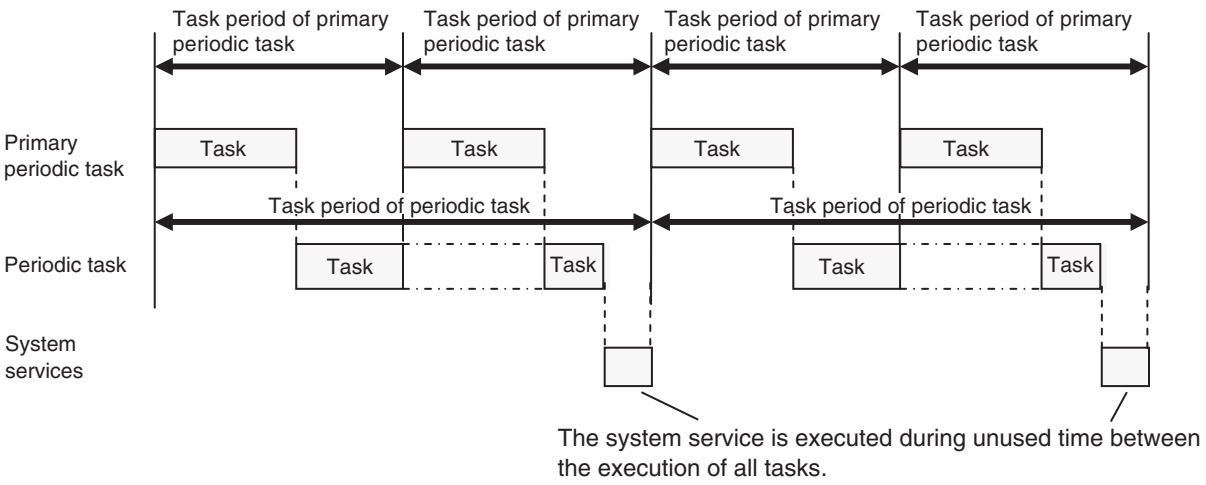
For the NX701 CPU Units, if a request comes from the hardware or from outside of the CPU Unit, system services are executed in parallel with other processes. System services are executed at the required time without being affected by the execution of tasks.

For the NX502 CPU Units, if a request comes from the hardware or from outside of the CPU Unit, system services are executed in parallel with other processes. System services are executed at the required time without being affected by the execution of tasks. However, during execution of the tag data link service, system services are not executed.

For the NX102 CPU Units, if a request comes from the hardware or from outside of the CPU Unit, system services are executed in parallel with other processes. System services are executed at the required time without being affected by the execution of tasks. However, during execution of the tag data link service, system services are not executed.

For the NX1P2 CPU Units, if a request comes from the hardware or from outside of the CPU Unit, system services are executed in parallel with other processes. System services are executed at the required time without being affected by the execution of tasks. However, during execution of the tag data link service or option board service, system services are not executed.

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



Refer to *5-7 Services Other Than Tasks* on page 5-86 for details on the system services.

2-3 Accessing I/O

The CPU Unit uses variables to access I/O. This section describes how variables are used to access I/O.
In this manual, I/O on EtherCAT slaves, NX Units on the CPU Unit, X Bus Units, built-in I/O, Analog I/O Option Boards, and CJ-series Units are treated as I/O. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for details on how to access data in other Controllers with tag data links.

2-3-1 Type of Variables

In an NJ/NX-series CPU Unit, you use variables in the user program to access I/O and memory in the CPU Unit.
The type of a variable depends on whether it has attributes that are set by the user, and what it can access.

Variables			Attribute settings	Accessed data
User-defined variables			All attributes can be set.	CPU Unit
Semi-user-defined variables	Device Variables	Device variables for EtherCAT slaves	Some attributes can be set. *1	EtherCAT slaves*2*3
		Device variables for NX Units*4		NX Units on the CPU Unit
		Device variables for X Bus Units*5		X Bus Units
		Device variables for built-in I/O*6		Built-in I/O
		Device variables for Option Boards*6		Analog I/O Option Boards
		Device variables for CJ-series Units*7		CJ-series Basic I/O Units and CJ-series Special Units
	Cam data variables			Servo Drives, encoder input slaves, and CPU Unit
System-defined variables	System-defined variables for PLC Function Module		No attributes can be set.	CPU Unit
	System-defined variables for motion control	MC Common Variable		Servo Drives, encoder input slaves, and CPU Unit
		Axis Variables		
		Axes Group Variables		
	System-defined variables for EtherNet/IP			Built-in EtherNet/IP Port
	System-defined variables for EtherCAT master			Built-in EtherCAT master port

Variables		Attribute settings	Accessed data
	System-defined variables for X Bus*9		CPU Unit
	System-defined variables for NX bus*8		CPU Unit

*1. Refer to *Device Variable Attributes* on page 3-12 for the attributes that can be set.
*2. "EtherCAT slaves" includes any NX Units on EtherCAT Slave Terminals.
*3. With the Sysmac Studio version 1.08 or lower, the EtherCAT slaves that are assigned to *axes* cannot be accessed via EtherCAT slave device variables.
*4. You can use device variables for NX Units only with the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*5. You can use device variables for X Bus Units only with the NX502 CPU Units.
*6. You can use device variables for built-in I/Os and Option Boards only with the NX1P2 CPU Units.
*7. You can use device variables for CJ-series Units only with NJ-series CPU Units.
*8. You can use system-defined variables for NX bus only with the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*9. You can use system-defined variables for X Bus only with the NX502 CPU Units.

User-defined Variables

The user defines all of the attributes of a user-defined variable. Refer to *6-3 Variables* on page 6-28 for details on user-defined variables.

Semi-user-defined Variables

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

● Device Variables

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

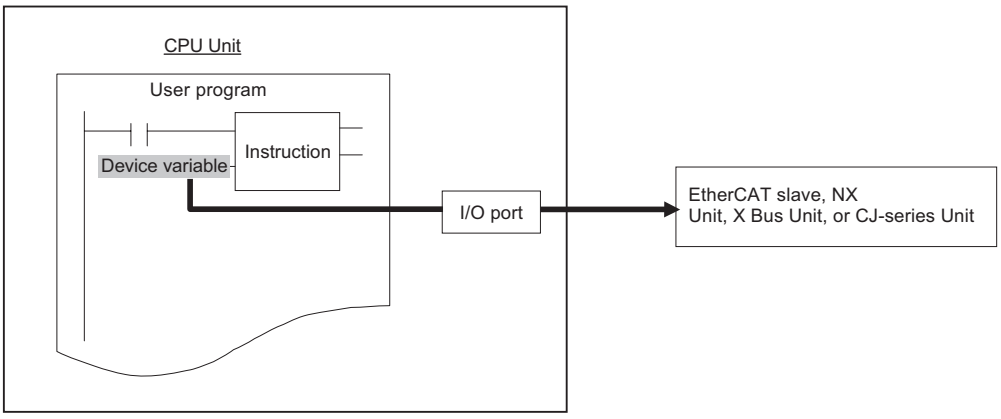
The device and the data to access in that device determine the type of device variable, as shown below.

Type of device variable	Device	Accessed data
Device variables for EtherCAT slaves	EtherCAT slaves*1	Process data for EtherCAT slaves*2
Device variables for NX Units*3	NX Units on the CPU Unit	I/O data for NX Units
Device variables for X Bus Units*8	X Bus Units	I/O data for X Bus Units
Device variables for built-in I/O*4	Built-in I/O	I/O data for built-in I/O
Device variables for Option Boards*4	Analog I/O Option Boards	I/O data for Analog Option Boards

Type of device variable	Device	Accessed data
Device variables for CJseries Units*5	CJ-series Basic I/O Units	Real I/O data in Basic I/O Units
	CJ-series Special Units	Operating data*6 and setup data*7 for Special Units

*1. With the Sysmac Studio version 1.08 or lower, the EtherCAT slaves that are assigned to *axes* cannot be accessed via EtherCAT slave device variables.
*2. This refers to I/O data that is exchanged during the process data communications cycle between the master and slaves.
*3. You can use device variables for NX Units only with the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*4. You can use device variables for built-in I/Os and Option Boards only with the NX1P2 CPU Units.
*5. You can use device variables for CJ-series Units only with NJ-series CPU Units.
*6. This data is used in the operation of CJ-series Units. The CIO Area portion of the memory used for CJ-series Units is used.
*7. This data is used to set up the CJ-series Units. The DM Area portion of the memory used for CJ-series Units is used.
*8. You can use device variables for X Bus Units only with the NX502 CPU Units.

Device variables are used to access data for EtherCAT slaves, NX Units on the CPU Unit, X Bus Units, built-in I/O, Analog I/O Option Boards, and CJ-series Units through the I/O ports. The I/O ports are logical ports that are used to access devices.



Refer to *3-3-1 I/O Ports* on page 3-8 for details on I/O ports and device variables.

● Cam Data Variables

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

System-defined Variables

System-defined variables are provided in advance in an NJ/NX-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
NX Bus Function Module*1	System-defined variables for NX bus

Function module	Type of system-defined variable
X Bus Function Module *2	System-defined variables for X Bus
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

*1. Only the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units have the NX Bus Function Module.
*2. Only the NX502 CPU Units have the X Bus Function Module.

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 Variables	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-7 System-defined Variables* on page A-91 for details on system-defined variables.

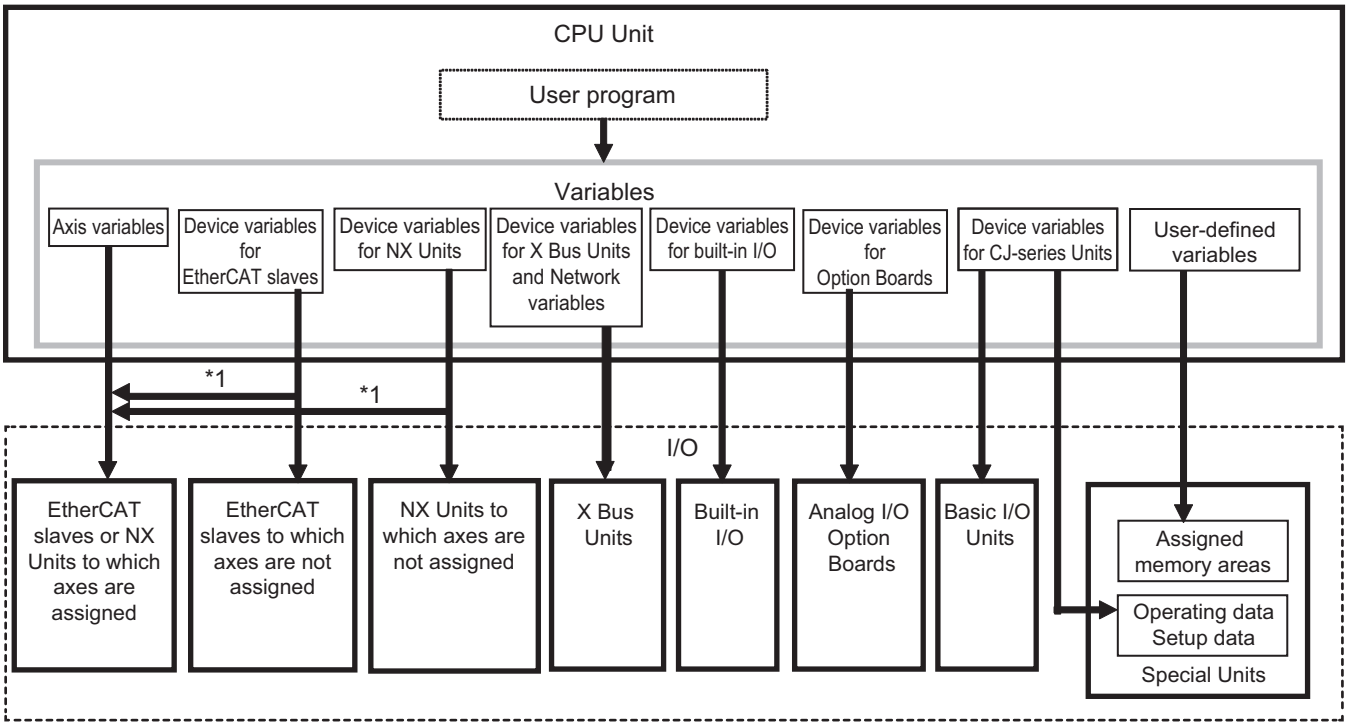
2-3-2 Accessing I/O with Variables

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

I/O		Data	Variables
EtherCAT slaves	EtherCAT slaves that are not assigned to *axes*	---	Device variables for EtherCAT slaves
	EtherCAT slaves that are assigned to *axes*	--- *1	• Device variables for EtherCAT slaves • Axis Variables
NX Units on the CPU Unit*2	NX Units that are not assigned to *axes*	---	Device variables for NX Units
	NX Units that are assigned to *axes*	---	• Device variables for NX Units • Axis variables
X Bus Units*7		---	• Device variables for X Bus Units • Network variables*8
Built-in I/O*3		---	Device variables for built-in I/O
Analog I/O Option Boards*4		---	Device variables for Option Boards

I/O		Data	Variables
CJ-series Units*5	Basic I/O Units	---	Device variables for CJ-series Units
	Special Units	• Operating data • Setup data	Device variables for CJ-series Units
		Assigned memory area data*6	User-defined variables

*1. With the Sysmac Studio version 1.08 or lower, the EtherCAT slaves that are assigned to *axes* cannot be accessed via EtherCAT slave device variables.
*2. You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*3. You can use built-in I/O only with the NX1P2 CPU Units.
*4. You can use Analog Option Boards only with the NX1P2 CPU Units.
*5. You can use CJ-series Units only on the NJ-series CPU Units.
*6. This data is for extended functions and slave I/O that you assign by specifying addresses in memory. You cannot access assigned memory area data with device variables.
*7. You can use X Bus Units only on the NX502 CPU Units.
*8. For NX-series EtherNet/IP Units, the data is used for tag data links.



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*1 • General-purpose slaves	These slaves are accessed through I/O ports by using device variables for EtherCAT slaves.
Servo Drive and encoder input slaves that are assigned to axes	These slaves are accessed directly with axis variables.*2

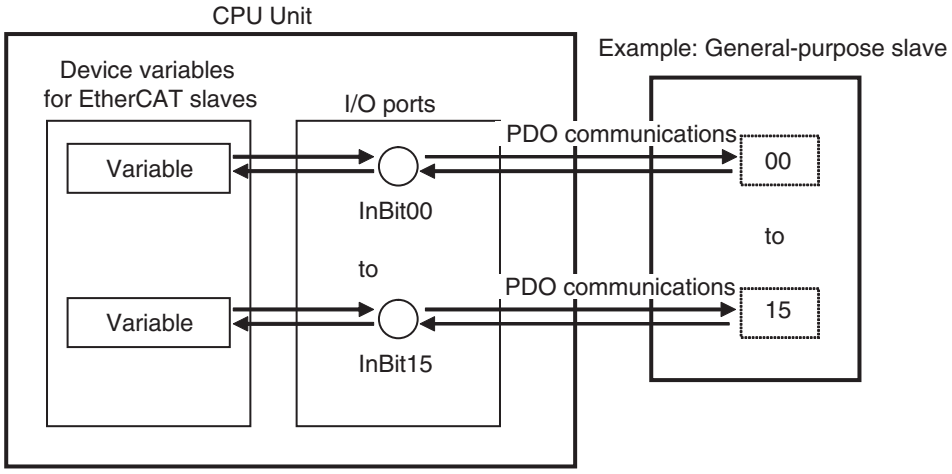
*1. With the Sysmac Studio version 1.08 or lower, the EtherCAT slaves that are assigned to axes cannot be accessed via EtherCAT slave device variables.

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

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

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

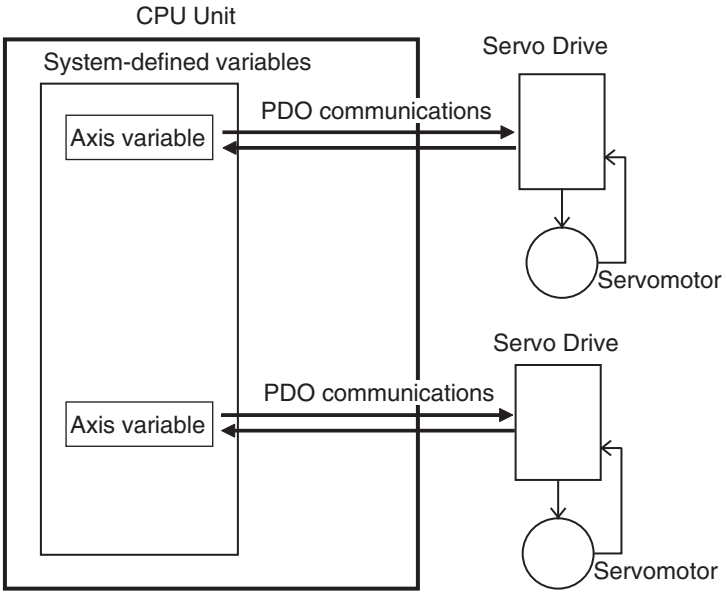
These slaves are accessed through I/O ports for device variables for EtherCAT slaves. PDO communications are used to access data from I/O ports.



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

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

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



Refer to *3-5-2 Axis Variables and Axes Group Variables* on page 3-19 for details on axis variables.

Version Information

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

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

Precautions for Correct Use

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

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 that are assigned to axes.
2. Save the project data.
3. Open the saved project data with the Sysmac Studio version 1.08 or lower.

Additional Information

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

Accessing NX Units on the CPU Unit

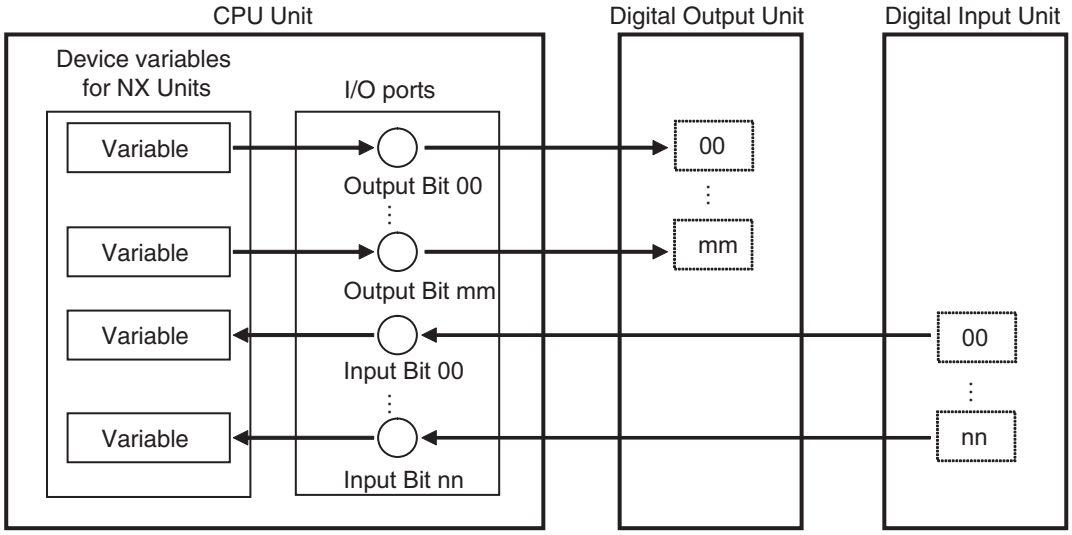
The method that is used to access an NX Unit on the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit depends on the type of NX Unit.

Type of NX Unit on the CPU Unit*1	Access method
NX Units that are not assigned to *axes*	These Units are accessed through I/O ports by using device variables for NX Units.
NX Units that are assigned to *axes*	These Units are accessed directly with axis variables.
	These Units are accessed through I/O ports by using device variables for NX Units.

*1. You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

● Accessing NX Units That Are Not Assigned to Axes

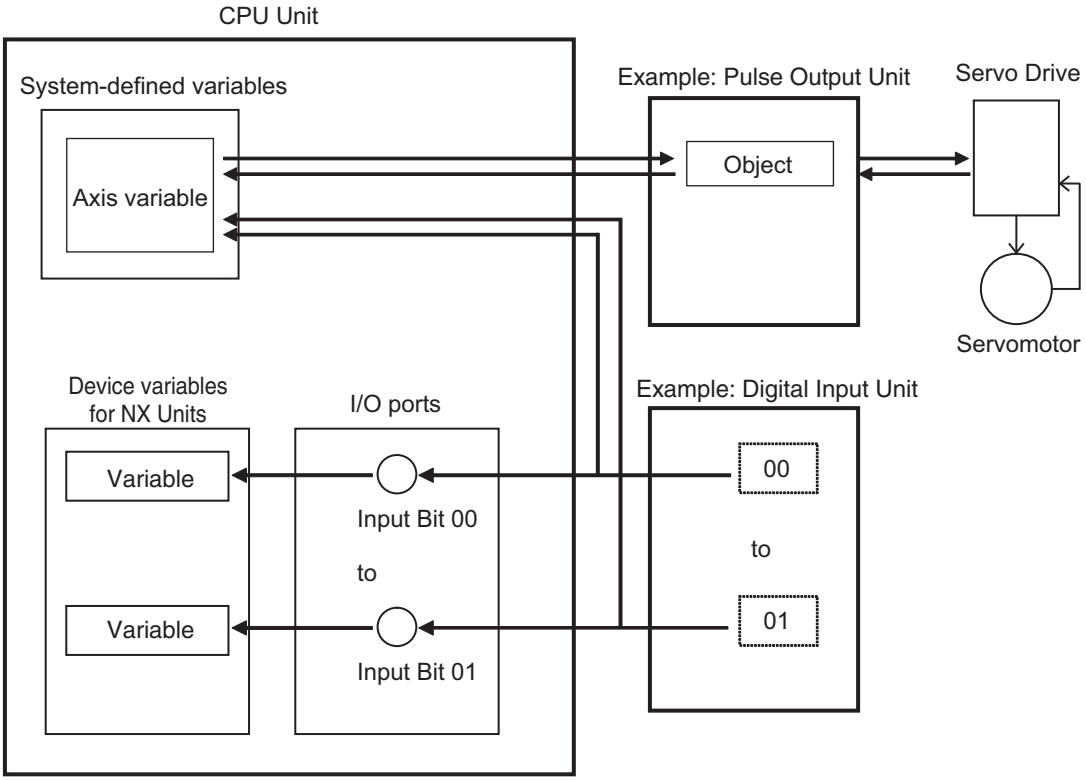
These Units are accessed through I/O ports for device variables for NX Units.



● Accessing NX Units That Are Assigned to Axes

NX Units that are assigned to axes are accessed directly through the axis variable.

You can also assign the device variables to the I/O ports of NX Units that are assigned to axes.



Refer to *3-5-2 Axis Variables and Axes Group Variables* on page 3-19 for details on axis variables.

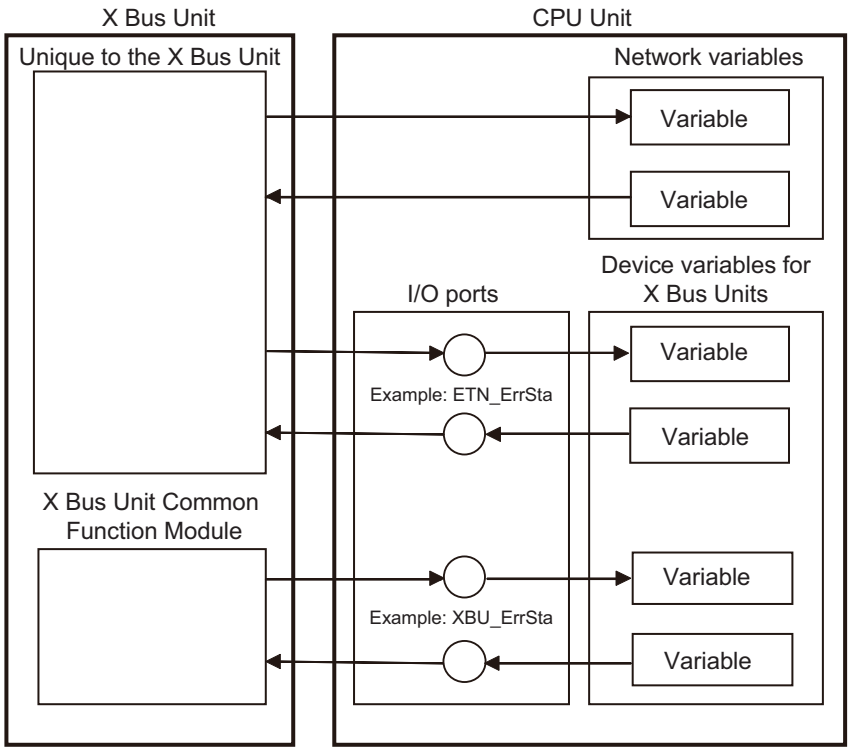
Precautions for Correct Use

The I/O port to which a device variable can be assigned in the NX Unit that is assigned to an axis 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.

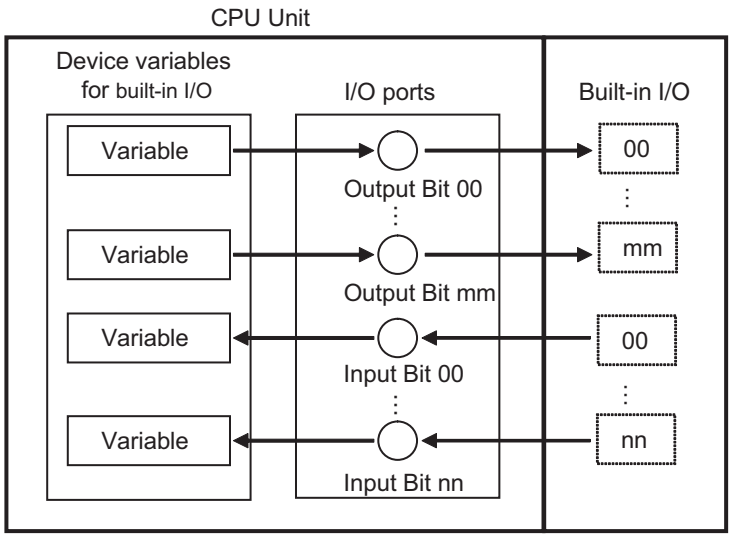
Accessing X Bus Units

You access X Bus Units through the I/O ports for network variables and device variables for X Bus Units.



Accessing Built-in I/O

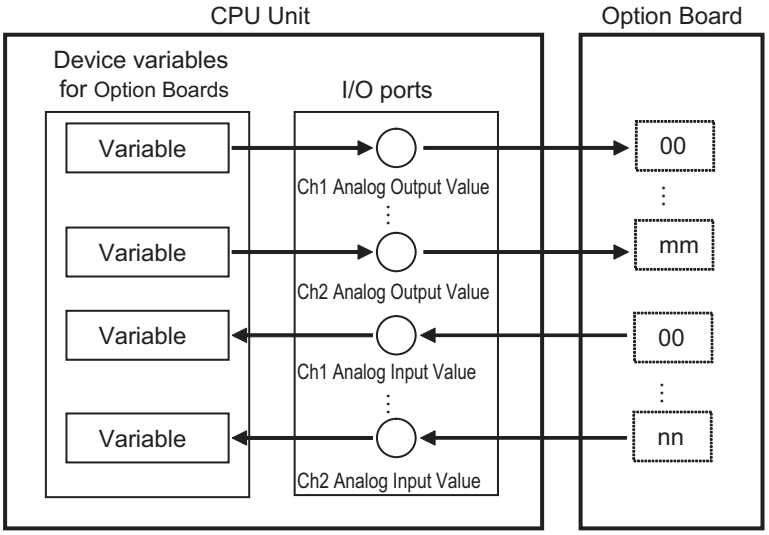
You access I/O that is built in an NX1P2 CPU Unit through the I/O ports for device variables for built-in I/O.



Refer to *3-3-1 I/O Ports* on page 3-8 for details.

Accessing Analog I/O Option Boards

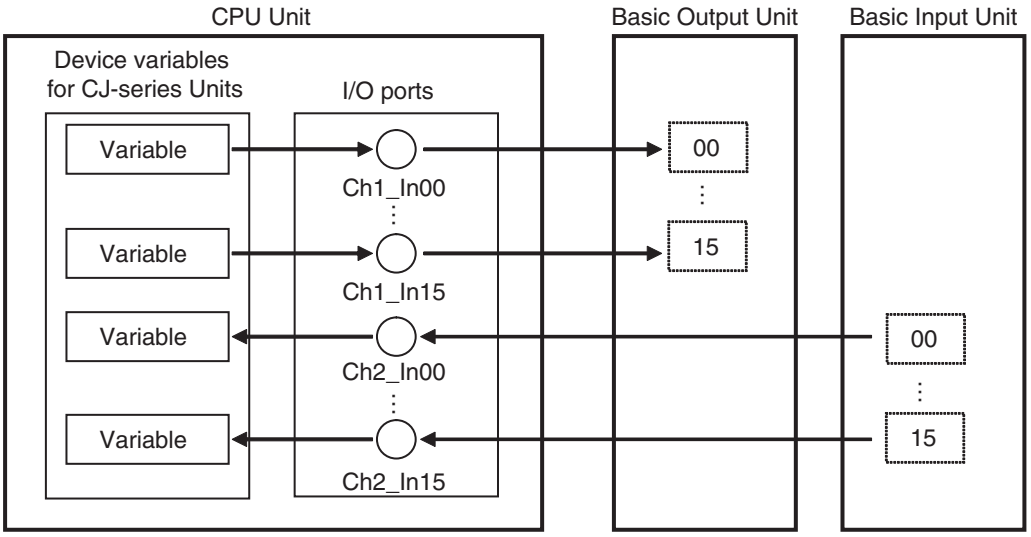
You access Analog I/O Option Boards in an NX1P2 CPU Unit through the I/O ports for device variables for Option Boards.



Refer to *3-3-1 I/O Ports* on page 3-8 for details.

Accessing Basic I/O Units

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



Refer to *3-3-1 I/O Ports* on page 3-8 for details.

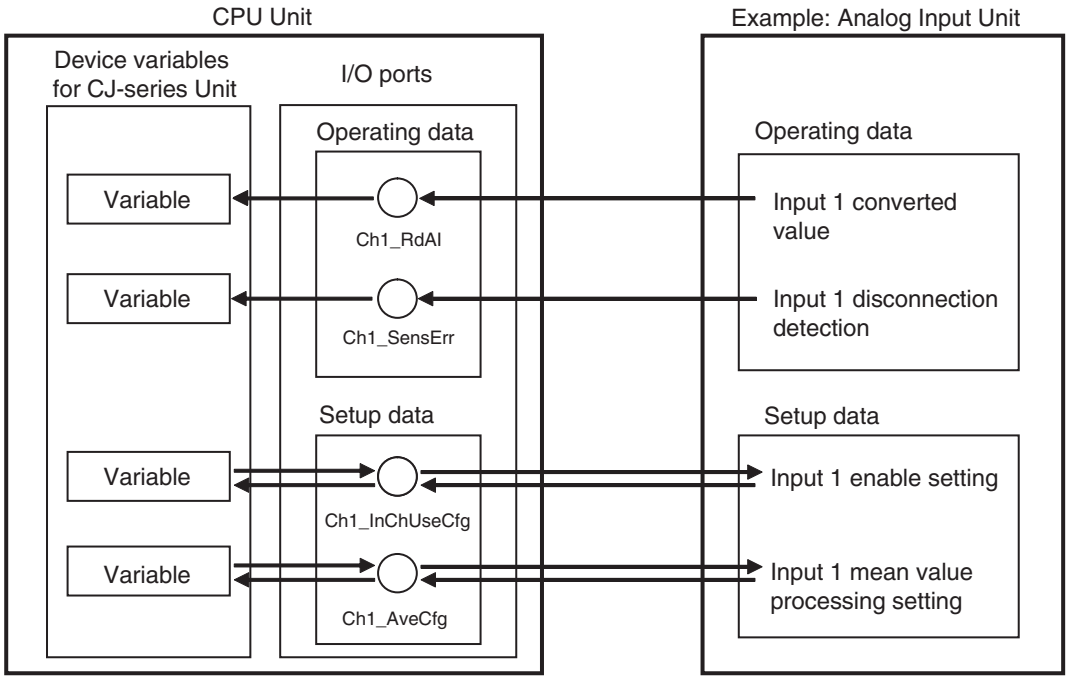
Accessing Special Units

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

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

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

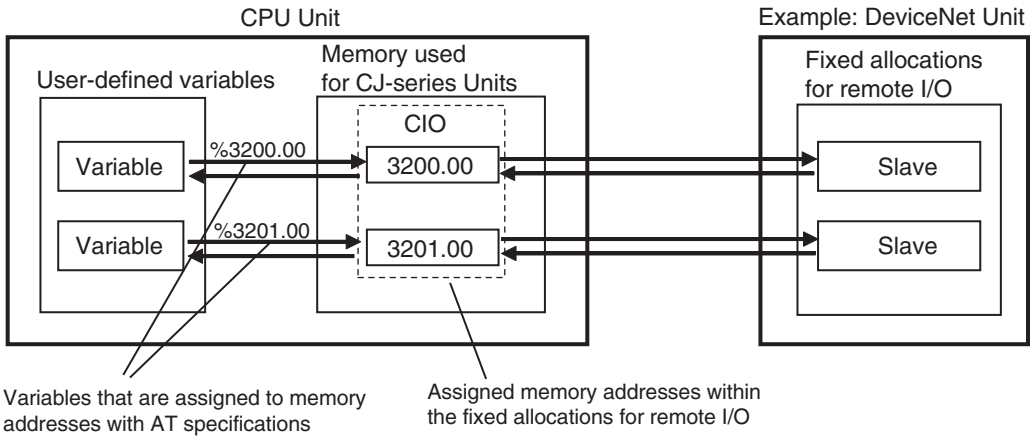
The operating data and setup data are accessed through the I/O ports for the device variables for the CJ-series Unit. The data is exchanged during I/O refreshing.



Refer to *3-3-1 I/O Ports* on page 3-8 for details.

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

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



The assigned memory addresses including the following addresses.

- Addresses in fixed allocations for DeviceNet Units
- Addresses in user-specified allocations for DeviceNet Units or CompoNet Master Units from the CX-Integrator (A CompoNet Master Unit must be set to communications mode 8 to use the CX-Integrator.)

- Addresses in expansion memory for High-speed Counter Units
- Addresses in expansion memory for Process I/O Units

Refer to *A-10 Contents of Memory Used for CJ-series Units* on page A-247 for information on the memory used for CJ-series Units. Refer to *6-3-8 Variable Attributes* on page 6-60 for information on AT specifications.

2-4 I/O Refreshing of NX Bus Function Module

This section describes I/O refreshing of the NX Bus Function Module for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

The NX Bus Function Module of the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit performs the data exchange cyclically with the NX Units on the CPU Unit.
The data exchange is executed by I/O refreshing in the primary periodic task. Therefore, the period of I/O refreshing is the task period of the primary periodic task.

Precautions for Correct Use

Only the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units have the NX Bus Function Module.

2-4-1 I/O Refreshing Methods

The I/O refreshing methods of NX Units that are mounted on the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit are listed below.

I/O refreshing method for NX Units	Actual operation
NX Units that support both Free-Run refreshing and synchronous I/O refreshing	Operates with synchronous I/O refreshing.
NX Units that support Free-Run refreshing, synchronous I/O refreshing, and task period prioritized refreshing	
NX Units that support only time stamp refreshing	Operates with time stamp refreshing.
NX Units that support only Free-Run refreshing	Operates with Free-Run refreshing.

Note NX Units with different I/O refreshing methods can be mixed on the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit.

Note You cannot change the I/O refreshing methods for NX Units on the NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit.

An outline of the I/O refreshing methods that are actually operated in the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit is listed below.

I/O refreshing method		Outline
Synchronous I/O refreshing		With this I/O refreshing method, the timing to read inputs or to refresh outputs is synchronized on a fixed interval between more than one NX Units.
Time stamp refreshing		With this I/O refresh method, the NX Units record the DC times*1 of the input change asynchronously with the refresh cycle of the NX bus or outputs at the specified DC time. Data exchange between the NX Units and CPU Unit are performed cyclically on the NX bus refresh cycles.
	Input refreshing with input changed times	With this I/O refreshing method, the Input Units record the DC times when inputs change.
	Output refreshing with specified time stamps	With this I/O refreshing method, the Output Units refresh outputs at specified DC times.

I/O refreshing method	Outline
Free-Run refreshing	With this I/O refreshing method, the refresh cycle of the NX bus and the I/O refresh cycles of the NX Units are asynchronous.

*1. The slaves or Units that support distributed clock synchronization have a clock that is shared by the slaves and Units. The time that is based on this distributed clock is called the DC time.

The I/O refreshing methods that you can use depend on the model of the NX Unit. Select the NX Units according to the I/O refreshing method to use.
Refer to the manuals for the specific Units for the I/O refreshing methods that are supported by individual NX Units.

2-4-2 I/O Refreshing Method Operation

This section describes the operation of each I/O refreshing method in the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit.
The detailed operation of the I/O refreshing methods depends on the NX Units. Refer to the manuals for the NX Units.

Operation of Synchronous I/O Refreshing

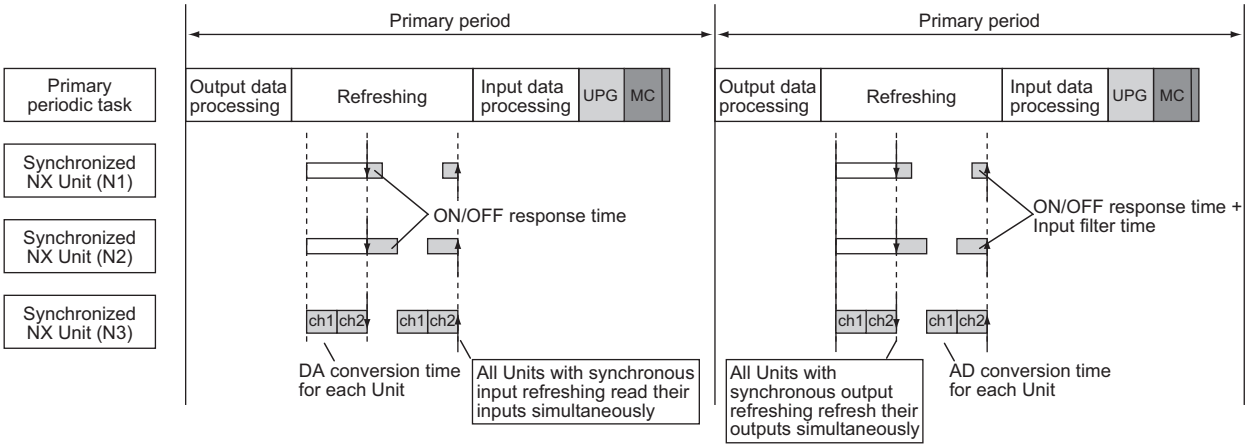
All NX Units that support synchronous I/O refreshing on the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit read their inputs at the same time that is synchronized with the I/O refreshing in the primary periodic task. Outputs are also refreshed simultaneously, but at a separately set timing from inputs.

Precautions for Correct Use

The NX Units on the CPU Unit, EtherCAT slaves, and EtherCAT Slave Terminals do not read inputs or refresh outputs simultaneously.

● Operation Example of Synchronous I/O Refreshing

All NX Units that support synchronous I/O refreshing read their inputs at the same time.
All NX Units that support synchronous I/O refreshing refresh their outputs at the same time.



Operation of Time Stamp Refreshing

The following describes the operation of time stamp refreshing.

● Operation of Input Refreshing with Input Changed Times

All NX Units that support input refreshing with input changed times on the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit record the DC times when inputs change (called input changed times). Then, the input changed times are read at the time that is synchronized with the I/O refreshing in the primary periodic task. The most recent values are always read as the input values.

Additional Information

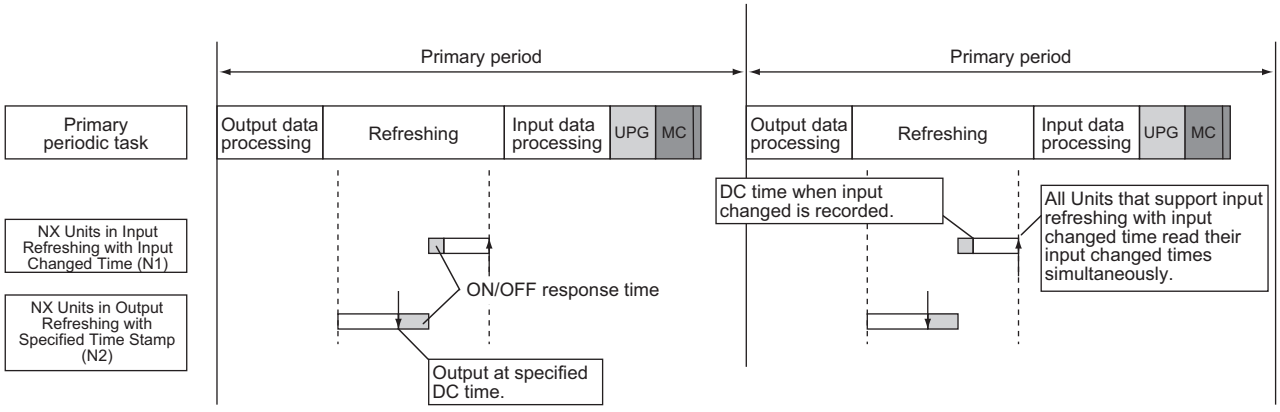
The timing when the input changed times are read is the same as that for reading the inputs for NX Units that support synchronous input refreshing.

● Operation of Output Refreshing with Specified Time Stamps

All NX Units that support output refreshing with specified time stamps on the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit change the outputs at the specified DC times for each NX Unit.

● Operation Example of Time Stamp Refreshing

All NX Units that support input refreshing with input changed times record the DC times when inputs change for each NX Unit and read the input changed times at the same time. All NX Units that support output refreshing with specified time stamps change the outputs at the specified DC times for each NX Unit.



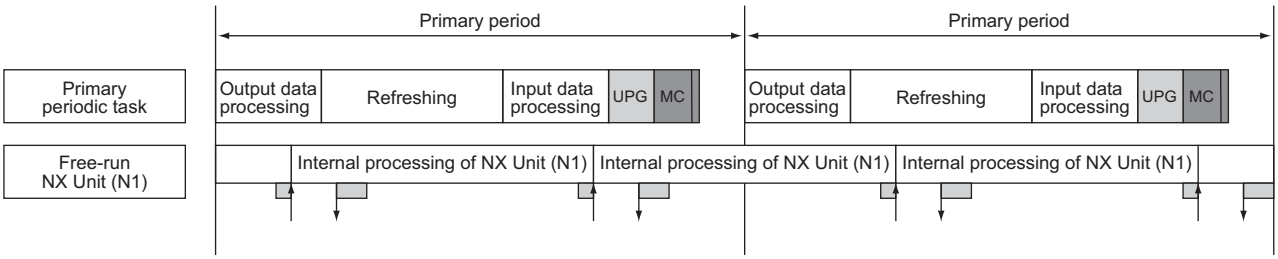
Operation of Free-Run Refreshing

With Free-Run refreshing, the refresh cycle of the NX bus and the I/O refresh cycle of the NX Units operate asynchronously.

● Operation Example of Free-Run Refreshing

An example of operation for Free-Run refreshing method in the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit is provided in the following figure.

The NX Units that support Free-Run refreshing perform I/O processing based on their own unique timings that is asynchronous with the I/O refreshing in the primary periodic task.


Primary period
Primary period
Primary periodic task
Output data processing
Refreshing
Input data processing
UPG
MC
Output data processing
Refreshing
Input data processing
UPG
MC
Free-run NX Unit (N1)
Internal processing of NX Unit (N1)
Internal processing of NX Unit (N1)
Internal processing of NX Unit (N1)

2-5 I/O Refreshing of X Bus Function Module

This section describes I/O refreshing of the X Bus Function Module for the NX502 CPU Units.
The X Bus Function Module of the NX502 CPU Unit performs the data exchange cyclically with the X Bus Units.
The data exchange is executed by I/O refreshing in the task.

Precautions for Correct Use

- Only the NX502 CPU Units have the X Bus Function Module.
- The task execution time is affected by data exchange between the NX-series EtherNet/IP Unit and the CPU Unit because the exchange is performed by I/O refreshing in the task.
  The following table shows the difference in processing timing when tag data link communications are performed between the built-in EtherNet/IP port and an NX-series EtherNet/IP Unit.

EtherNet/IP port	Data exchange processing for tag data links	Data update processing
NX-series EtherNet/IP Unit	I/O refreshing in the task	System common processing 2 in the task
Built-in EtherNet/IP	Tag data link service	System service

Additional Information

The method to reduce task execution time is to divide the I/O refreshing task.
The I/O refreshing task can be divided into the primary periodic task and the priority-16 periodic task.
The I/O refreshing task is set in the settings for exclusive control of variables in tasks.
If the settings for exclusive control of variables in tasks are not set, the I/O refreshing task is determined depending on the presence of priority-16 periodic task. The table is shown below.

Priority-16 periodic task	Settings for exclusive control of variables in tasks	I/O refreshing task
Not present	No settings or primary periodic task	Primary periodic task
Present	No settings or priority-16 periodic task	Priority-16 periodic task
	Primary periodic task	Primary periodic task

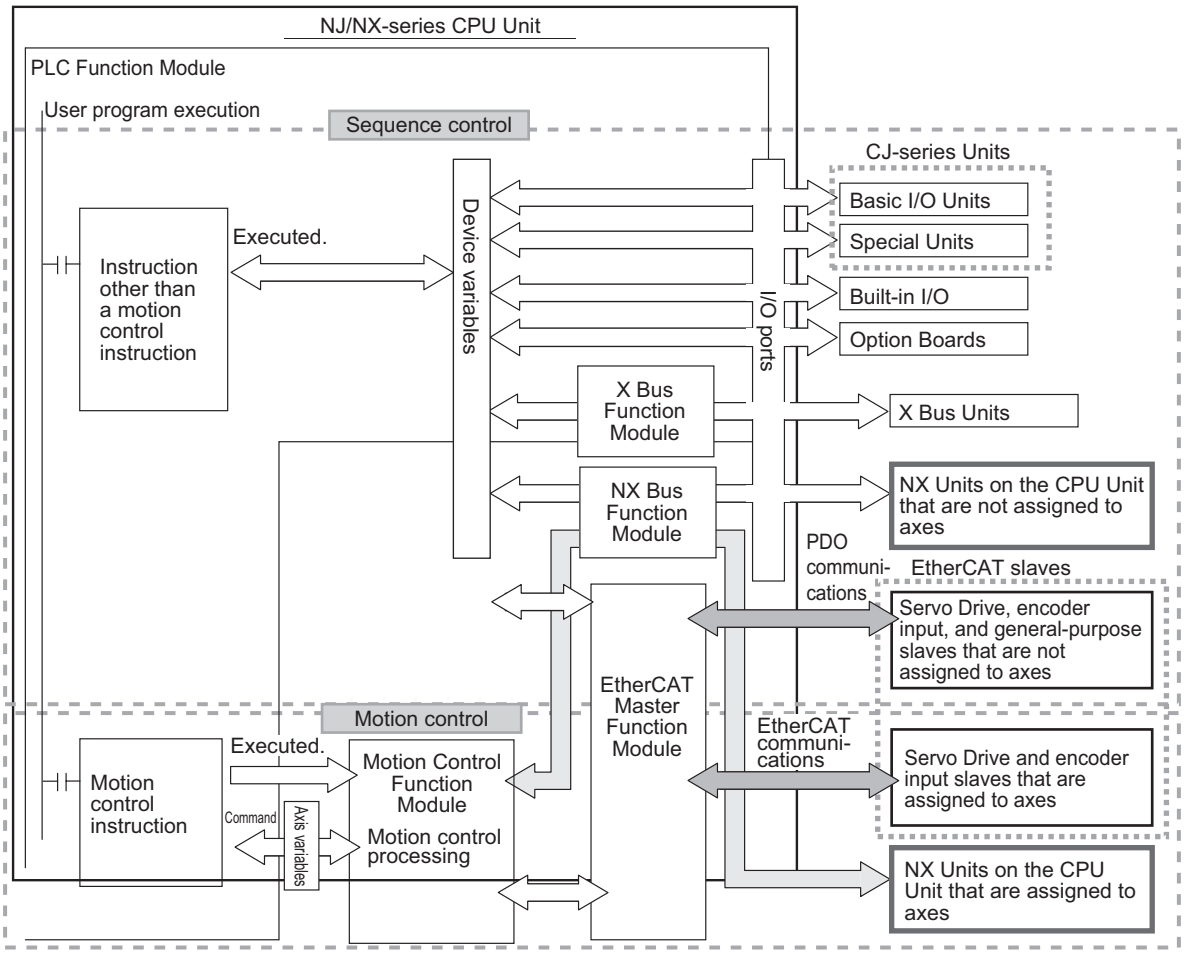
For NX-series EtherNet/IP Units, the data exchange for tag data link communications and CIP Safety communications is performed.
The task to perform I/O refreshing with CIP Safety communications is fixed for the primary periodic task. However, the task to perform I/O refreshing with tag data link communications can be set for each tag set.

2-6 Sequence Control and Motion Control

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

2-6-1 Overview of Control

The NJ/NX-series CPU Unit can perform both sequence control and motion control.
You execute sequence control with instructions other than motion control instructions in the user program. Sequence control is for EtherCAT slaves, NX Units on the CPU Unit, X Bus Units, built-in I/O, Option Boards, and CJ-series Units. Control is performed by the PLC Function Module, NX Bus Function Module, X Bus Function Module, and the EtherCAT Master Function Module.
You perform motion control with motion control instructions in the user program for EtherCAT Servo Drives and encoder input slaves that are assigned to *axes*. Control is performed by the PLC Function Module, Motion Control Function Module, NX Bus Function Module, and the EtherCAT Master Function Module.



Note You can use CJ-series Units only on the NJ-series CPU Units.
Note You can use the NX Bus Function Module only with the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
Note You can use the X Bus Function Module only with the NX502 CPU Units.
Note You can use the built-in I/Os and Option Boards only with the NX1P2 CPU Units.

Additional Information

Instruction Types in Terms of Control Systems

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

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

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

Version Information

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

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

Precautions for Correct Use

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

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 that are assigned to *axes*.
2. Save the project data.
3. Open the saved project data with the Sysmac Studio version 1.08 or lower.

2-6-2 Sequence Control System

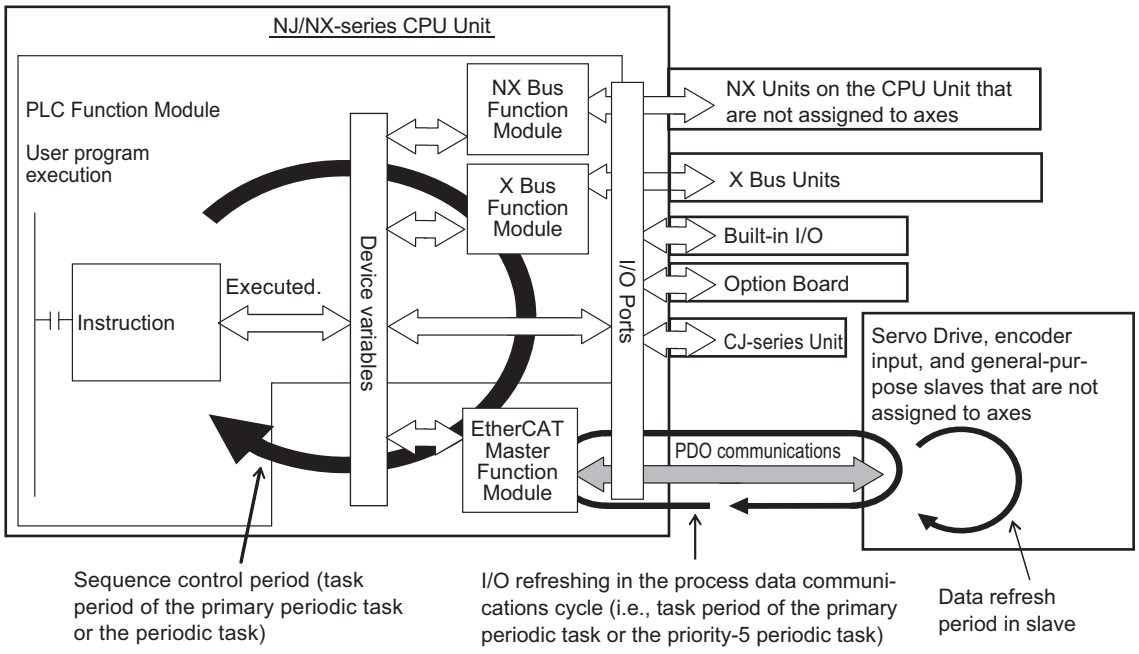
The way that the sequence control works depends on the device to control. This section describes the operation of the function modules and the control period as part of the sequence control system.

Device	Sequence Control System	
	Operation of Function Modules	Control Period
Servo Drive, encoder input[*1], and general-purpose slaves that are not assigned to *axes*	• The PLC Function Module executes the user program and refreshes the device variables. • The EtherCAT Master Function Module exchanges data with the slaves through the I/O ports for device variables.	The task period of the task to which the program is assigned (i.e., the task period of the primary periodic task or a periodic task)[*2]

Device	Sequence Control System	
	Operation of Function Modules	Control Period
NX Units on the CPU Unit that are not assigned to *axes*[*3]	• The PLC Function Module executes the user program and refreshes the device variables. • The NX Bus Function Module exchanges data with the NX Units on the CPU Unit through the I/O ports for device variables.	Primary periodic task[*4]
X Bus Units[*10]	• The PLC Function Module executes the user program and refreshes the device variables. • The X Bus Function Module exchanges data with the X Bus Units 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)
Built-in I/O[*5]	The PLC Function Module executes the user program, refreshes the device variables, and exchanges data with the built-in I/O.	Primary periodic task
Option Boards[*6]	The PLC Function Module executes the user program, refreshes the device variables, and exchanges data with the Option Boards.	Primary periodic task[*7]
CJ-series Units[*8]	The PLC Function Module executes the user program, refreshes the device variables, and exchanges data with the CJ-series Units.	The task period of the task to which the program is assigned (i.e., the task period of the primary periodic task or a periodic task)[*9]

*1. With the Sysmac Studio version 1.09 or higher, a Servo Drive and encoder input slave that is assigned to an *axis* can also be a part of sequence controls if you assign the device variable to the I/O port of the slave.
*2. The data refresh period in the slave depends on settings in the slave.
*3. You can use NX Units on the CPU Unit only with the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units. An NX Unit on the CPU Unit to which an axis is assigned can also be a part of sequence controls if you assign the device variable to the I/O port of the NX Unit.
*4. The I/O refresh cycles of the NX Units that operate with Free-Run refreshing and the control period of the CPU Unit are asynchronous.
*5. You can use built-in I/O only with the NX1P2 CPU Units.
*6. You can use Option Boards only with the NX1P2 CPU Units.
*7. The primary periodic task and the data exchange period of Option Boards are asynchronous.
*8. You can use CJ-series Units only on the NJ-series CPU Units.
*9. The data exchange period with a CJ-series Unit is the task period to which I/O refreshing for the CJ-series Unit is assigned.
*10. You can use X Bus Units only on the NX502 CPU Units.

Servo Drive, encoder input, and general-purpose slaves that are not assigned to *axes* are refreshed in the process data communications cycle. This means that I/O refreshing takes place in the task period of the primary periodic task or the priority-5 periodic task. However, execution of the programs and refreshing of the device variables take place in the task period of the task to which the programs are assigned. Therefore, the slave values are not reflected and not controlled by the device variables until the task period of the task to which the programs are assigned.
If it is necessary to control a slave in the process data communications cycle, assign the program that controls the slave to the primary periodic task or the priority-5 periodic task.
Refer to *5-12-3 System Input and Output Response Times* on page 5-137 for details.



Additional Information

- You can use the priority-5 periodic task only with the NX701 CPU Unit.
- With an NX701 CPU Unit, you can perform process data communications in the primary periodic task and the priority-5 periodic task. If these two process data communications cycles need to be identified, the communications cycle for the primary periodic task is called *process data communications cycle 1*, while the communications cycle for the priority-5 periodic task is called *process data communications cycle 2*.
- The NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units perform process data communications only in the primary periodic task.
- You can use NX Units only on the NX502, NX102, and NX1P2 CPU Units.
- You can use the built-in I/Os and Option Boards only with the NX1P2 CPU Units.
- You can use CJ-series Units only on the NJ-series CPU Units.
- You can use X Bus Units only on the NX502 CPU Units.

2-6-3 Motion Control System

This section describes the operation of the function modules and the control period as part of the sequence 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 slaves by using EtherCAT communications.
- The NX Bus Function Module outputs the command values of the axis variable to the NX-series Pulse Output Unit or other Unit on the CPU Unit.

Control Period

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

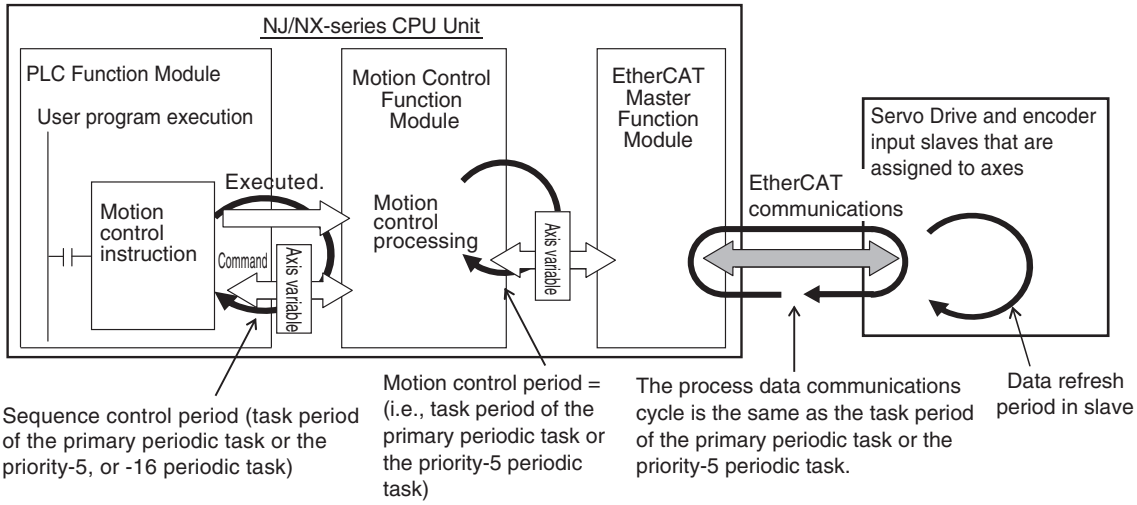
Motion control processing in the Motion Control Function Module is executed in the task period of the primary periodic task or the priority-5 periodic task. The Motion Control Function Module also exchanges data with Servo Drive and encoder input slaves that are assigned to the *axes* to control in the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

Furthermore, in the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units, the Motion Control Function Module exchanges data with NX Units on the CPU Unit that are assigned to the axes to control in the primary periodic task.

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

However, the following restrictions apply:

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



Precautions for Correct Use

- You can use the priority-5 periodic task only with the NX701 CPU Unit.
- With an NX701 CPU Unit, you can execute motion control in the primary periodic task and in the priority-5 periodic task. If these two motion controls need to be identified, the motion control in the primary periodic task is called *motion control 1*, while the motion control in the priority-5 periodic task is called *motion control 2*.
- The NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units perform motion control only in the primary periodic task.

Additional Information

- Use the Sysmac Studio to assign an axis to an EtherCAT slave and NX Unit mounted on the NX502, NX102, and NX1P2 CPU Units, which are controlled by 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. Refer to *5-12-3 System Input and Output Response Times* on page 5-137 for details.

2-6-4 Synchronizing Sequence Control and Motion Control

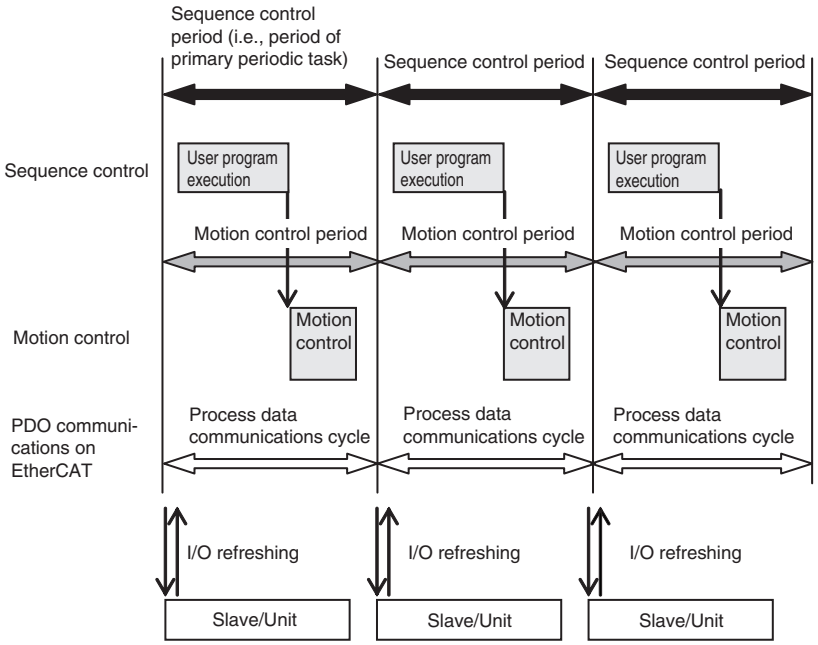
The sequence control period is the task period of the task to which the program and I/O refreshing are assigned. However, motion control is always executed in the task period of the primary periodic task or the priority-5 periodic task. The process data communications cycle for the EtherCAT slave to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

With an NX701 CPU Unit, you can set motion control to execute motion processing for each axis and each axes group. For axes or axes groups for which you set motion control 1, motion control and process data communications are executed in the task period of the primary periodic task. For axes or axes groups for which you set motion control 2, motion control and process data communications are executed in the task period of the priority-5 periodic task.

The NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units perform motion control and process data communications for axes and axes groups in the task period of the primary periodic task.

If you assign the sequence control program to the task that motion control is executed, you can synchronize the sequence control period and motion control period 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 sequence control period, motion control period, and process data communications cycle on EtherCAT are all synchronized.



Additional Information

- Relationship among motion controls, tasks, and process data communications cycles for the NX701 CPU Units

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

- Relationship among motion controls, tasks, and process data communications cycles for the NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units

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

2-7 Overview of CPU Unit Data

The CPU Unit also contains settings, the user program, present values, and other data. The main data is described in the following table.
For all data on the CPU Unit, refer to *A-9 Attributes of CPU Unit Data* on page A-241.

Type of data			Description
Settings	**EtherCAT Configuration**	**EtherCAT Slave Configuration**	This is information on the EtherCAT slave configuration.
		EtherCAT Master Settings	The EtherCAT Master Settings contain parameter settings for the EtherCAT Master Function Module, such as the communications cycle.
	Unit Configuration and Unit Setup		The Unit Configuration and Unit Setup contain information on the Unit configuration that enables the CPU Unit to recognize the Units, and the initial settings of Units.
	I/O Map		The I/O Map contains assignment information between the variables and the I/O ports that are automatically created based on the Unit configuration.
	Controller Setup	**Operation Settings**	The Operation Settings include the Startup Mode setting, Security Settings, and System Service Monitoring Settings.
		Built-in EtherNet/IP Port Settings	The Built-in EtherNet/IP Port Settings contain the following settings: TCP/IP settings, Ethernet settings, DHCP settings, DNS settings, FTP settings, NTP settings, and SNMP settings
	Motion Control Setup		The Motion Control Setup consists of settings for Axis Variables and Axes Group Variables for *axis and axes groups*, and motion control parameter settings.
	Cam Data Settings		The cam data includes cam tables that consist of phase/displacement data for use in cam operation for motion control instructions.
	Event Settings		These settings are for user-defined errors and user-defined information.
	Task Settings		The Task Settings contain settings for the task types, number of tasks, task execution conditions, task names, programs executed in the task, and other task settings.
	Data Trace Settings		The Data Trace Settings include settings for trigger conditions.
	Tag Data Link Tables		The Tag Data Link Tables contain the tag data link settings for EtherNet/IP.
	Controller Name		The Controller name is the name of the CPU Unit.
	Operation Authority Verification		This data contains the operation authority passwords to perform Sysmac Studio operations for the CPU Unit.
	User Authentication		This function authenticates each user when Sysmac Studio is going online with the Controller and restricts operation according to the user's privileges.
	Built-in Clock	**Set Time**	This is the time information that is used inside the CPU Unit.
		Time Zone Setting	This is the time zone that is set for the clock in the CPU Unit.

Type of data			Description
User Program	**POUs (program organization units)**		These are the definitions of the programs, functions, and function blocks. The local variable tables and the initial values of the variables are also included.
	Data	**Data Types**	This data contains the definitions of the data types.
		Global Variables	This data gives the attribute information of the global variables. It includes the Initial Value and Retain attributes.
Present Values	**Values of Variables**		This data contains the values of the variables.
	Contents of memory used for CJ-series Units*1		These are the values of the CIO, Working, Holding, DM, and EM Areas in the memory for CJ-series Units.
Other Data	**Event Logs**		The event logs include the error log for the Controller, and logs of events other than errors, such as when the power supply was turned ON and OFF and when operation started.
	Absolute Encoder Home Offsets		This data is used to restore the actual position of a Servo Drive with an absolute encoder in motion control. The offset is the difference between the command position after homing and the absolute data that is read from the absolute encoder.

*1. You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

2-8 Operation for CPU Unit Status

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

2-8-1 CPU Unit Status

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

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

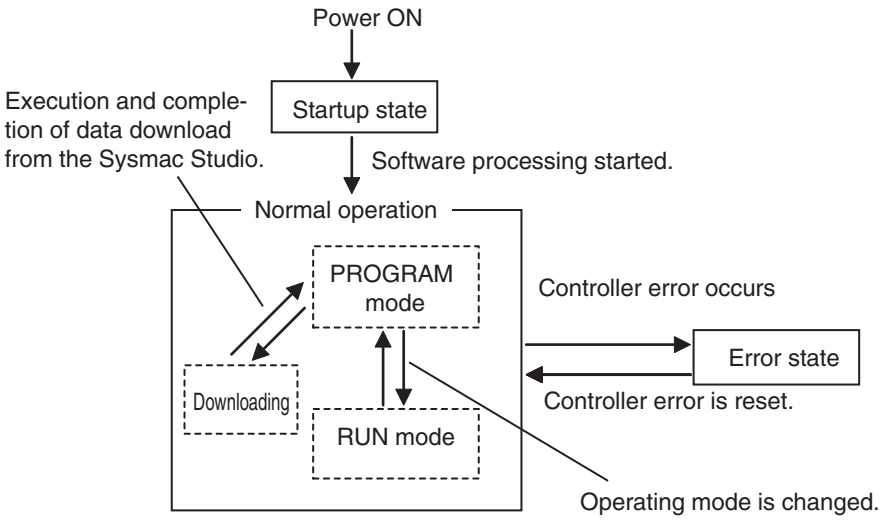
The normal operation state has these three states for operation: PROGRAM mode, RUN mode, and downloading. A CPU Unit in the normal operation state changes to the other states due to user interaction. This state is defined as follows:

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 *2-8-3 Operating Modes* on page 2-43 for details on PROGRAM mode and RUN mode.

● CPU Unit Status

The CPU Unit enters the startup state after the power supply is turned ON. About 10 to 20 seconds after the CPU Unit enters the startup state, software processing begins and the CPU Unit changes to normal operation. If a Controller error occurs during normal operation, the CPU Unit changes to the error state. When you reset the Controller error, the CPU Unit returns to normal operation.
When the CPU Unit changes from startup state to normal operation, it will change to the operating mode that you specify in the Controller Setup. You can set the operating mode at startup to PROGRAM mode or RUN mode. Thereafter, changing the operating mode causes the CPU Unit to change between PROGRAM mode and RUN mode. If you download data from the Sysmac Studio during PROGRAM mode, the CPU Unit will change to the downloading state. The CPU Unit will return to PROGRAM mode when the download is completed.



Additional Information

- You can check the operating status of the CPU Unit with the status indicators on the front panel of the CPU Unit. Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for troubleshooting procedures using the status indicators.
- Refer to *A-9 Attributes of CPU Unit Data* on page A-241 for information on data operations when the CPU Unit status changes.
- Refer to *6-3-9 Changes to Variables for Status Changes* on page 6-69 for the values that variables take when the status of the CPU Unit changes.

2-8-2 Operation for CPU Unit Status

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

Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for information on the *Non-fatal Errors*.

CPU Unit processing	Operation during start-up	Operation during normal operation		
		PROGRAM mode	RUN mode	Downloading
User program	Stopped.	Stopped.	Executed.	Stopped.
I/O refreshing for Ether-CAT slaves	Stopped.	Executed.		EtherCAT communications changes to safe-operational state. *1*2
I/O refreshing for NX Units on the CPU Unit, X Bus Units, built-in I/O, or Analog I/O Option Boards	Stopped.	Executed.		Stopped. *2
I/O refreshing for CJ-series Units	Stopped.	Executed.		
External communica-tions	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 6-72 for the device output hold configurations. A CPU Unit with unit version 1.13 or later and Sysmac Studio version 1.17 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
Outputs from NX Units on the CPU Unit	Changed to the initial values of Units.	The outputs have the values of the device variables for NX Units.		Controlled by the Unit settings. *3
Outputs from X Bus Units	Changed to the initial values of Units.	The outputs have the values of the device variables for X Bus Units.		Stopped I/O refreshing. *3
Outputs from built-in I/O	Turned OFF.	The outputs have the values of the device variables for built-in I/O.		Turned OFF. *3
Outputs from Analog I/O Option Boards	Turned OFF.	The outputs have the values of the device variables for Option Boards.		Turned OFF. *3
Outputs from CJ-series Basic Output Units	Turned OFF.	The outputs have the values of the device variables for CJ-series Units. *3*4		

*1. Refer to the manuals 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 variable. Refer to *6-3-9 Changes to Variables for Status Changes* on page 6-69 for details.
A CPU Unit with unit version 1.13 or later and Sysmac Studio version 1.17 or higher are required to use the *_DeviceOutHoldCfg* (Device Output Hold Configuration) system-defined variable.

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

Refer to *6-3-8 Variable Attributes* on page 6-60 for information on the Initial Value attribute for variables.

Refer to *Device Output Hold Configurations* on page 6-72 for details on the device output hold configurations.

Precautions for Correct Use

- You can use NX Units only on the NX502, NX102, and NX1P2 CPU Units.
- You can use the built-in I/Os and Analog I/O Option Boards only with the NX1P2 CPU Units.
- You can use CJ-series Units only on the NJ-series CPU Units.
- You can use X Bus Units only on the NX502 CPU Units.

Additional Information

Servo Drive Response to Changes in Operating Mode
If the operating mode changes from RUN to PROGRAM mode during a motion control operation, the axes will decelerate to a stop at the maximum deceleration rate.
Changing the Operating Mode during Initialization of EtherCAT Slaves
You can change the operating mode of the CPU Unit to RUN mode while EtherCAT slaves initialization is in progress. If you do, provide programming to confirm that communications are established before you attempt to use slave data in control operations. Your program can use the *_EC_PDSlavTbl* (Process Data Communicating Slave Table) system-defined variable to see if the process data inputs and outputs are valid for all of the slaves.

2-8-3 Operating Modes

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

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

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

Additional Information

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

Operations You Can Perform 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 or memory used for CJ-series Units*1	Possible.	
HMI	Changing the values of variables or memory used for CJ-series Units*1	Possible.	

*1. You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

Retention of Variable Values during Changes in Operating Mode

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

Retain attribute of variable	Value 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 6-66.

Refer to *6-3-9 Changes to Variables for Status Changes* on page 6-69 for the values that variables take when the status of the CPU Unit changes.

Setting and Changing the Operating Mode

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

● Operating Mode Setting After the Power Supply Is Turned ON

When the CPU Unit starts operating after the power supply is turned ON, the CPU Unit operates in the operating mode that you set as the Startup Mode. Specify RUN mode or PROGRAM mode in the *Startup Mode* setting in the *Operation Settings* in the Controller Setup. Refer to *Section 4 Controller Setup* on page 4-1 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 form **Controller** - **Operating Mode** on the menu bar.

Precautions for Safe Use

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

Checking the Operating Mode

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

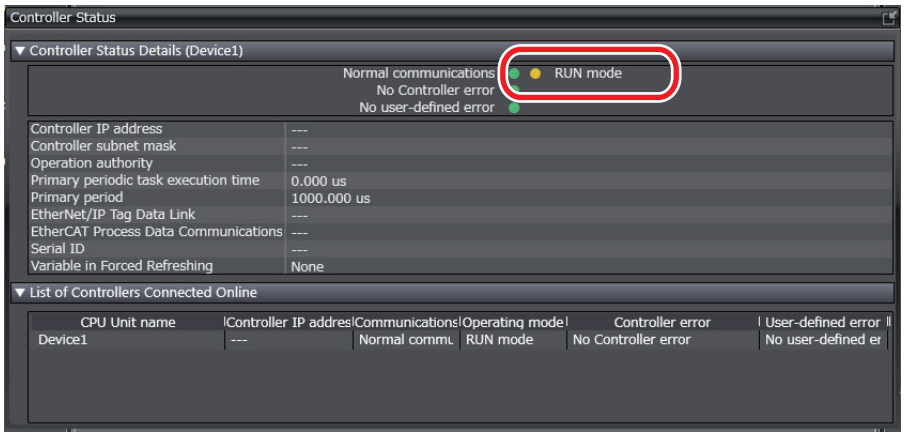
● Checking the RUN Indicator

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

RUN indicator status	Operation 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 Panel indicates that the CPU Unit is in RUN mode.



Additional Information

With an NX701 CPU Unit or NJ-series CPU Unit, if you want to output a signal when the CPU Unit is in RUN mode, use the RUN output on the Power Supply Unit. Refer to the *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)* and the *NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)* for details on the RUN output on the Power Supply Unit.

3

I/O Ports, Slave Configuration, and Unit Configuration

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

3-1 Procedure to Create the Slave and Unit Configurations 3-2

3-2 Creating and Comparing the Slave and Unit Configurations..... 3-5

3-2-1 Creating the EtherCAT Slave Configuration 3-5
3-2-2 Creating the Unit Configuration 3-6
3-2-3 Verifying the Unit Configuration 3-6

3-3 I/O Ports and Device Variables 3-8

3-3-1 I/O Ports 3-8
3-3-2 I/O Port Names 3-9
3-3-3 Device Variables 3-11

3-4 Allocating Variables to Units 3-14

3-4-1 Procedure to Assign Variables to Units 3-14
3-4-2 Using Variables Assigned to Units 3-15

3-5 Creating the Axes and Assigning Them to the Servo Drives/ Encoder Input Slaves/NX Units 3-19

3-5-1 Introduction 3-19
3-5-2 Axis Variables and Axes Group Variables 3-19
3-5-3 Creating and Using Axes and Axis Variables 3-21

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

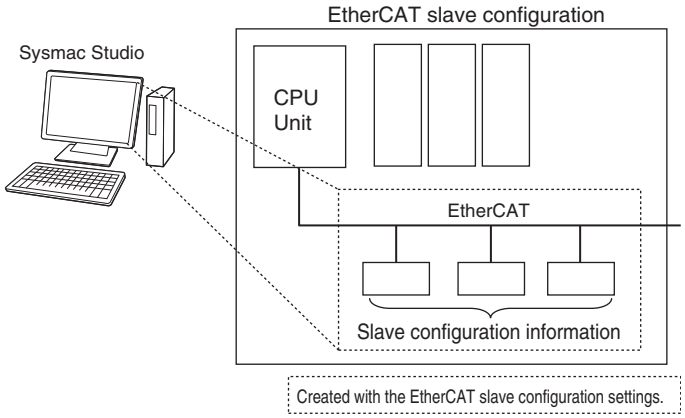
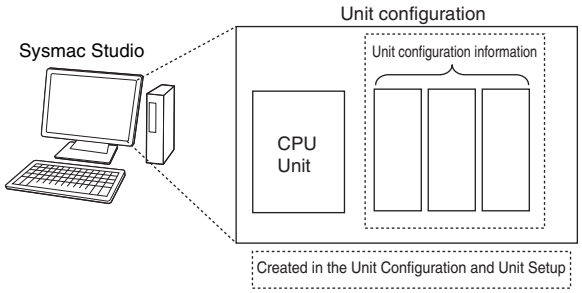
Step 2. Software Setups and Programming	
	Step 2-1 Slave and Unit Configurations
	Step 2-2 Controller Setup
	Step 2-3 Programming
	Step 2-4 Offline Debugging

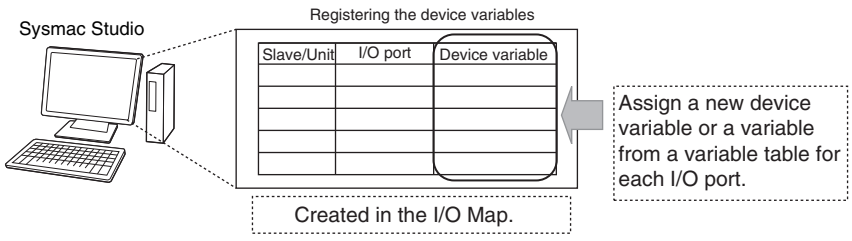
Step 3. Mounting and Setting Hardware

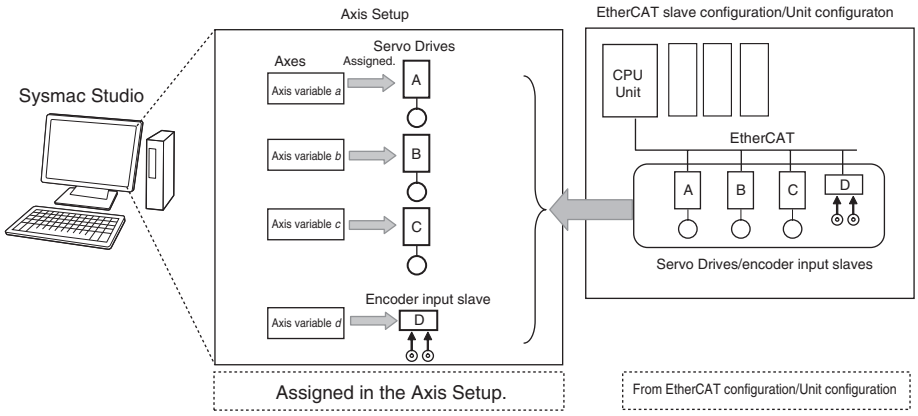
Step 4. Wiring

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

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series* on page 1-23 for details.

Step 1 Create the EtherCAT slave configuration (if EtherCAT is used) and the Unit configuration (if NX Units, X Bus Units, or CJ-series Units are used).	Reference
• Create the EtherCAT slave configuration.  • Create the Unit configuration.  	*3-2-1 Creating the EtherCAT Slave Configuration* on page 3-5 *3-2-2 Creating the Unit Configuration* on page 3-6

Step 2 Assign Device Variables to I/O ports.	Reference
• Register the device variables.  	*2-3-1 Type of Variables* on page 2-13 *3-3 I/O Ports and Device Variables* on page 3-8

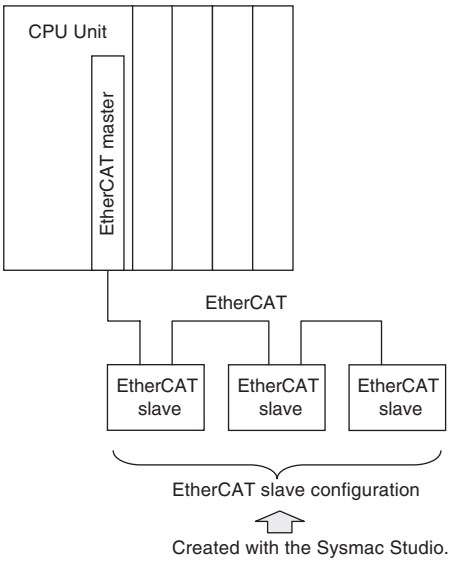
Step 3 Create the axes and assigning them to the slaves or Units (if motion control is used).	Reference
1. Create the axes. 2. Assign the axes to the slaves or Units in the EtherCAT configuration or Unit configuration.  	*3-5 Creating the Axes and Assigning Them to the Servo Drives/Encoder Input Slaves/NX Units* on page 3-19

3-2 Creating and Comparing the Slave and Unit Configurations

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

3-2-1 Creating the EtherCAT Slave Configuration

In the EtherCAT Tab Page of the Sysmac Studio, create the EtherCAT slave configuration that is recognized as “correct” by the CPU Unit.



The I/O ports are automatically registered for the slaves in the configuration. Later, the user assigns device variables to the I/O ports.
You can specify device variables in the user program to access the slaves.
Refer to *EtherCAT Configuration and Settings* in the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for specific procedures to create the EtherCAT slave configuration.

Additional Information

- If you connect EtherCAT Slave Terminals, create the EtherCAT slave configuration, create the Slave Terminal configuration, and set the operation settings. Refer to the *NX-series EtherCAT Coupler Units User’s Manual* (Cat. No. W519) for information on the Slave Terminal configuration and operation settings.
- If you use built-in I/O or Option Boards with an NX1P2 CPU Unit, make the configuration settings and other settings in the Controller Setup.

Version Information

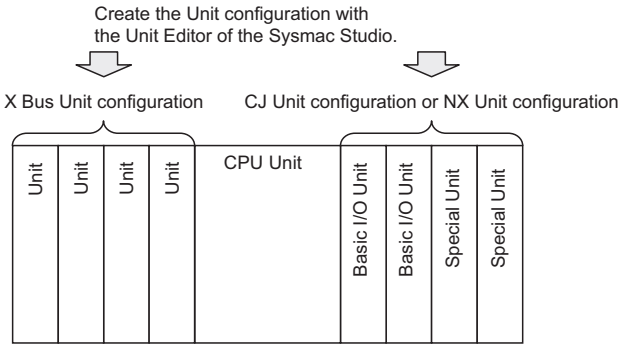
A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use EtherCAT Slave Terminals.

Additional Information

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

3-2-2 Creating the Unit Configuration

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



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

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

Precautions for Correct Use

- You can start the Controller without creating the *Unit configuration*. However, if you do so, the I/O ports and device variables will not be registered automatically, so you will not be able to access the Units from the user program.
- Create the *Unit Configuration* to use NX Units with an NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, to use X Bus Units with an NX502 CPU Unit, or to use CJ-series Units with an NJ-series CPU Unit.
  The NX701 CPU Units do not have the *Unit configuration* because any Units are not used.

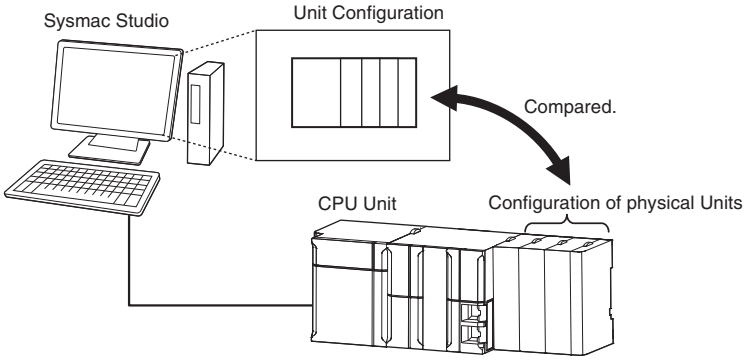
3-2-3 Verifying the Unit Configuration

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

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

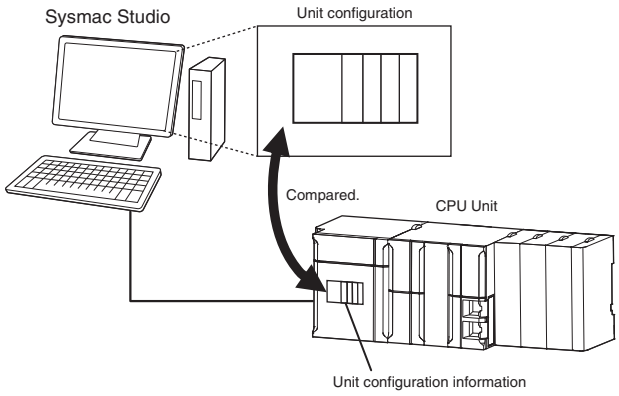
You can verify if the Unit configuration on the Sysmac Studio and the physical Unit configuration are the same.

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



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

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



3-3 I/O Ports and Device Variables

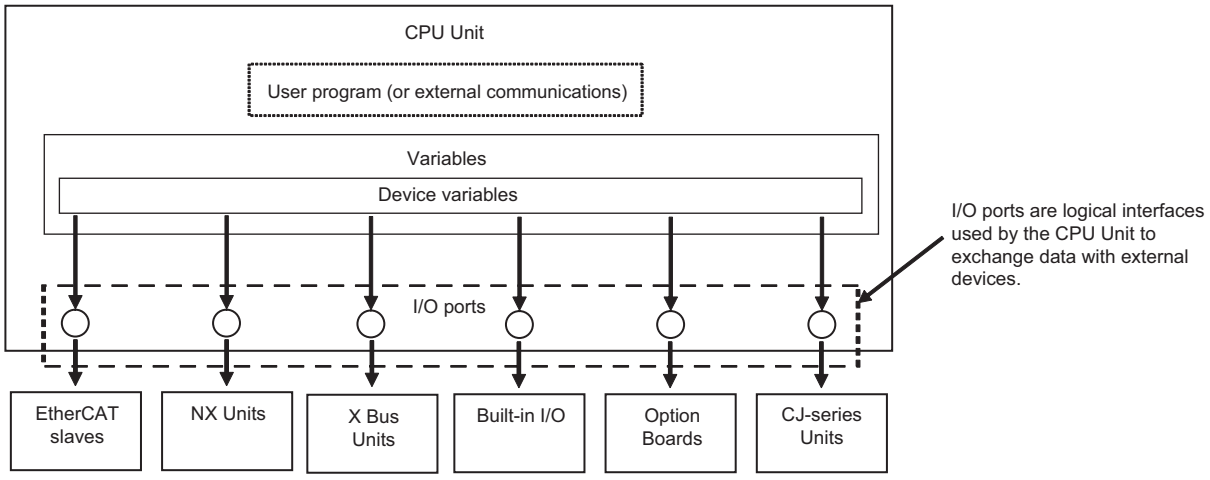
This section describes the I/O ports and device variables that you use to access the EtherCAT slaves, NX Units on the CPU Unit, X Bus Units, built-in I/O, Option Boards, and CJ-series Units of an NJ/NX-series Controller.

Precautions for Correct Use

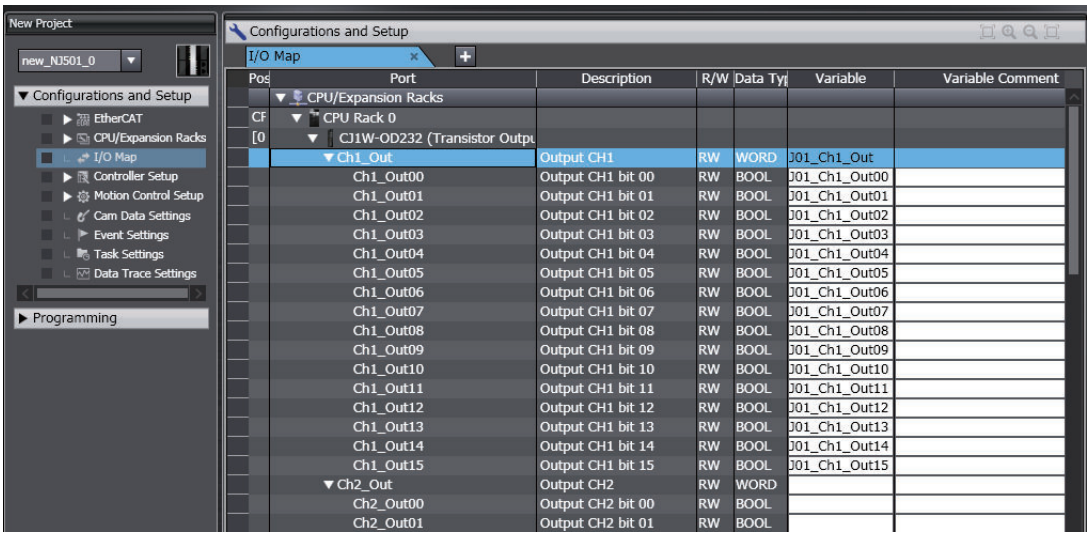
- You can use NX Units only on the NX502, NX102, and NX1P2 CPU Units.
- You can use the built-in I/Os and Analog I/O Option Boards only with the NX1P2 CPU Units.
- You can use CJ-series Units only with NJ-series CPU Units.
- You can use X Bus Units only on the NX502 CPU Units.

3-3-1 I/O Ports

An I/O port is a logical interface that is used by the CPU Unit to exchange data with external devices (slaves and Units), built-in I/O, and Analog I/O Option Boards.
I/O ports are automatically created when you create the slave and Unit configurations and other 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, Unit configuration, and other configurations on the Sysmac Studio, or when you read either of these configurations from the physical Controller from the Sysmac Studio.
You can check the I/O ports that were registered in the I/O Map of the Sysmac Studio.
I/O Map



3-3-2 I/O Port Names

The I/O port names are registered automatically. The I/O port names differ as given below depending on whether the device is an EtherCAT slave, NX Unit on the CPU Unit, X Bus Unit, built-in I/O, Analog I/O Option Board, CJ-series Basic I/O Unit, or CJ-series Special Unit.

EtherCAT Slave Devices

The following I/O port names are used for Remote I/O Terminals that are EtherCAT slave devices.
Example for a 16-point Remote I/O Terminal:

Bit00 to Bit15

For other slaves, all or part of the object names that are defined in the EtherCAT object dictionary are used.
Example for Analog Input Unit:

CH0_input16-bit

Example for R88D-KN50H-ECT:

Position actual value and Digital inputs

NX Unit Devices on the CPU Units

If the device is an NX Unit on the CPU Unit, I/O port names are determined by the model number of the NX Unit and the functionality.
Example for an Digital Input Unit:

Input Bit 00

Example for an Analog Output Unit:

Ch1 Analog Output Value

X Bus Units

If the device is an X Bus Unit, there are two kinds of I/O port names, X Bus Unit common name and unique name determined by the model number of the X Bus Unit and the functionality.

Example for an NX-series EtherNet/IP Unit:
X Bus Unit common: PowerOnHour
Unique to EtherNet/IP Unit: ETN_Errsta

Built-in I/O Devices

If the device is a built-in I/O, the I/O port names are created only for the number of bits for model number of the NX1P2 CPU Unit.
Example for an NX1P2-9024DT CPU Unit:
Input bits: Input Bit 00 to Input Bit 13
Output bits: Output Bit 00 to Output Bit 09

Option Board Devices

If the device is an Option Board, I/O port names are determined by the model number of the Option Board and the functionality.
Example for an Analog Input Option Board:
Ch1 Analog Input Value
Example for an Analog Output Option Board:
Ch1 Analog Output Value

CJ-series Basic I/O Unit Devices

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

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

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

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

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

CJ-series Special Unit Devices

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

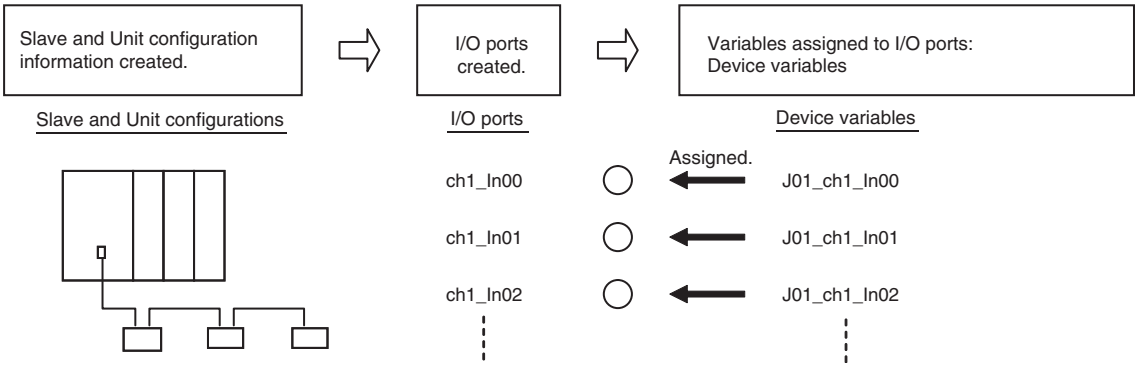
Examples for a CJ1W-AD041-V1 Analog Input Unit:

Ch1_PkHdCmd and Ch1_AveCfg

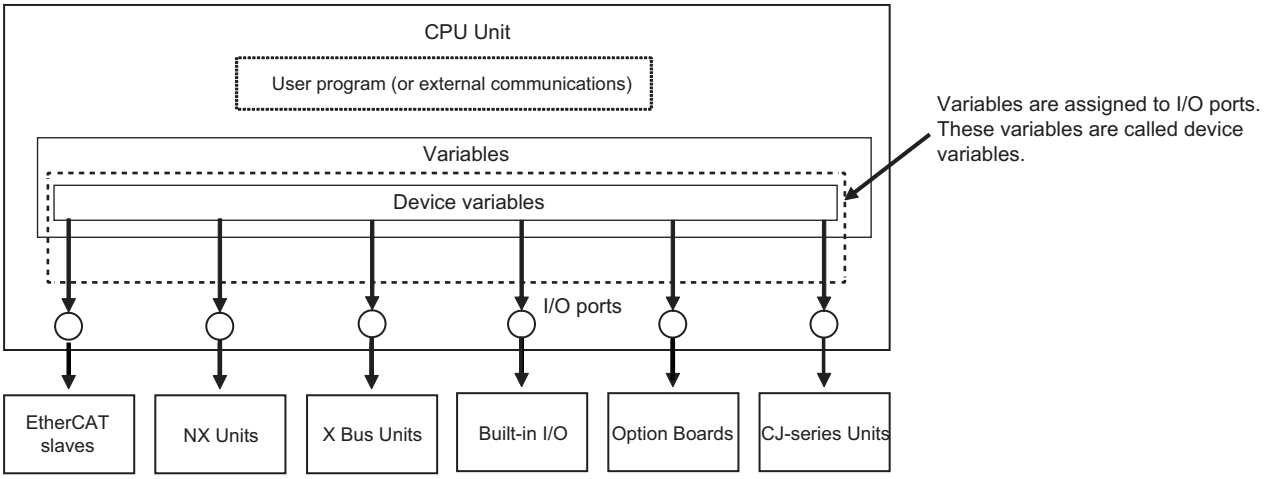
3-3-3 Device Variables

In an NJ/NX-series Controller, external devices are not assigned to specific memory addresses in the CPU Unit. Rather, variables are assigned to the I/O ports.

These variables are called device variables.



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



Refer to *2-3-1 Type of Variables* on page 2-13 for the relationship of device variables to other variables.

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

Device Variable Attributes

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

Attribute	Setting	Changes to settings
Variable Name	Automatically generated variables: [**device_name**] + [**I/O_port_name**] The default **device names** are as follows: • For EtherCAT slaves, an *E* followed by a *sequential number starting from 001*. • For NX Units, an *N* followed by a *sequential number starting from 1*. • For X Bus Units, an *X* followed by a *sequential number starting from 1*. Refer to *3-3-2 I/O Port Names* on page 3-9 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] • NX Units on the CPU Unit: IOBus://unit#[NX_Unit_number]/[I/O_port_name] • X Bus Units: XBus://unit#[Unit_number]/io/[I/O_port_name] • Built-in I/O BuiltInIO://cpu/#0/[I/O_port_name] • Option Boards: BuiltInIO://opt/#[physical_port_number]/[I/O_port_name] • EtherCAT slave ECAT://node#[node_address]/[I/O_port_name] • CJ-series Units IOBus://rack#[rack_number]/slot#[slot_number]/[I/O_port_name]	Not allowed.
Retain	• Device variables for EtherCAT slaves: Not retained. • Device variables for NX Units: Not retained. • Device variables for X Bus Units: Not retained. • Device variables for built-in I/O: Not retained. • Device variables for Option Boards: Not retained. • Device variables for CJ-series Units Assigned to the *Operating Data* (CIO Area): *Not retained* Assigned to the *Setup Data* (DM Area): *Retained*	Not allowed.
Initial Value	None	Allowed.
Constant	None	Allowed.
Network Publish	Do not publish.	Allowed.
Edge	None	Not allowed.

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

Note You can use NX Units on the CPU Unit only with the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
Note You can use device variables for NX Units only with the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
Note You can use X Bus Units only on the NX502 CPU Units.
Note You can use device variables for X Bus Units only with the NX502 CPU Units.
Note You can use the built-in I/Os and Option Boards only with the NX1P2 CPU Units.
Note You can use device variables for built-in I/Os and Option Boards only with the NX1P2 CPU Units.
Note The *physical port number* indicates a physical location for an I/O port. *0* is given for the option board slot 1 and *1* is given for the option board slot 2.

Refer to *6-3-4 Attributes of Variables* on page 6-30 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, NX Units on the CPU Units and X Bus Units are specified by using variables. Therefore, you must assign variables to the Units in advance. After you assign variables to the Units, the connection locations of the Units are automatically updated in the variables even if you change the locations. This means that you do not have to assign variables again every time you change the Unit connection locations.

Version Information

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

Additional Information

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

3-4-1 Procedure to Assign Variables to Units

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

1 Select the following, then double-click or right-click and select **Edit**.

- EtherCAT slaves or EtherCAT Slave Terminals on the EtherCAT network
  Configurations and Setup – **EtherCAT**
- NX Units on the EtherCAT Slave Terminal
  EtherCAT Slave Terminal on the network configuration from the **Configurations and Setup** – **EtherCAT**.
- NX Units on the CPU Unit
  Configurations and Setup − **CPU/Expansion Racks** − **CPU Rack: NX Bus**
- X Bus Units
  Configurations and Setup − **CPU/Expansion Racks** − **CPU Rack: X Bus**

The setting screen for each configuration is displayed. Make configuration settings for the Unit.

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 network, EtherCAT Slave Terminals, NX bus, and X Bus.

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 is added on the I/O Map.

4 Right-click the *Node location information* and select **Create Device Variable**.
The variable name is written to the Variable Field of the *Node location information* port.

For the EtherCAT Slave Terminals or NX Units on the CPU Unit, 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 Communications Coupler Unit*1	UINT
IPAdr	IP address*2	IP address of the Communications Coupler Unit*1	BYTE[5]
UnitNo	Unit number	Unit number of the specified Unit	UDINT
Path	Path*2	Path information to the specified Unit	BYTE[64]
PathLength	Valid path length	Valid path length	USINT

*1. This address is not used for an NX Unit on the CPU Unit.

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

For the X Bus Units, the data type of variables assigned to the Units is _sXBU_ID structure. The details on the _sXBU_ID structure data type are given in the following table.

Variable	Name	Meaning	Data type
User specified	Specified Unit	Specified Unit	_sXBU_ID
UnitNo	Unit number	Unit number of the specified Unit	UINT

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 of a variable, 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.

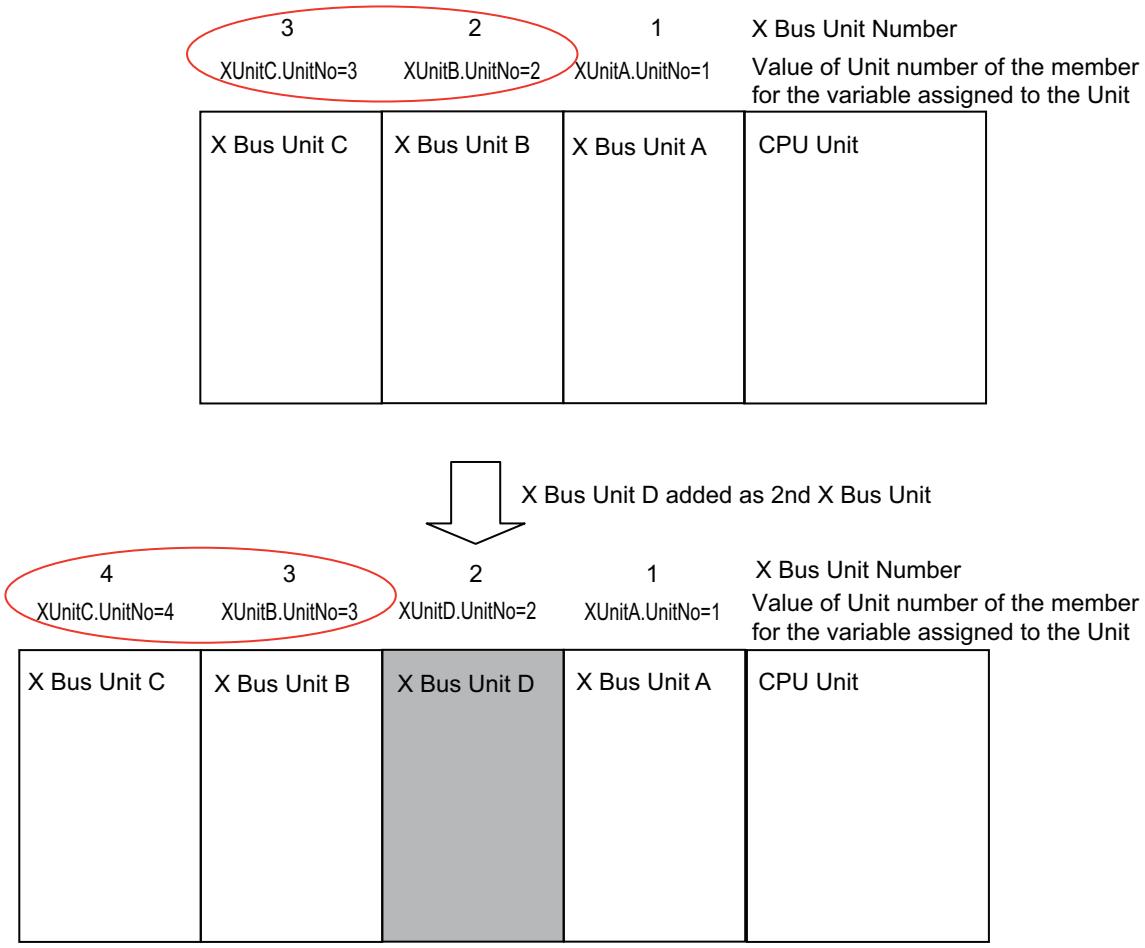
Variable	Name	Meaning	Data type
User specified	Specified slave	Specified slave	_sECAT_ID
NodeAdr	Node address	Node address of the specified slave	UINT

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 as an external variable in the variable table in advance.

Also, the values of Unit number and node address of the member for the variable assigned to the Unit automatically reflect the connected position of the Unit when the variable was registered. Therefore, if the variable assigned to the Unit is used for the variable that specifies the Unit number and node address, you do not need to change the user program even if the connected position changes.

The following figure shows an example of changing the configuration of the X Bus Units.

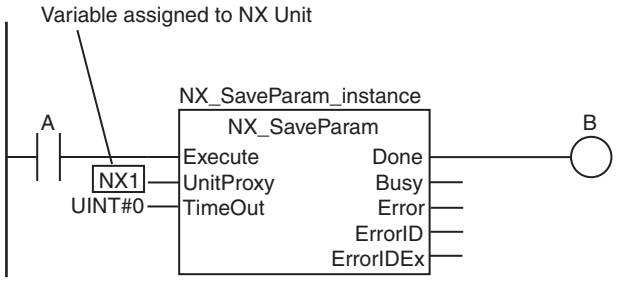


If X Bus Unit D is added to the left of X Bus Unit B, the connected positions of X Bus Unit B and X Bus Unit C will change, and the X Bus Unit numbers will change.
If the variable name assigned to the X Bus Unit B is XUnitB and the variable name assigned to the X Bus Unit C is XUnitC, the XUnitB.UnitNo and XUnitC.UnitNo values of Unit numbers of the member follow to the X Bus Unit numbers of the connected positions that are changed. Therefore, if the variable assigned to the Unit is used in the part that specifies the Unit number of the X Bus Unit in the user program, you do not need to change the user program even if the connected position changes.

Designating Units

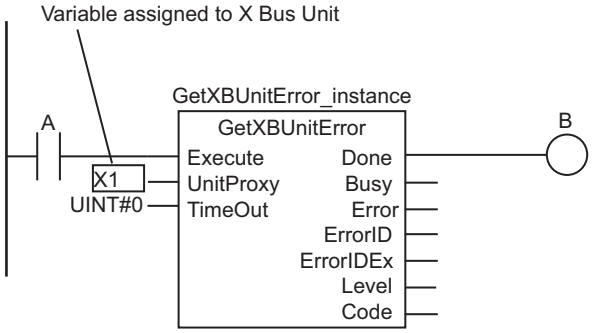
The variables assigned to the Units are passed as parameters to the instructions for which specify the Units.

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 of the _sNXUNIT_ID structure data type.



Example: Executing the GetXBUnitError Instruction

In the following example, the X Bus Unit to which the *X1* variable is assigned is specified when the GetXBUnitError instruction is executed. *X1* is passed to the *UnitProxy* variable of the _sXBU_ID structure data type.



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:=Rda
t);
    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

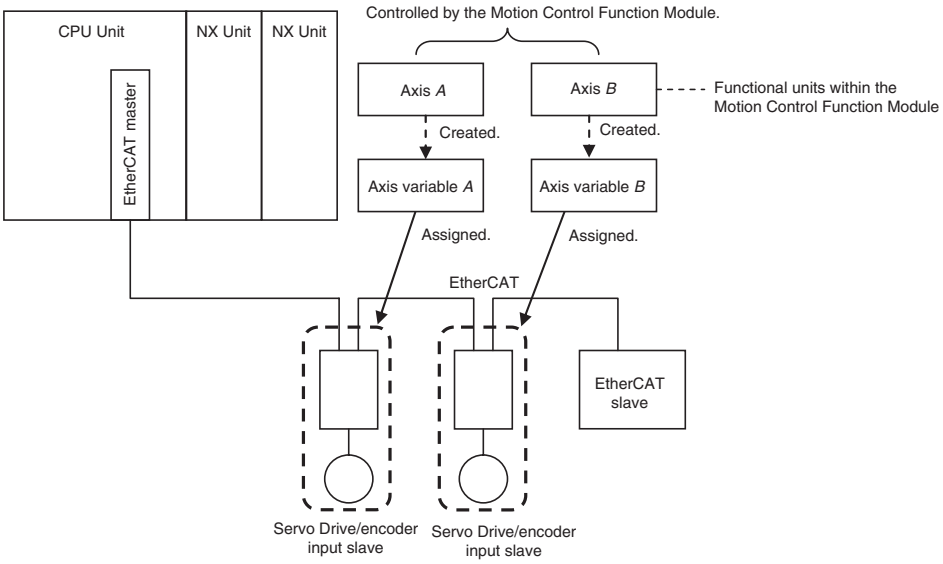
```
FOR i:= 0 TO 2 DO
    NX_ChangeWriteMode_instance[i](Execute:=FALSE);
END_FOR;

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/NX Units

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

3-5-1 Introduction

When you use the Motion Control Function Module for operation with EtherCAT Servo Drive, encoder input slaves, or NX Units, create axes in the Sysmac Studio and define them as EtherCAT servo axes, encoder axes, or NX Units.
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, encoder input slaves, and NX Units.

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) or NX Unit 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.

Type of variable		Application	Device to access	Creation method
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) or NX Units that are assigned to the axes group	Provided by the system.
	Axes Group Variables automatically created when axes groups are created with the Sysmac Studio			You must create an axes group with the Sysmac Studio.

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

• Specifying Axis and Axes Group Variables

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

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

*1. With the NX701 CPU Unit, you can also use *_MC1_AX[***]*, *_MC2_AX[***]*, *_MC1_GRP[***]*, and *_MC2_GRP[***]*. For details, refer to the .

• 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, encoder input slave, or NX Unit.

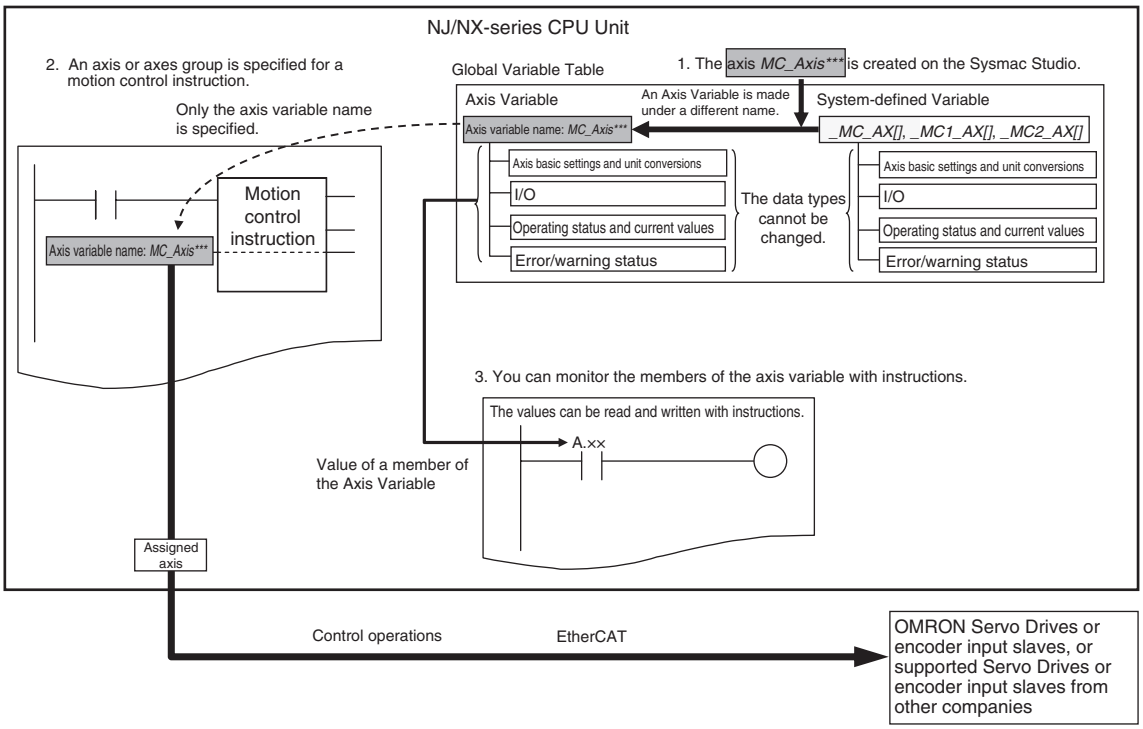
2. Monitoring Axis Variable Members:

   You can use instructions to monitor the actual position, error information, or other information on the Servo Drive, encoder input slaves, and NX Units.

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, supported Servo Drive or encoder input slave from another company, or NX Unit 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, encoder input slave, or NX Unit.



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

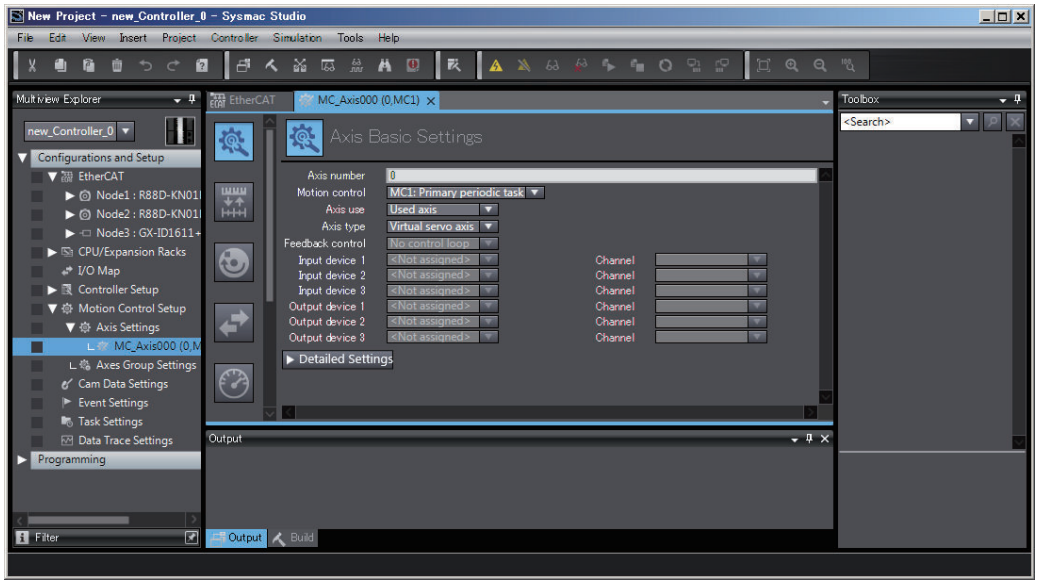
You can create and use axes and Axis Variables as described below.

1 Right-click **Axis Settings** under **Configurations and Setup** − **Motion Control Setup** in the Multiview Explorer and select **Add** − **Axis Settings** from the menu.
If necessary, you can change the axis variable names from the default names of *MC_Axis****.
("***" is incremented from 000 in the order that the axis variables are created.)

2 Assign the axes that you created to Servo Drives or encoder input slaves in the EtherCAT slave configuration of the Sysmac Studio.
Set the Axis Basic Settings from the Sysmac Studio.

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

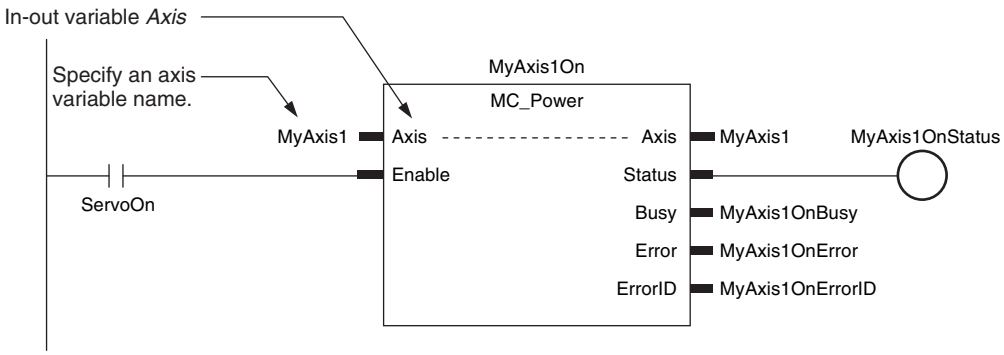
*1. You can select this parameter for NX701 CPU Units.



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

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

Example:



Refer to *3-5-2 Axis Variables and Axes Group Variables* on page 3-19 for details on axis variables.

Precautions for Correct Use

- For an NX701 CPU Unit, use the system-defined variables that correspond to the Motion Control parameter for Axis Settings or Axes Group Settings. A building error occurs if you use the system-defined variables that do not correspond.

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

4

Controller Setup

This section describes the initial settings of the function modules.

4-1 Overview of the Controller Setup 4-2

4-2 Initial Settings for the PLC Function Module 4-4
- 4-2-1 Introduction 4-4
- 4-2-2 Controller Setup 4-4
- 4-2-3 Task Settings 4-7
- 4-2-4 Configuration and Setup for NX Units 4-13
- 4-2-5 Configuration and Setup for X Bus Units 4-16
- 4-2-6 Unit Configuration and Unit Setup for NJ-series CPU Units 4-19

4-3 Initial Settings for NX Units 4-21
- 4-3-1 NX Unit Settings 4-21
- 4-3-2 I/O Allocation Settings 4-22
- 4-3-3 Unit Operation Settings 4-24

4-4 Initial Settings for X Bus Units 4-25
- 4-4-1 X Bus Unit Settings 4-25

4-5 Initial Settings for Special Units 4-26

4-6 Initial Settings for the Motion Control Function Module 4-28
- 4-6-1 Introduction 4-28
- 4-6-2 Setting Methods 4-28

4-7 Initial Settings for the EtherCAT Master Function Module 4-30

4-8 Initial Settings for the EtherNet/IP Function Module 4-31

4-9 Initial Settings for Built-in I/O 4-32

4-10 Initial Settings for Option Boards 4-33

4-11 Memory Settings for CJ-series Units 4-34
- 4-11-1 Setting Procedure 4-34
- 4-11-2 Setting Screen 4-34
- 4-11-3 Settings 4-35

4-1 Overview of the Controller Setup

This section provides an overview of the Controller Setup.
The shaded steps in the overall procedure that is shown below are related to the Controller Setup.

***STEP 1.* Software Design**	
	Step 1-1 Designing I/O and Processing
	Step 1-2 Designing Tasks
	Step 1-3 Designing Programs

***STEP 2.* Software Setups and Programming**	
	Step 2-1 Slave and Unit Configurations
	Step 2-2 Controller Setup
	Step 2-3 Programming
	Step 2-4 Offline Debugging

***STEP 3.* Mounting and Setting Hardware**

***STEP 4.* Wiring**

***STEP 5.* Checking Operation and Starting Operation on the Actual System**

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series* on page 1-23 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* on page 4-4
● **Initial Settings for NX Units:** Unit Configuration and Setup: Initial settings for the NX Units	*4-3 Initial Settings for NX Units* on page 4-21
● **Initial Settings for X Bus Units:** Unit Configuration and Setup: Initial settings for the X Bus Units	*4-4 Initial Settings for X Bus Units* on page 4-25
● **Initial Settings for Special Units:** Unit Configuration and Setup: Initial settings for the Special Units	*4-5 Initial Settings for Special Units* on page 4-26
● **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	*4-6 Initial Settings for the Motion Control Function Module* on page 4-28
● **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	*4-7 Initial Settings for the EtherCAT Master Function Module* on page 4-30
● **Initial Settings for the EtherNet/IP Function Module:** Ethernet Port Setup: EtherNet/IP Port TCP/IP Settings, Ethernet Settings, and other settings	*4-8 Initial Settings for the EtherNet/IP Function Module* on page 4-31

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

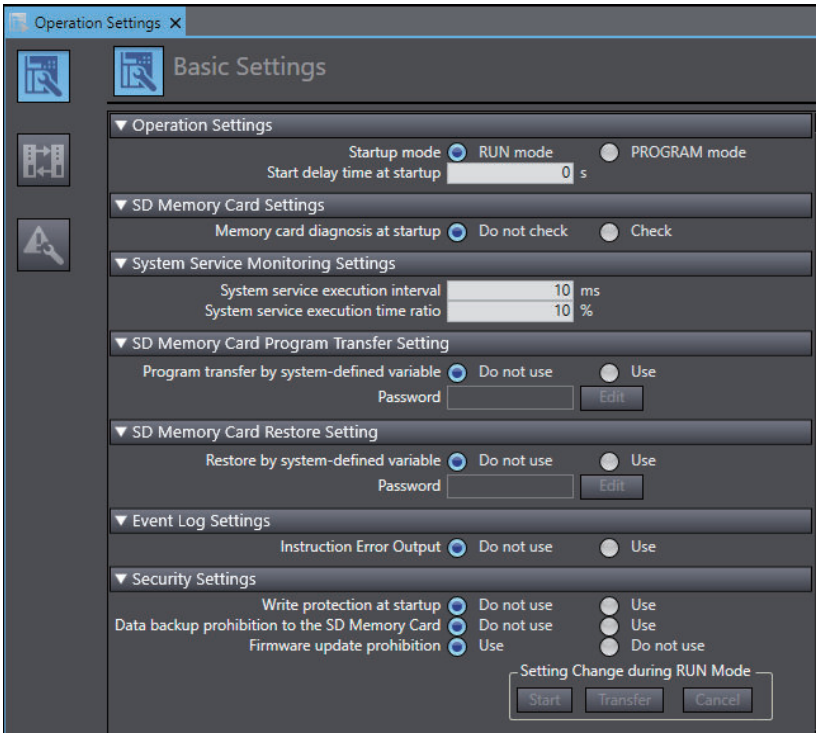
Select **Controller Setup** and **Task Settings** under **Configurations and Setup** on the Sysmac Studio to make these settings.

4-2-2 Controller Setup

Operation Settings Tab Page

● Basic Settings

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



Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
Operation Settings	Startup mode	Sets the CPU Unit's operating mode at startup.	RUN mode PROGRAM mode	RUN mode	When downloaded to CPU Unit	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*3	0 s	When downloaded to CPU Unit	Not allowed.
	Battery-related error detection*4	Sets whether to use battery-related error detection.	Do not use Use	Do not use	When downloaded to CPU Unit	Allowed.
SD Memory Card Settings	Memory card diagnosis at startup	Sets whether to execute self-diagnosis (file system check and restoration) on the inserted SD Memory Card when the power is turned ON.	Do not check Check	Do not check	When downloaded to CPU Unit	Not allowed.
System Service Monitoring Settings *5	System service execution interval [ms]	Sets the interval of system service execution.	10 ms to 1 s	10 ms	When downloaded to CPU Unit	Not allowed.
	System service execution time ratio [%]	Sets the ratio of execution for monitoring system services in relation to overall processing of the CPU Unit.	5% to 50%	10%	When downloaded to CPU Unit	Not allowed.
SD Memory Card Program Transfer Setting*6	Program transfer by system-defined variable	Sets whether to use the program transfer from SD Memory Card.	Use Do not use	Do not use	When downloaded to CPU Unit	Not allowed.
	Password	Sets the password that is used for verification when you execute the program transfer from the SD Memory Card.	8 to 32 single-byte alphanumeric characters (case sensitive)	None	When downloaded to CPU Unit	Not allowed.

Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
SD Memory Card Restore Setting*7	Restore by system-defined variable	Set whether to use the restore of SD Memory Card backups by the system-defined variable.	Use Do not use	Do not use	When downloaded to CPU Unit	Not allowed.
	Password	Sets the password that is used for verification when you execute the restore of SD Memory Card backups by the system-defined variable.	8 to 32 single-byte alphanumeric characters (case sensitive)	None	When downloaded to CPU Unit	Not allowed.
Event Log Settings*8	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 CPU Unit	Not allowed.
Security Settings	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	At power 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 CPU Unit	Allowed.
	Firmware update prohibition*10	Sets whether to use or do not use the firmware update prohibition.	Do not use Use	Use	When downloaded to CPU Unit	Not allowed.

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

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

*3. For the following CPU Unit and Sysmac Studio, *0 to 30s* can be set for the start delay time at startup. Make this setting up to approximately 20 seconds because the time from when the power supply to the CPU Unit is turned ON until the Unit changes to the *normal operation* is affected.

CPU Unit	Sysmac Studio
• The NX102-□□20 CPU Units with unit version 1.35 or later • The NX701-□□□□, NJ501-1□20, NJ501-4320, and NJ101-□□20 CPU Units with unit version 1.23 or later	1.41 or higher
The NX102-□□00, NX1P2-□□□□□□□□, NJ501-1□00, NJ301-1□00, and NJ101-□□00 CPU Units with unit version 1.42 or later	1.43 or higher
The NX502-□□□□ CPU Units with unit version 1.60 or later	1.54 or higher

However, when more than 10 seconds are set, a minor fault level Controller error may occur. If a minor fault level Controller error occurred, reset the error.

*4. These settings are only provided for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

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

*6. A CPU Unit with unit version 1.11 or later and Sysmac Studio version 1.15 or higher are required to use the function.

*7. A CPU Unit with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required to use the function.

*8. A CPU Unit with unit version 1.02 or later and Sysmac Studio version 1.03 or higher are required to use the function.

*9. A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to use the function.

*10. The following CPU Units and Sysmac Studio version 1.53 or higher are required to use the function.
- NJ-series, NX102, NX502, NX1P2 CPU Unit: Version 1.60 or later
- NX701 CPU Unit: Version 1.32 or later

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.

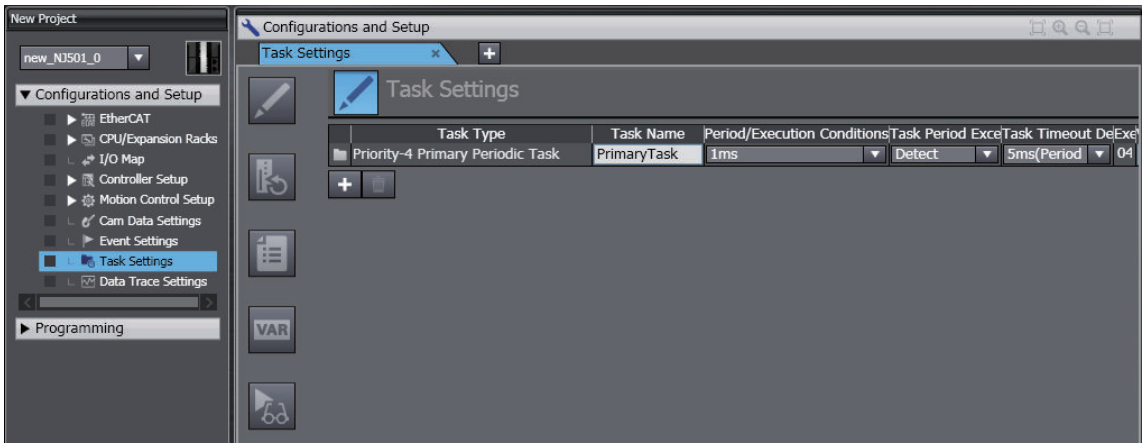
Precautions for Correct Use

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 Periodic task Priority-5 periodic task*1 Periodic task Priority-16 periodic task*2 Periodic task Priority-17 periodic task Periodic task Priority-18 periodic task Event task Priority-8 event task Event task Priority-48 event task	Primary periodic task	When downloaded	Not allowed.
	Execution Priority	Sets the task execution priority.	Automatically set according to the *task type*.	Primary periodic task: 4	When downloaded	Not allowed.
Task Name		Sets the task name.	Text string	Primary periodic task: PrimaryTask Periodic tasks: PriodicTask0 Event tasks: EventTask0	When downloaded	Not allowed.

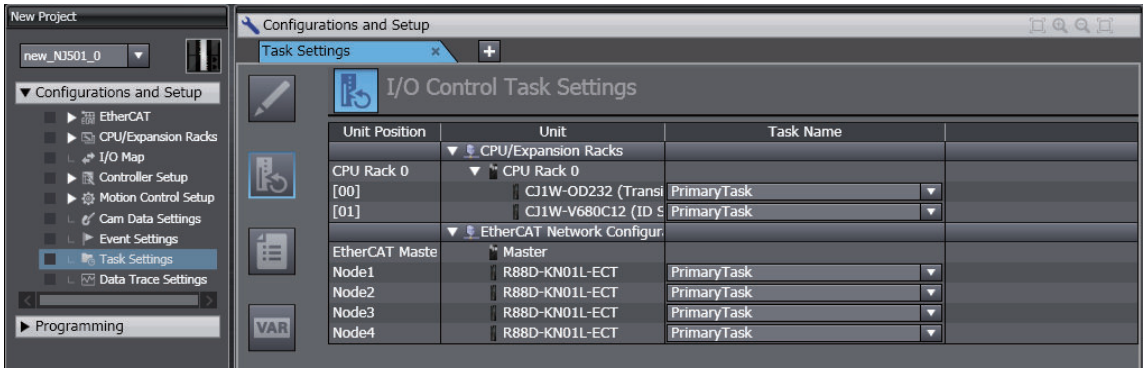
Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
Period/ Execution Conditions		Sets the task period.	Primary periodic task and periodic tasks:Refer to *5-3-1 Specifications of Tasks for NX701 CPU Units* on page 5-11. Refer to *5-4-1 Specifications of Tasks for NX502 CPU Units* on page 5-31. Refer to *5-5-1 Specifications of Tasks for NX102 CPU Units and NX1P2 CPU Units* on page 5-51. Refer to *5-6-1 Specifications of Tasks for NJ-series Controllers* on page 5-68. Event tasks: Executed with an instruction or when a variable expression is satisfied.	Primary periodic task: 1ms *3 Periodic task: 2ms (Priority 5) 10 ms (Priority 16, 17, and 18) Event tasks: Execution with an instruction	When downloaded	Not allowed.
Task Period Exceeded Detection		Sets whether to detect an error when the task period is exceeded.	• Detect (a minor fault level Controller error occurred). • Do not detect (Store and observation level log record.)	Primary periodic task and periodic tasks: Detect Event tasks: Do not detect (cannot be changed).	When downloaded	Not allowed.

Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
Task Time-out 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 x 5 Event tasks: Execution priority of 8: 200 ms Execution priority of 48: 1 s	When downloaded	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	Not allowed.

*1. You can use the priority-5 periodic task only with the NX701 CPU Units.
*2. You cannot use the priority-16 periodic task with the NX102 CPU Unit or NX1P2 CPU Unit.
*3. For the NX102 CPU Unit, NX1P2 CPU Unit, and NJ101 CPU Unit, the default of the primary periodic task is 2 ms.
However, for the NX1P2-9B□□□□ CPU Unit, the default of the primary periodic task is 4 ms.

● I/O Control Task Settings

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

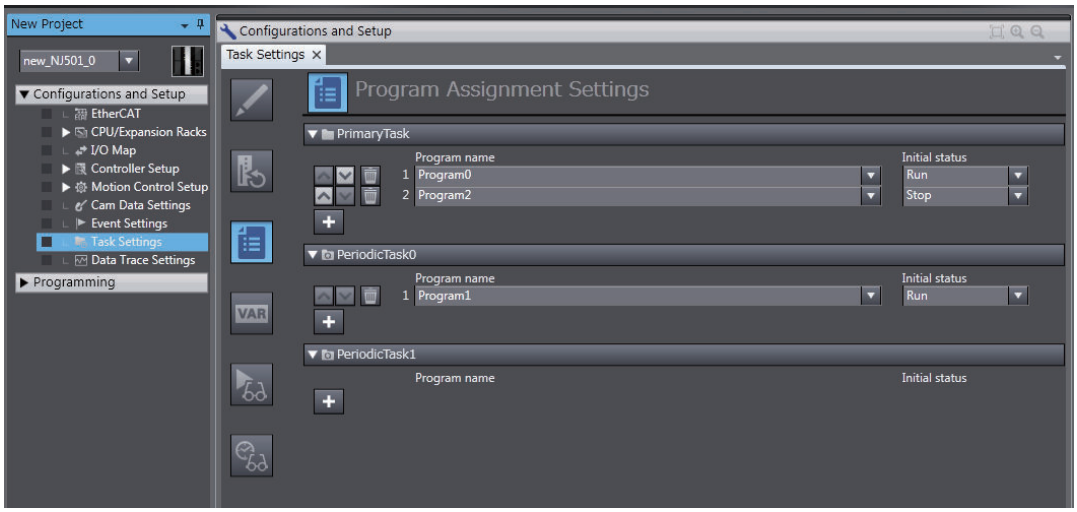


Parameter	Description	Set value	Default	Update timing	Changes in RUN mode
Task Name	Sets the task to use to refresh the specified Units or slaves. *1	PrimaryTask or PeriodicTask	PrimaryTask	When downloaded to CPU Unit	Not allowed.

*1. The NX-series EtherNet/IP Units are not applied for this setting. The timing of refresh execution of inputs and outputs is set to the task for each tag set with the settings for exclusive control of variables in tasks.

● Program Assignment Settings

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



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

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

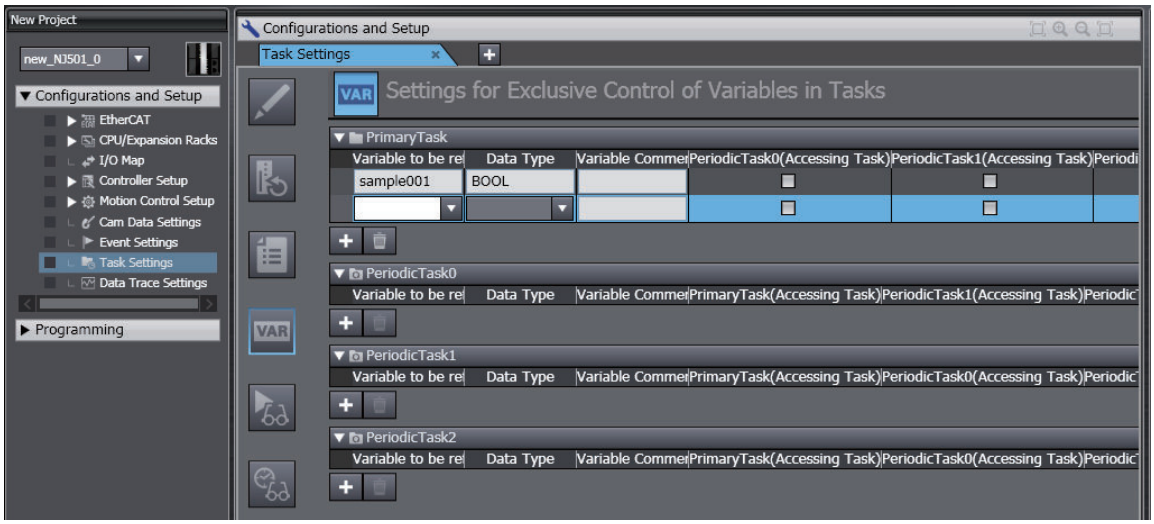
Precautions for Correct Use

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

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

● Settings for Exclusive Control of Variables in Tasks

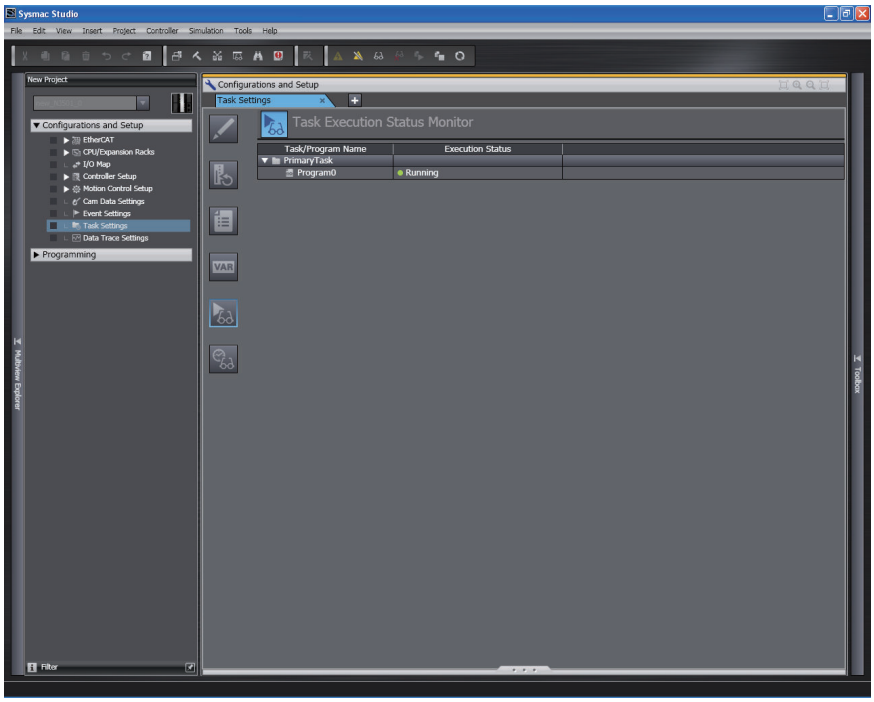
The Settings for Exclusive Control of Variables in Tasks are used to set the tasks that refresh specified global variables and the tasks that access specified global variables.



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

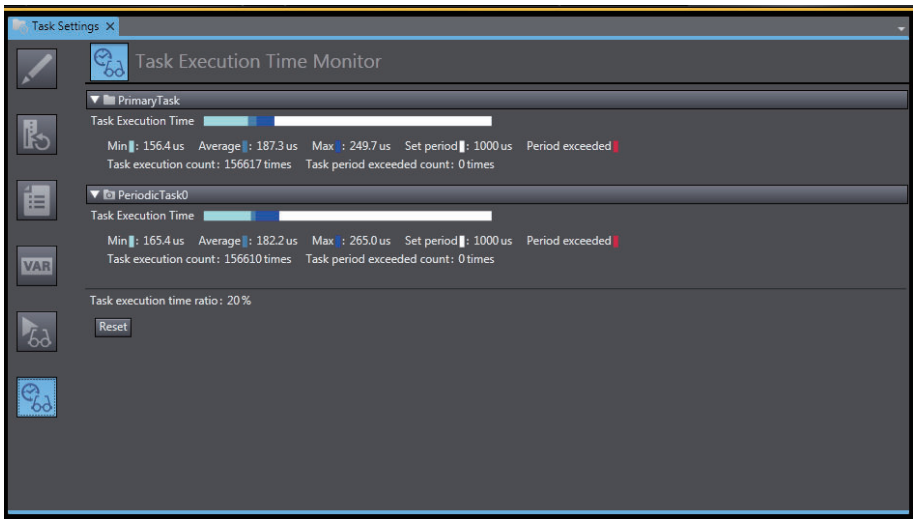
● Task Execution Status Monitor

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



- **Task Execution Time Monitor**

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



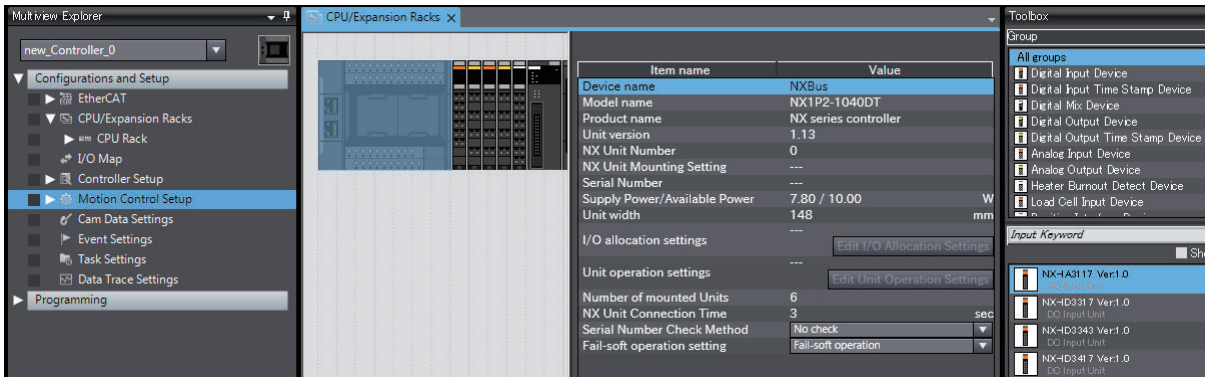
4-2-4 Configuration and Setup for NX Units

This section describes the Unit configurations and settings for the NX Units on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units. These settings are not provided for the NX701 CPU Units and NJ-series CPU Units.

Create the configuration of the NX Units on the CPU Unit and set each Unit.
Configure the settings on the CPU and Expansion Racks Tab Page on the Sysmac Studio.

Unit Configuration Information

The Unit configuration information describes the configuration of the NX Bus Function Module: the number and order of NX Units connected to the NX bus of the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, individual Unit information, and information about the CPU Unit itself.



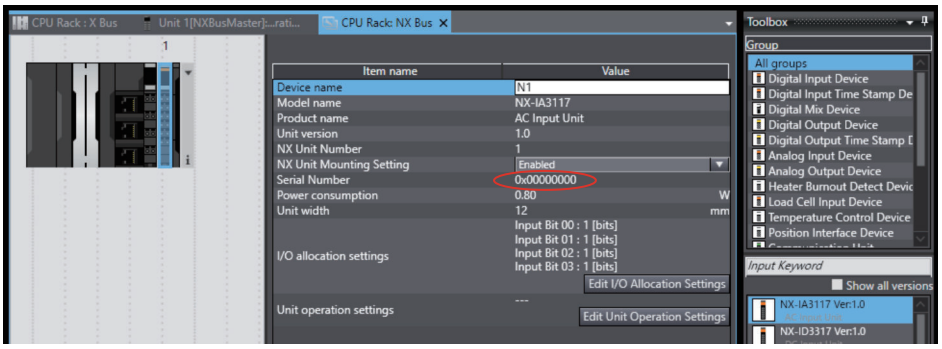
The settable items for the selected CPU Unit are listed below.

Item	Description	Set value	Default
Serial Number Check Method	Set this setting to *Setting = Actual device* to compare the serial numbers of the NX Units at these times: when the power is turned ON and after the CPU Unit is restarted. The serial numbers of the NX Units saved in the Unit configuration information are compared with the serial numbers of the NX Units actually connected. *1 If differences are found, a Unit Configuration Verification Error will occur.	No check Setting = Actual device	No check
Fail-soft Operation Setting	Set whether to use fail-soft operation for the NX Bus Function Module. Select *Fail-soft operation* to perform fail-soft operation. Refer to *8-2-5 Fail-soft Operation for NX Units on the CPU Unit* on page 8-16 for details on fail-soft operation.	Fail-soft operation Stop	Fail-soft operation

*1. If this setting is set to *Setting = Actual device* and you replace an NX Unit on the CPU Unit, a Unit Configuration Verification Error will occur. A Unit Configuration Verification Error will also occur if you swap the mounting position of two Units of the same model.
If it becomes necessary to replace an NX Unit, or swap the mounting positions of two Units of the same model while this setting is set to *Setting = Actual device*, you must correct the Unit configuration information and download it to the CPU Unit. Set this parameter to *Setting = Actual device* if strict management of the equipment configuration is required.

Additional Information

The serial number of the NX Unit set in the Unit configuration information is the **Serial Number** in the CPU Rack: NX Bus Tab Page shown below.



After the Unit configuration information is created, read the serial number of the NX Unit from the actual Unit configuration and set it.
Refer to *CPU/Expansion Rack Configuration and Setup* in the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for specific procedures to set the serial number.

Precautions in Changing the Unit Configuration

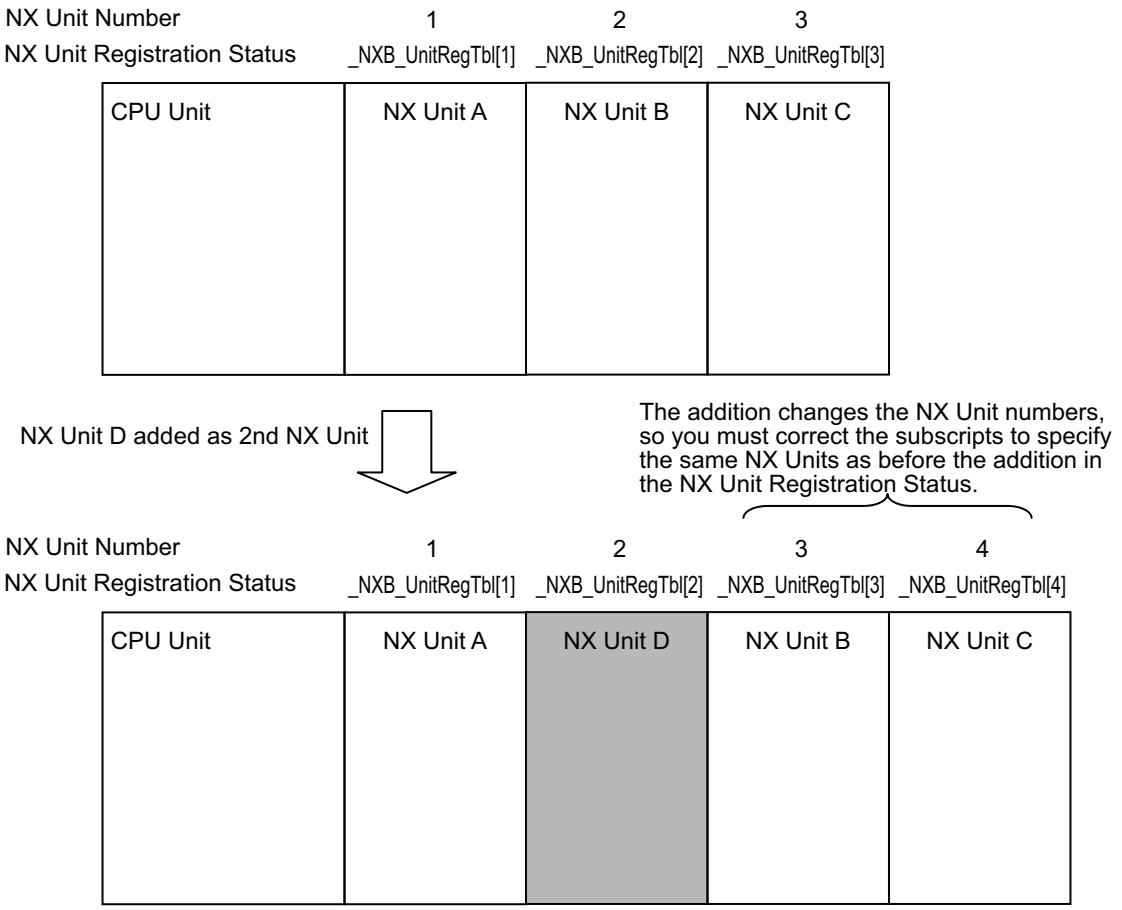
This section provides precautions that apply when you change the NX Unit configuration on an NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit.

● I/O Data That Require Specification of NX Unit Numbers

You must specify the NX Unit number to access some system-defined variables in I/O data. If you change the Unit configuration on the CPU Unit, you must correct the specified NX Unit numbers, such as those in programs.

_NXB_UnitRegTbl (NX Unit Registration Status) and *_NXB_UnitErrFlagTbl* (NX Unit Error Status) are examples of the system-defined variables to specify the NX Unit number. The system-defined variable for this status information uses BOOL arrays, and the NX Unit number is specified as the subscript.

For example, if an NX Unit D is added to the CPU Unit, the NX Unit numbers of all NX Units to the right of the new NX Unit will change.



If you specify the subscripts of arrays directly with numbers, the subscripts in the program must be corrected to specify the same NX Unit as before the addition.

If you use _sNXUNIT_ID data type variables that are assigned to the Units to specify the array subscripts, you do not need to correct the program even if the Unit configuration changes. The NX Unit number of a Unit is stored in the *UnitNo* member of the _sNXUNIT_ID structure variable.
For example, if the _sNXUNIT_ID variable *NXUnitB* is assigned to NX Unit B in the above figure, the program would not need to be corrected even if the Unit configuration changed as long as the array subscript is specified with *_NXB_UnitRegTbl[NXUnitB.UnitNo]*.

Additional Information

NX Unit Mounting Settings
If an NX Unit is removed or if you know in advance that an NX Unit is planned to be added, you can use the NX Unit mounting settings to prevent the NX Unit numbers from changing.
Refer to *8-2-2 Mounting Settings of NX Units on the CPU Unit* on page 8-11 for details on the NX Unit mounting settings.

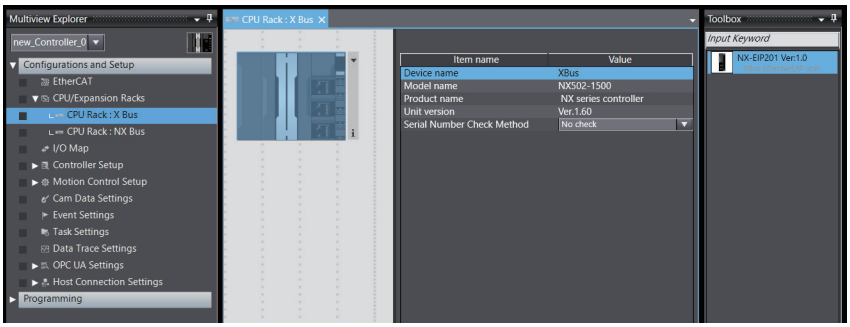
4-2-5 Configuration and Setup for X Bus Units

This section describes the Unit configurations and settings for the X Bus Units on the NX502 CPU Units. These settings are not provided for the NX701 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units.

Create the X Bus Unit configuration and set up each Unit.
Configure the settings on the CPU and Expansion Racks Tab Page on the Sysmac Studio.

Unit Configuration Information

The Unit configuration information describes the configuration of the X Bus Function Module: the number and order of X Bus Units connected to the X Bus of the NX502 CPU Unit, individual Unit information, and information about the CPU Unit itself.



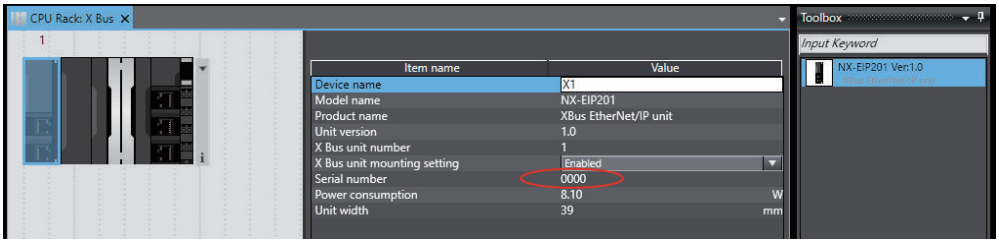
The settable items for the selected CPU Unit are listed below.

Item	Description	Set value	Default
Serial Number Check Method	Set this setting whether to compare the serial numbers of the X Bus Units at these times: when the power is turned ON and after the CPU Unit is restarted. The serial numbers of the X Bus Units saved in the Unit configuration information are compared with the actual serial numbers of the X Bus Units. *1	No check Setting = Actual device	No check

*1. If this setting is set to *Setting = Actual device* and you replace an X Bus Unit, a Unit Configuration Verification Error will occur. A Unit Configuration Verification Error will also occur if you swap the mounting position of two Units of the same model.
If it becomes necessary to replace an X Bus Unit, or swap the mounting positions of two Units of the same model while this setting is set to *Setting = Actual device*, you must correct the Unit configuration information and download it to the CPU Unit. Set this parameter to *Setting = Actual device* if strict management of the equipment configuration is required.

Additional Information

The serial number of the X Bus Unit set in the Unit configuration information is the **Serial number** in the CPU Rack: X Bus Tab Page shown below.



After the Unit configuration information is created, read the serial number of the X Bus Unit from the actual Unit configuration and set it.
Refer to *CPU/Expansion Rack Configuration and Setup* in the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for specific procedures to set the serial number.

Precautions in Changing the Unit Configuration

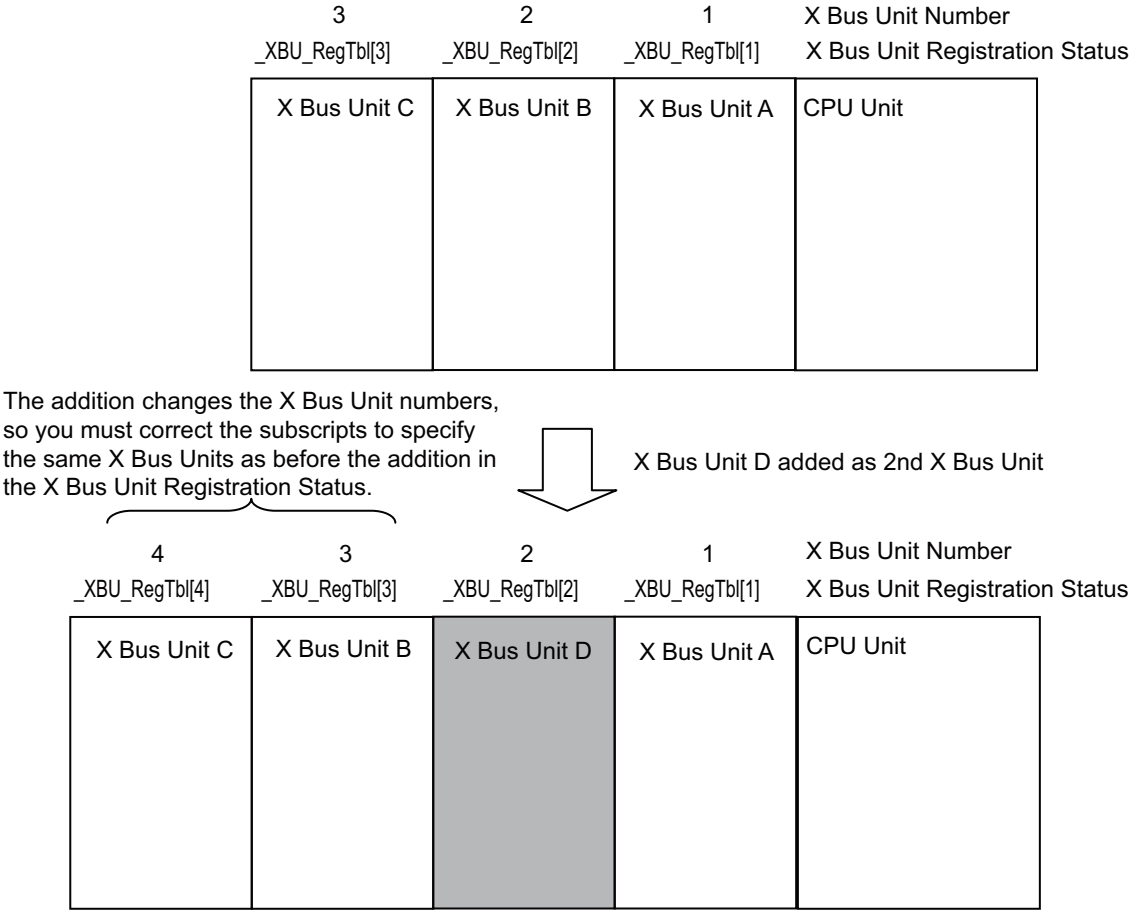
This section provides precautions that apply when you change the X Bus Unit configuration.

● I/O Data That Specifies the X Bus Unit Number

You must specify the X Bus Unit number to access some system-defined variables in I/O data. If you change the X Bus Unit configuration on the CPU Unit, you must correct the specified Unit numbers, such as those in programs.

System-defined variables that specify the X Bus Unit number include, for example, *_XBU_RegTbl* (X Bus Unit Registration Status). The system-defined variable for this status information uses BOOL arrays, and the X Bus Unit number is specified as the subscript.

For example, if an X Bus Unit D is added to the CPU Unit, the X Bus Unit numbers of all X Bus Units to the left of the new X Bus Unit will change.



If you specify the subscripts of arrays directly with numbers, the subscripts in the program must be corrected to specify the same X Bus Unit.

If you use _sXBU_ID data type variables that are assigned to the Units to specify the array subscripts, you do not need to correct the program even if the Unit configuration changes. The X Bus Unit number of a Unit is stored in the *UnitNo* member of the _sXBU_ID structure variable.
For example, if the _sXBU_ID variable *XUnitB* is assigned to X Bus Unit B in the above figure, the program would not need to be corrected even if the Unit configuration changed as long as the array subscript is specified with *_XBU_RegTbl[XUnitB.UnitNo]*.

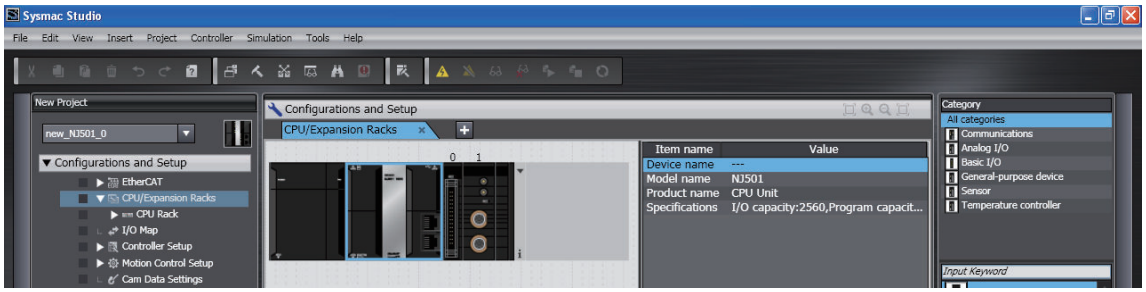
Additional Information

X Bus Unit Mounting Settings
If an X Bus Unit is removed or if you know in advance that an X Bus Unit is planned to be added, you can use the X Bus Unit mounting settings to prevent the X Bus Unit numbers from changing.
Refer to *8-3-3 X Bus Unit Mounting Settings* on page 8-24 for details on the X Bus Unit mounting settings.

4-2-6 Unit Configuration and Unit Setup for NJ-series CPU Units

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

● Unit Information



● Settings for All Units

Set the device names.

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

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

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

Additional Information

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

● Special Units

Set the unit numbers of the Special Units.

Precautions for Correct Use

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

● Basic I/O Units

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

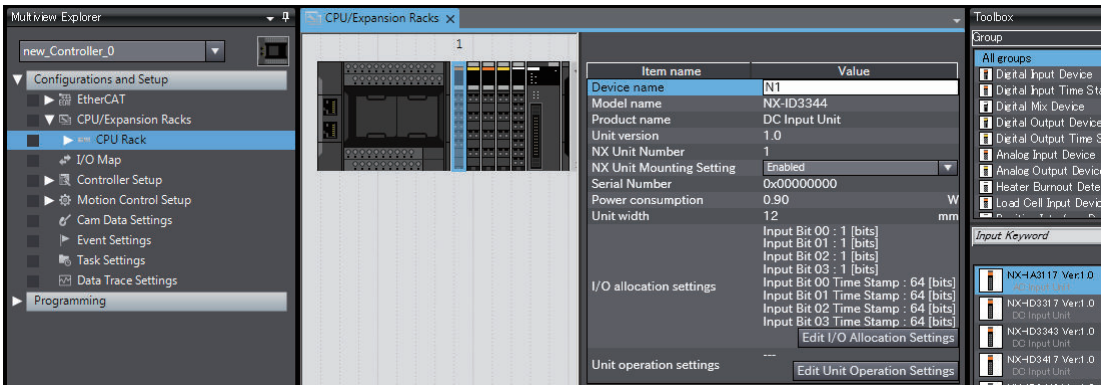
Parameter	Setting group	Description	Set value	Default	Update timing	Changes in RUN mode
Unit Information Note: Set the information for each slot.	Basic Input Unit Input Response Time	Sets the input response time (ON response time = OFF response time) of the Basic Input Unit. You can set no filter or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

4-3 Initial Settings for NX Units

This section describes the initial settings that are required for NX Units on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units. These settings are not provided for the NX701 CPU Units and NJ-series CPU Units.

4-3-1 NX Unit Settings

Select an NX Unit to set and make the settings on the CPU and Expansion Racks Tab Page on the Sysmac Studio.



The settable items for the selected NX Unit are listed below.

Item	Description	Set value	Default
Device name	The name of the NX Unit.	Any value*1	*2
NX Unit Mounting Setting	This setting enables or disables the mounting of an NX Unit. Refer to *8-2-2 Mounting Settings of NX Units on the CPU Unit* on page 8-11 for details on this setting.	Enabled Disabled	Enabled
I/O allocation settings	These are the settings for which I/O data in the NX Unit to exchange. *3 Click the **Edit I/O Allocation Settings** Button to edit these settings. You cannot change this setting for System Units. Refer to *4-3-2 I/O Allocation Settings* on page 4-22 for an overview of I/O allocation settings and how to edit the settings.	---	Refer to the manual for the specific NX Unit.
Unit operation settings	These are the Unit operation settings for the NX Unit. *4 Click the **Edit Unit Operation Settings** Button to edit these settings. You cannot change this setting for System Units. Refer to *4-3-3 Unit Operation Settings* on page 4-24 for the Unit operation settings.	---	Refer to the manual for the specific NX Unit.
Unit application data	This data controls the functionality that is specific to each NX Unit. Not all NX Units have Unit application data.	---	Refer to the manual for the specific NX Unit.

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

*2. "N" followed by serial numbers that start from "1".
*3. The NX Units contain default values for the I/O settings. You do not need to edit the default values for a standard exchange. Change the settings as necessary.
*4. The settings that are available depend on the type of the NX Unit. For example, Digital Input Units have a setting for the input filter value, and Digital Output Units have a setting for the output value at load rejection. Refer to the manual for the specific NX Unit for the settings and their meanings.

4-3-2 I/O Allocation Settings

This section provides an overview of I/O allocation settings and how to edit the settings.

An Overview of I/O Allocation Settings

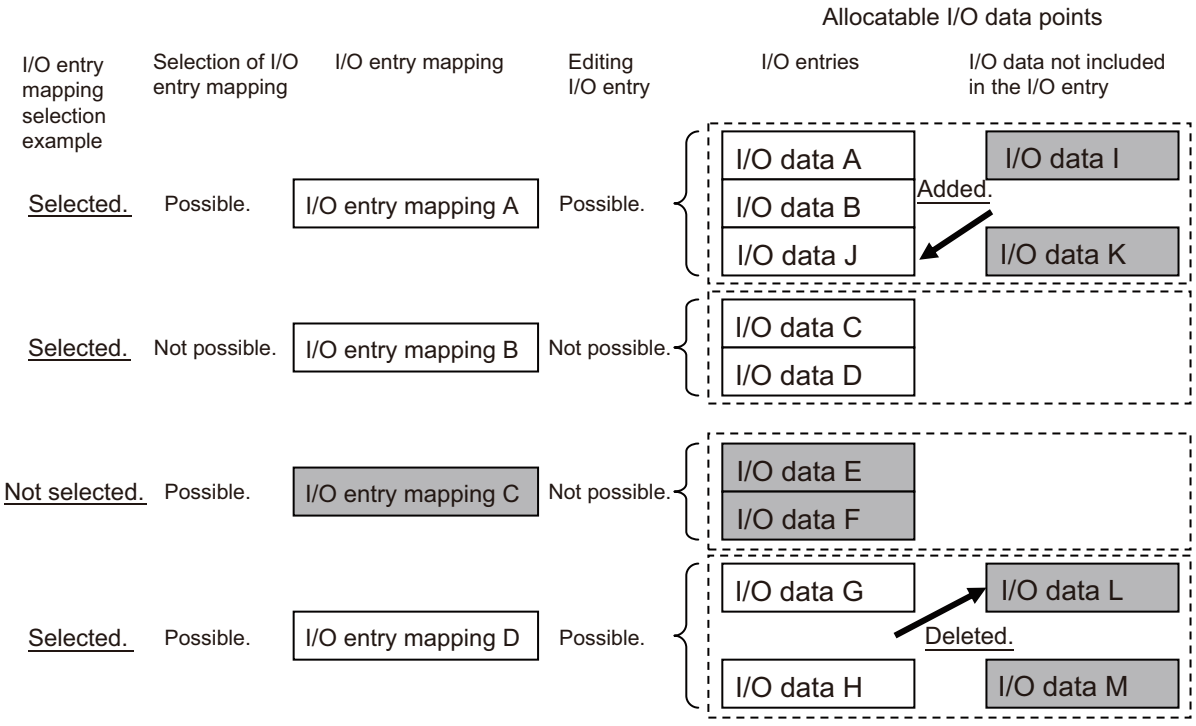
To allocate I/O, select an I/O entry mapping and register an I/O entry to the I/O entry mapping.

● Selecting I/O Entry Mappings

An I/O entry mapping defines a set of I/O data. Each Unit has its own I/O entry mapping.
The data for each I/O entry included in the selected I/O entry mappings are exchanged. Default values are assigned to the I/O entry mapping selections. Change the I/O entry mapping selections as necessary. If an I/O entry mapping must be selected, the option to deselect it will not be available.

● Registering I/O Entries

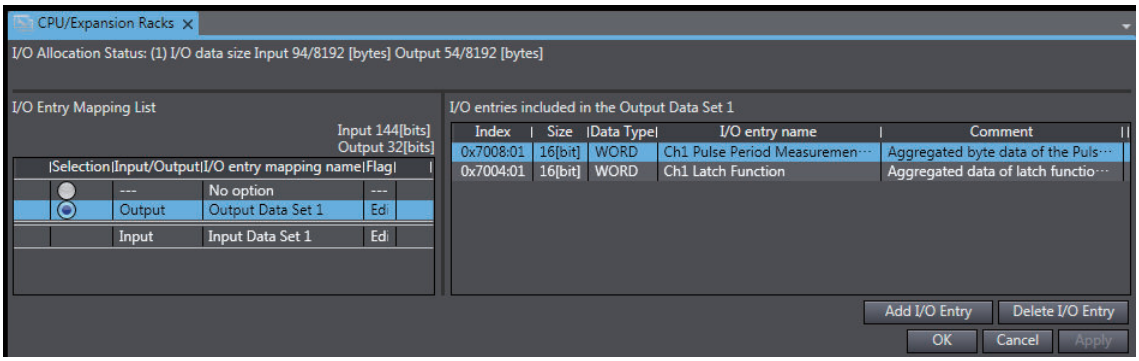
The I/O data assigned to an I/O entry mapping is called an I/O entry.
Default values are assigned to the I/O entries in each I/O entry mapping. Some I/O entry mappings allow you to add or delete I/O entries. Also, the I/O data that you can assign to an I/O entry mapping is predetermined. Change the settings as necessary.



*The shaded I/O data is not exchanged with process data communications.

Editing the I/O Allocation Settings

You can edit the I/O allocation settings for the NX Units as necessary.
When you click the **Edit I/O Allocation Settings** Button in the CPU and Expansion Racks Tab Page on the Sysmac Studio, the Edit I/O Allocation Settings Pane is displayed.



Name/Label	Description
I/O Allocation Status	The usage of I/O allocation for the entire CPU Unit is displayed here. • (1)I/O data size: The size of the I/O data that is allocated for the entire CPU Unit is given. The denominator is the maximum allocatable size. The I/O data size gives the amount of memory that is used by the I/O data. This value will not necessarily be the same as the total sum of all I/O entry sizes. • (2)Number of I/O entry mappings: The number of I/O entry mappings that are allocated to the entire CPU Unit is given. The denominator is the maximum number of allocatable I/O data.
I/O Entry Mapping List	This is a mapping list of the I/O entries in the corresponding Unit. The I/O entry mapping list shows up to four inputs and outputs respectively. The I/O entry mapping list shows the following items. • Selection: This column is used to select the I/O entry mappings that you wish to allocate. Select the I/O entry mapping that you wish to allocate. If you do not want to allocate the I/O entry mapping as part of the I/O allocation information, select *No option*. • Input/Output: This column shows whether the data is an input or an output in terms of the CPU Unit. • I/O entry mapping name: This column gives the name of the I/O entry mapping. • Flag: If the I/O entry is editable, this column says "Editable." • If the I/O entry is not editable, this column says "---."
I/O entries	This pane allows you to view and edit the I/O entries for the I/O entry mappings that are selected in the I/O Entry Mapping List. Each I/O entry contains the following information. • Index: This is the index number for the NX object. The index is displayed after “0x” as index_number: subindex_number. • Size: This column gives the size of the I/O entry data. • Data Type: This column gives the data type of the I/O entry. • I/O entry name: This column gives the name of the I/O entry. • Comment: This column gives a description of the I/O entry.

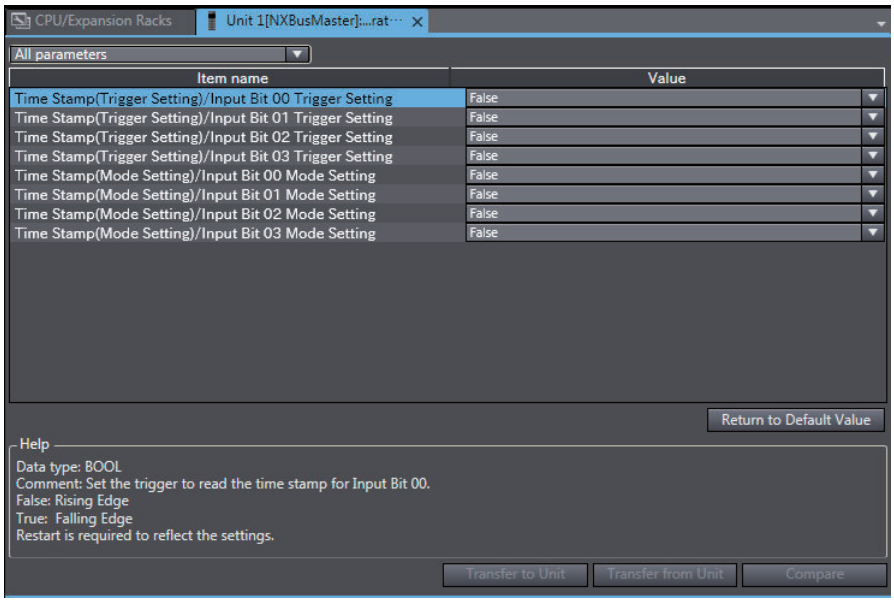
Name/Label	Description
Control buttons for the Edit I/O Allocation Settings Pane	• **Add I/O Entry** Button: This button adds an I/O entry to the selected I/O entry mapping. • **Delete I/O Entry** Button: This button deletes the selected I/O entry from the selected I/O entry mapping. • **OK** Button: This button confirms the settings in the Edit I/O Allocation Settings Pane, and returns the display to the CPU and Expansion Racks Tab Page. • **Cancel** Button: This button cancels the settings in the Edit I/O Allocation Settings Pane, and returns the display to the CPU and Expansion Racks Tab Page. • **Apply** Button: This button confirms the settings in the Edit I/O Allocation Settings Pane, and allows you to edit other I/O entries.

4-3-3 Unit Operation Settings

The settings that are available depend on the type of the NX Unit. For example, Digital Input Units have a setting for the input filter value, and Digital Output Units have a setting for the output value at load rejection.
Refer to the manual for the specific NX Unit for the settings and their meanings.

Editing the Unit Operation Settings

You can edit the Unit operation settings for the NX Units as necessary.
When you click the **Edit Unit Operation Settings** Button in the CPU and Expansion Racks Tab Page on the Sysmac Studio, the Edit Unit Operation Settings Tab Page is displayed.



Precautions for Correct Use

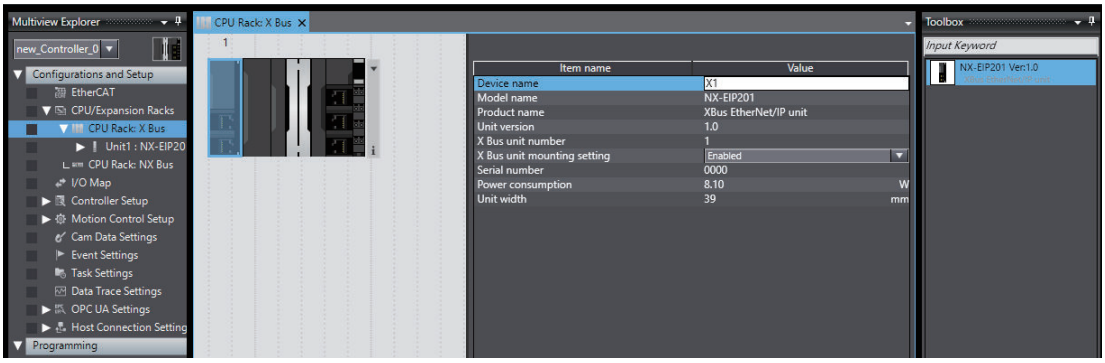
If the connected position of an NX Unit is changed, the Unit operation settings return to initial values.
Transfer the Unit operation settings again as necessary.

4-4 Initial Settings for X Bus Units

This section describes the initial settings of X Bus Units on the CPU Unit.
These settings are provided only for the NX502 CPU Units.

4-4-1 X Bus Unit Settings

Select an X Bus Unit to set and make the settings on the CPU and Expansion Racks Tab Page on the Sysmac Studio.



The settable items for the selected X Bus Unit are listed below.

Item	Description	Set value	Default
Device name	The name of the X Bus Unit.	Any value	*1
X Bus unit mounting setting	This setting enables or disables the mounting of an X Bus Unit. Refer to *8-3-3 X Bus Unit Mounting Settings* on page 8-24 for details on this setting.	Enabled Disabled	Enabled

*1. "X" followed by serial numbers that start from "1".

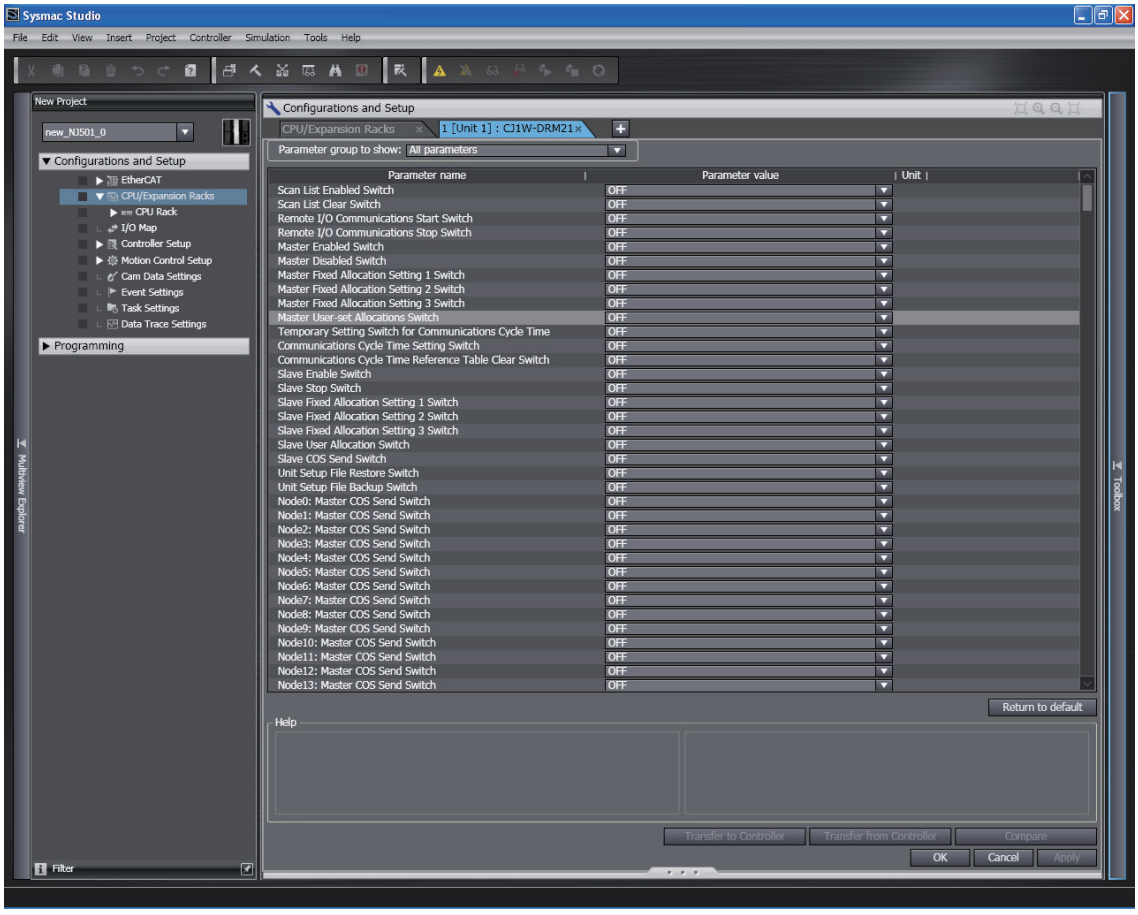
4-5 Initial Settings for Special Units

This section describes the initial settings that are required for the Special Units with NJ-series CPU Units. These settings are not provided for NX-series CPU Units.
You can use any of the following methods to set the initial settings of the Special Units.

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

1 Select the Unit in the Unit Configuration and Setup.

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



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

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

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

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

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

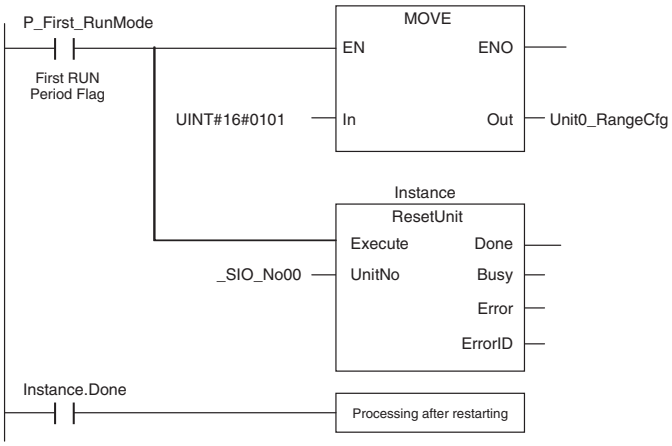
1 Use the Sysmac Studio to specify the initial values for the device variables of the CJ-series Units.

2 Download the variable table from the Sysmac Studio to the CPU Unit.
Select the **Clear the present values of variables with Retain attribute** Check Box.

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

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

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



Precautions for Safe Use

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

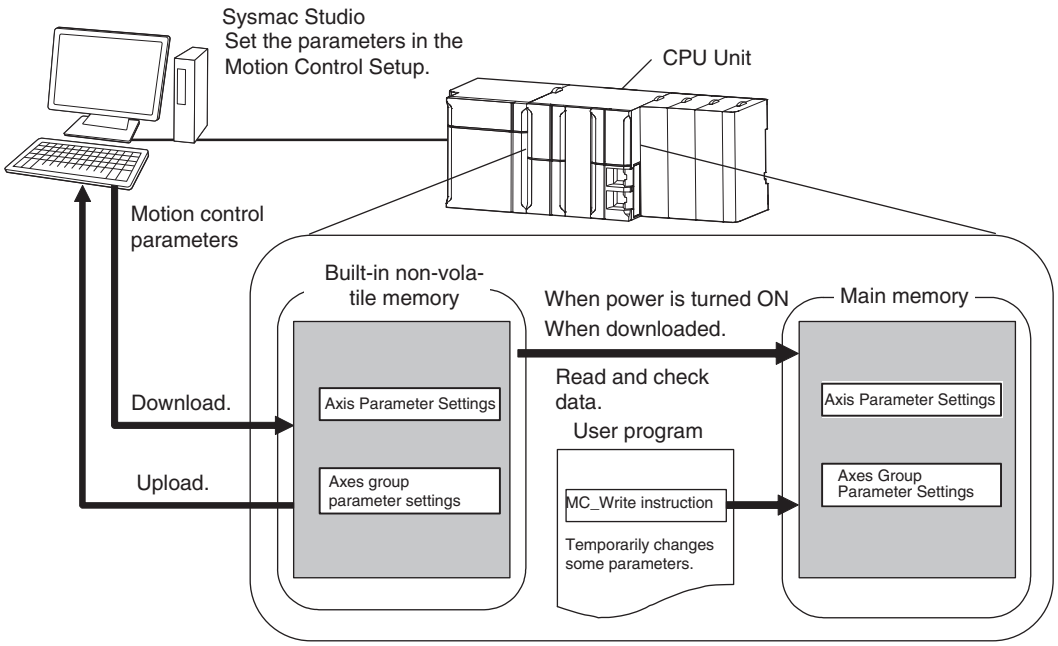
4-6 Initial Settings for the Motion Control Function Module

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

4-6-1 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

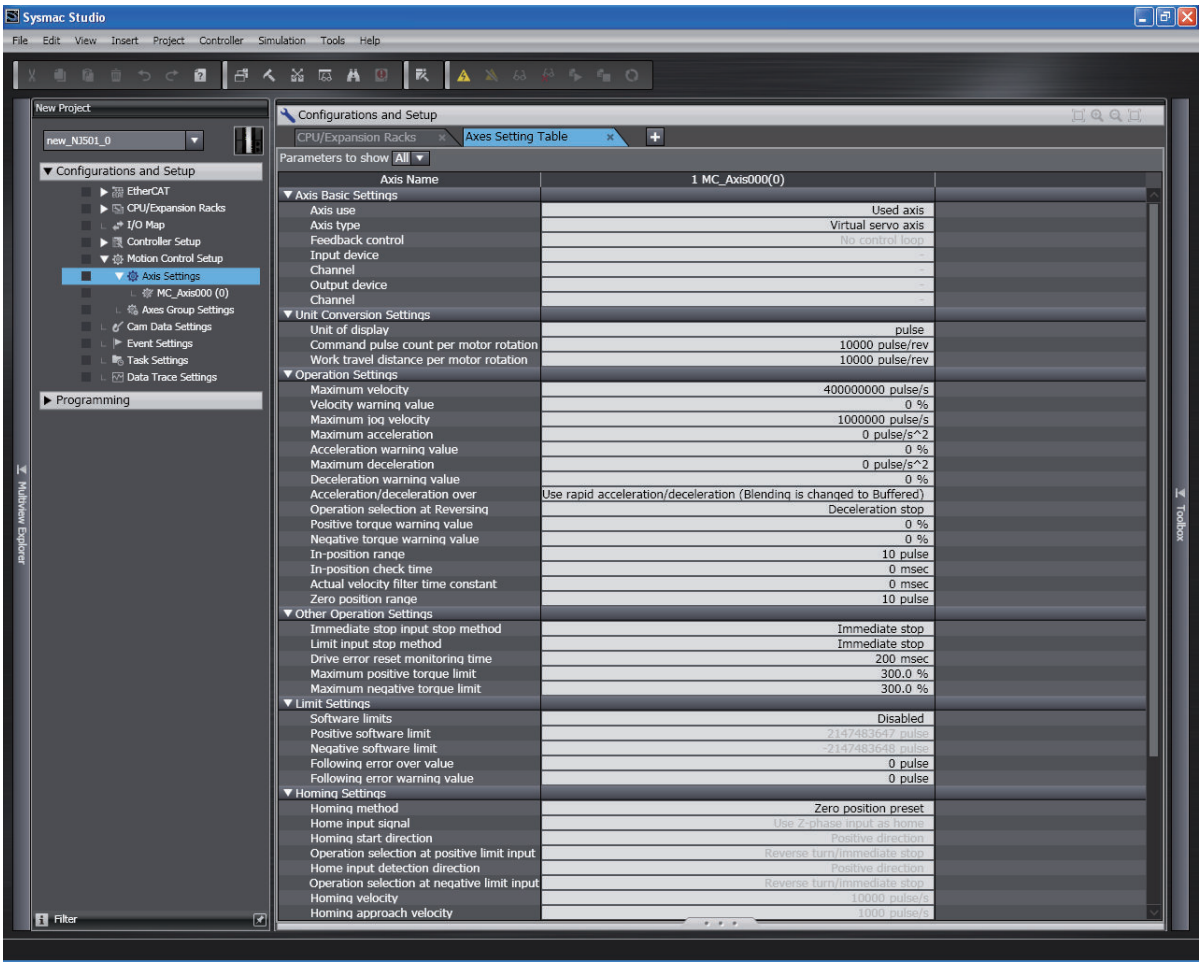


4-6-2 Setting Methods

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



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

Method 2: Setting with the MC_Write Instruction

You can temporarily overwrite some motion control parameters with the MC_Write instruction.
Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details.

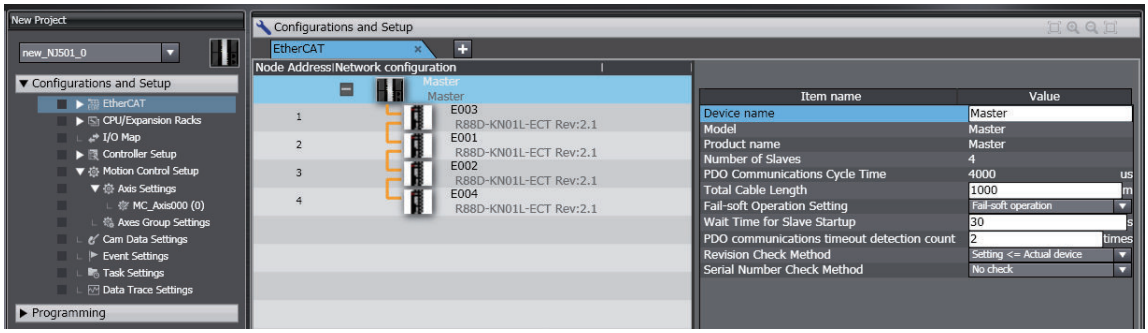
4-7 Initial Settings for the EtherCAT Master Function Module

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

The initial settings for the EtherCAT Master Function Module include the following and other items.

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

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



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

4-8 Initial Settings for the EtherNet/IP Function Module

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

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

- TCP/IP Settings
- Link Settings
- FTP Settings
- NTP Settings
- SNMP Settings
- SNMP Trap Settings
- FINS Settings
- CIP Settings

Select **Configurations and Setup** − **Controller Setup** − **Built-in EtherNet/IP Port Settings** on the Sysmac Studio to make these settings.
Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for detail.

Precautions for Correct Use

For the NX701-□□00, FINS Settings are not provided.

Version Information

CIP Settings can be made in the following CPU Units of the stated unit versions.

- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

4-9 Initial Settings for Built-in I/O

This section describes the initial settings that are required for I/Os built in the NX1P2 CPU Units.

The initial settings for the built-in I/Os are listed below.

- Input Filter Setting
- Load Rejection Output Setting

Select **Configurations and Setup** − **Controller Setup** − **Built-in I/O Settings** on the Sysmac Studio to make these settings.

Refer to the *NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual* (Cat. No. W579) for details.

4-10 Initial Settings for Option Boards

This section describes the initial settings that are required for Option Boards in the NX1P2 CPU Units.

The initial settings for the Option Boards are listed below.

- Configuration
- Option Board 1 Serial Communications Settings
- Option Board 2 Serial Communications Settings

Select **Configurations and Setup** - **Controller Setup** - **Option Board Settings** on the Sysmac Studio to make these settings.
Refer to the *NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual* (Cat. No. W579) for details.

4-11 Memory Settings for CJ-series Units

This section describes the initial settings that are required for memory used for CJ-series Units in the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

The initial settings for memory for CJ-series Units are listed below. Set whether to enable or disable memory for CJ-series Units and set the memory size when it is enabled.

- CIO
- WR
- HR
- DM
- EM

Additional Information

You cannot use the EM Area for the NX1P2 CPU Units.

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

4-11-1 Setting Procedure

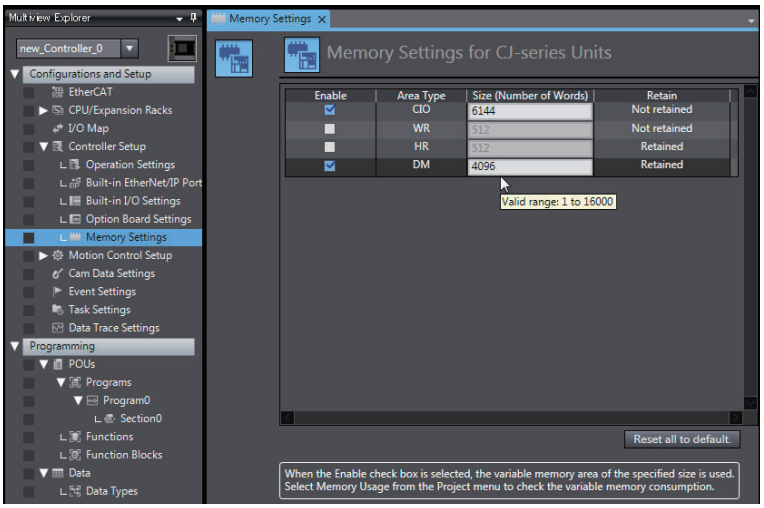
The procedure to make the memory settings for CJ-series Units is described below.

No.	Step	Description	Reference
1	Determining the usable memory	Determine the area type and the number of words of memory used for CJ-series Units to make available for data exchange with connected devices.	Manuals and technical materials for connected devices
2	Memory settings for CJ-series Units	In the Sysmac Studio, set the area type and the number of words of memory used for CJ-series Units to make available for connected devices.	*4-11-2 Setting Screen* on page 4-34
3	Programming	Create the user program that uses the memory used for CJ-series Units. If the set memory size is not sufficient, return to step 2 and increase the number of words.	*Section 6 Programming* on page 6-1
4	Downloading the project	Download the project from the Sysmac Studio.	*Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*
5	Checking operation and actual operation	Check the operation of the user program and connected devices.	*Section 7 Checking Operation and Actual Operation* on page 7-1

4-11-2 Setting Screen

Specify the memory used for CJ-series Units in the Memory Settings for CJ-series Units Tab Page, which is displayed by selecting **Memory Settings** under **Configurations and Setup** - **Controller Setup**.

The tab page used to make settings when an NX1P2 CPU Unit is used is shown below.



4-11-3 Settings

For the NX502 CPU Units and NX102 CPU Units, the settings are as follows.

Item	Setting group	Setting	Description	Set value	Default	Update timing	Changes in RUN mode
Memory settings for CJ-series Units	CIO*1	Enable	Enable or disable the generation of CIO area type memory used for CJ-series Units.	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	Specify the size of memory of area type CIO.	1 to 6,144	6,144	When downloaded to CPU Unit	Not allowed.
	WR*1	Enable	Enable or disable the generation of WR area type memory used for CJ-series Units.	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	Specify the size of memory of area type WR.	1 to 512	512	When downloaded to CPU Unit	Not allowed.
	HR*2	Enable	Enable or disable the generation of HR area type memory used for CJ-series Units.	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	Specify the size of memory of area type HR.	1 to 1,536	512	When downloaded to CPU Unit	Not allowed.
	DM*2	Enable	Enable or disable the generation of DM area type memory used for CJ-series Units.	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	Specify the size of memory of area type DM.	1 to 32,768	32768	When downloaded to CPU Unit	Not allowed.

Item	Setting group	Setting	Description	Set value	Default	Update timing	Changes in RUN mode
	EM0-EM18*2	Enable	Enable or disable the generation of EM area type memory used for CJ-series Units	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	Specify the size of memory of area type EM.	1 to 32,768	32,768	When downloaded to CPU Unit	Not allowed.

*1. Variables without a Retain attribute are used. The value can be set in 1-word increments.
*2. Variables with a Retain attribute are used. The value can be set in 1-word increments.

The settings for NX1P2 CPU Units are as follows:

Item	Setting group	Setting	Description	Set value	Default	Update timing	Changes in RUN mode
Memory settings for CJ-series Units	CIO*1	Enable	Enable or disable the generation of CIO area type memory used for CJ-series Units.	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	Specify the size of memory of area type CIO.	1 to 6,144	6,144	When downloaded to CPU Unit	Not allowed.
	WR*1	Enable	Enable or disable the generation of WR area type memory used for CJ-series Units.	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	Specify the size of memory of area type WR.	1 to 512	512	When downloaded to CPU Unit	Not allowed.
	HR*2	Enable	Enable or disable the generation of HR area type memory used for CJ-series Units.	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	Specify the size of memory of area type HR.	1 to 1,536	512	When downloaded to CPU Unit	Not allowed.
	DM*2	Enable	Enable or disable the generation of DM area type memory used for CJ-series Units.	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	Specify the size of memory of area type DM.	1 to 16,000	4,096	When downloaded to CPU Unit	Not allowed.

*1. Variables without a Retain attribute are used. The value can be set in 1-word increments.

*2. Variables with a Retain attribute are used. The value can be set in 1-word increments.

Additional Information

The setting for memory used for CJ-series Units can also be made when you use the NX701-□□20 Units. The settings for memory used for CJ-series Units for an NX701-□□20 Unit are as follows.

Item	Setting group	Setting	Set value	Default	Default Update timing	Changes in RUN mode
Memory Settings for CJ-series Units	CIO *1	Enable	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	1 to 6,144	6,144	When downloaded to CPU Unit	Not allowed.
	WR *1	Enable	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	1 to 512	512	When downloaded to CPU Unit	Not allowed.
	HR *2	Enable	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	1 to 1,536	512	When downloaded to CPU Unit	Not allowed.
	DM *2	Enable	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	1 to 32,768	32,768	When downloaded to CPU Unit	Not allowed.
	EM0-EM18 *2	Enable	Enable Disable	Disable	When downloaded to CPU Unit	Not allowed.
		Size (Number of Words)	1 to 32,768	32,768	When downloaded to CPU Unit	Not allowed.

*1. Variables without a Retain attribute are used. The value can be set in 1-word increments.

*2. Variables with a Retain attribute are used. The value can be set in 1-word increments.

5

Designing Tasks

This section describes the task system and types of tasks.

5-1 Overview of Task Designing Procedure 5-3

5-2 Overview of Tasks 5-6
5-2-1 Tasks 5-6
5-2-2 Instructions Related to Tasks 5-8
5-2-3 System-defined Variables Related to Tasks 5-8

5-3 Specifications and Basic Operation of Tasks for NX701 5-11
5-3-1 Specifications of Tasks for NX701 CPU Units 5-11
5-3-2 Guidelines for Separating Tasks for NX701 5-11
5-3-3 Basic Operation of Tasks for NX701 CPU Units 5-12
5-3-4 Event Task Execution Conditions for NX701 CPU Units 5-20
5-3-5 Event Task Execution Timing for NX701 CPU Units 5-25
5-3-6 Operation When Execution Condition Is Met Again Before Execution of the Event Task Is Completed 5-29

5-4 Specifications and Basic Operation of Tasks for NX502 CPU Units 5-31
5-4-1 Specifications of Tasks for NX502 CPU Units 5-31
5-4-2 Guidelines for Separating Tasks for NX502 CPU Units 5-31
5-4-3 Basic Operation of Tasks for NX502 CPU Units 5-32
5-4-4 Event Task Execution Conditions for NX502 CPU Units 5-40
5-4-5 Event Task Execution Timing for NX502 CPU Units 5-45
5-4-6 Operation When Execution Condition Is Met Again Before Execution of the Event Task Is Completed 5-49

5-5 Specifications and Basic Operation of Tasks for NX102 CPU Units and NX1P2 CPU Units 5-51
5-5-1 Specifications of Tasks for NX102 CPU Units and NX1P2 CPU Units 5-51
5-5-2 Guidelines for Separating Tasks for NX102 CPU Units and NX1P2 CPU Units 5-51
5-5-3 Basic Operation of Tasks for NX102 CPU Units and NX1P2 CPU Units 5-52
5-5-4 Event Task Execution Conditions for NX102 CPU Units and NX1P2 CPU Units 5-57
5-5-5 Event Task Execution Timing for NX102 CPU Units and NX1P2 CPU Units 5-62
5-5-6 Operation When Execution Condition Is Met Again Before Execution of the Event Task Is Completed 5-66

5-6 Specifications and Basic Operation of Tasks for NJ-series Controllers 5-68
5-6-1 Specifications of Tasks for NJ-series Controllers 5-68
5-6-2 Guidelines for Separating Tasks for NJ-series Controllers 5-68

5-6-3 Basic Operation of Tasks for NJ-series Controllers 5-69
5-6-4 Event Task Execution Conditions for NJ-series Controllers 5-76
5-6-5 Event Task Execution Timing for NJ-series Controllers 5-81
5-6-6 Operation When Execution Condition Is Met Again Before Execution of the Event Task Is Completed 5-85

5-7 Services Other Than Tasks 5-86
5-7-1 Execution Priorities and Execution Orders of Services Other Than Tasks 5-88
5-7-2 Processing Performed in and Execution Timing of the Tag Data Link Service 5-92
5-7-3 Processing Performed in and Execution Timing of the Option Board Service 5-96
5-7-4 Processing Performed in and Execution Timing of the Communications Bridge Service 5-97
5-7-5 Processing Performed in and Execution Timing of the System Services 5-98

5-8 Assignment and Settings Related to Tasks 5-102
5-8-1 Assigning I/O Refreshing to Tasks 5-102
5-8-2 Assigning Tasks to Programs 5-109
5-8-3 Parameters for Primary Periodic Task and Periodic Tasks 5-110

5-9 Ensuring Concurrency of Variable Values 5-114
5-9-1 Ensuring Concurrency of Variable Values between Tasks 5-114
5-9-2 Variable Access from Outside the Controller 5-120

5-10 Errors Related to Tasks 5-125

5-11 Monitoring Task Execution Status and Task Execution Times 5-128

5-12 Task Design Methods and I/O Response Times 5-133
5-12-1 Checking the Task Execution Time 5-133
5-12-2 Examples of Task Design 5-136
5-12-3 System Input and Output Response Times 5-137

5-1 Overview of Task Designing Procedure

This section provides an overview of the task designing procedure.

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

***STEP 1.* Software Design**	
	Step 1-1 I/O Processing Design
	STEP1-2 Designing Tasks
	Step 1-3 Designing Programs

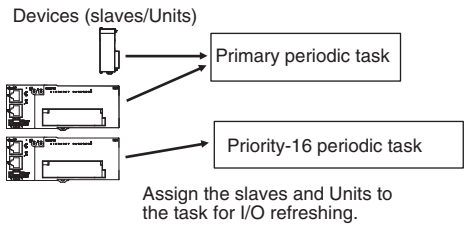
***STEP 2.* Software Setups and Programming**	
	Step 2-1 Slave and Unit Configurations
	Step 2-2 Controller Setup
	Step 2-3 Programming
	Step 2-4 Offline Debugging

***STEP 3.* Mounting and Setting Hardware**

***STEP 4.* Wiring**

***STEP 5.* Checking Operation and Starting Operation on the Actual System**

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series* on page 1-23 for detail.

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 NX701 CPU Units* on page 5-12 *5-4 Specifications and Basic Operation of Tasks for NX502 CPU Units* on page 5-31 *5-5-3 Basic Operation of Tasks for NX102 CPU Units and NX1P2 CPU Units* on page 5-52 *5-6-3 Basic Operation of Tasks for NJ-series Controllers* on page 5-69 *5-12 Task Design Methods and I/O Response Times* on page 5-133
Determine whether to use the primary periodic task, or the priority-5 or priority-16 periodic task for the I/O refreshing of each Unit and slave. 	*5-8-1 Assigning I/O Refreshing to Tasks* on page 5-102
Determine which programs to assign to the primary periodic task, to the priority-5, priority-16, priority-17, and priority-18 periodic tasks, and to the priority-8 and priority-48 event tasks.	*5-8-2 Assigning Tasks to Programs* on page 5-109
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.	*5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114
Design the tasks to access variables from outside of the Controller. Design the tasks to enable synchronization of accessing variables in the CPU Unit from outside of the Controller with the execution of a program in a specific task. EtherNet/IP tag data links are included in accessing variables.	*5-9-2 Variable Access from Outside the Controller* on page 5-120

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* on page 4-4

Offline Debugging with the Sysmac Studio

Desktop Operation Check	Reference
Perform desktop debugging of sequence control and motion control with the Simulator (virtual controller). **Monitor the task execution times in the Task Execution Time Monitor Display.**	*Section 7 Checking Operation and Actual Operation* on page 7-1 *5-11 Monitoring Task Execution Status and Task Execution Times* on page 5-128

5-2 Overview of Tasks

This section provides an overview of tasks.

5-2-1 Tasks

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

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

Type of task	Number of tasks	Task execution priority	Definition	Main processing content
Primary periodic task	1	4	The primary periodic task is executed once every task period. It has the highest execution priority. Motion control instructions and EtherCAT communications of the primary periodic task are executed on the primary periodic task period.	I/O refreshing, user program execution, and motion control
Periodic tasks	0 to 1	5*1	The priority-5 periodic task is executed once every task period. It has the second highest execution priority after the primary periodic task. Motion control instructions and EtherCAT communications of the priority-5 periodic task are executed on the priority-5 periodic task period.	I/O refreshing, user program execution, and motion control
	0 to 3	16*2, 17, or 18	The priority-16, priority-17, and priority-18 periodic tasks are executed once every task period. Motion control instructions and EtherCAT communications of the priority-16 periodic task are executed on the primary periodic task period.	The processing that can be performed depends on the task execution priority. Execution priority 16: I/O refreshing and user program execution Execution priority 17 or 18: User program execution
Event tasks	0 to 32	8 or 48	An event task is executed only once when the specified execution condition is met.	User program execution

*1. You can use the priority-5 periodic task only with the NX701 CPU Units.
*2. You cannot use the priority-16 periodic task with the NX102 CPU Unit or NX1P2 CPU Unit.

Version Information

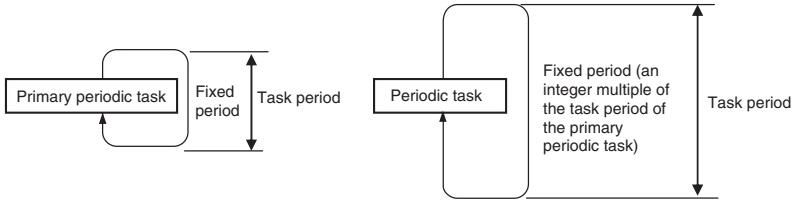
A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to use event tasks.

Additional Information

With an NX701 CPU Unit, you can execute motion control in the primary periodic task and in the priority-5 periodic task. If these two motion controls need to be identified, the motion control in the primary periodic task is called motion control 1, while the motion control in the priority-5 periodic task is called motion control 2.

● Primary Periodic Task and Periodic Tasks

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



From 1 to 128 programs can be assigned to one task. The programs that are assigned to a task are executed in the order that they are assigned. Execution of the all of the programs assigned to each task is called user program execution.
Exchanging data with CJ-series Units or EtherCAT slaves is called I/O refreshing.
You can assign I/O refreshing for each slave and Unit to the primary periodic task or the priority-5 or priority-16 periodic task. By default, I/O refreshing for all slaves and Units is assigned to the primary periodic task.

● Event Tasks

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

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

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

Precautions for Correct Use

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

Additional Information

Tasks operate differently between the NX701 CPU Units, NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units. Refer to *5-3 Specifications and Basic Operation of Tasks for NX701* on page 5-11, *5-4 Specifications and Basic Operation of Tasks for NX502 CPU Units* on page 5-31, *5-5 Specifications and Basic Operation of Tasks for NX102 CPU Units and NX1P2 CPU Units* on page 5-51, and *5-6 Specifications and Basic Operation of Tasks for NJ-series Controllers* on page 5-68 for details.

5-2-2 Instructions Related to Tasks

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

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.

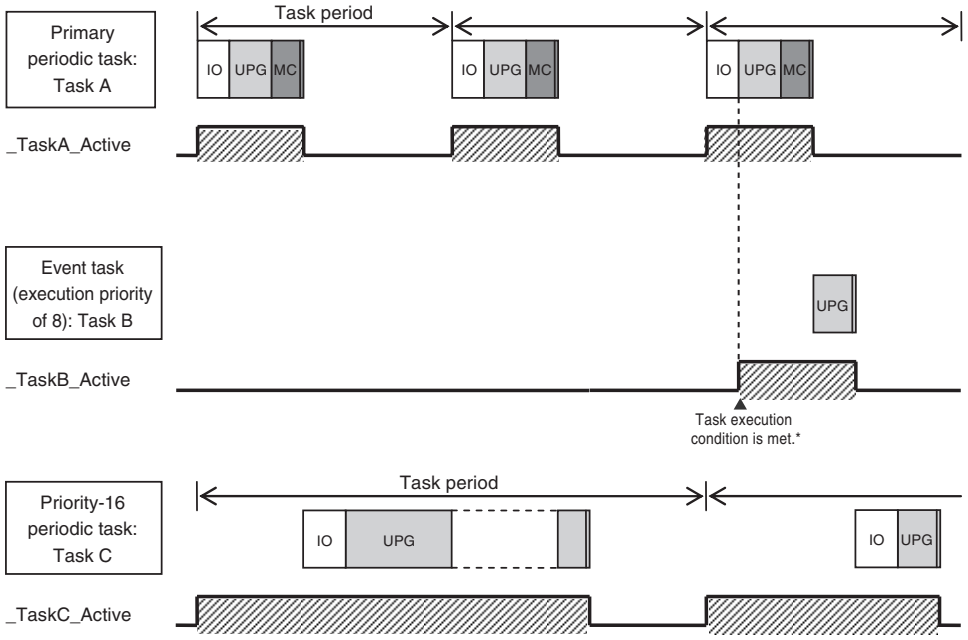
Variable name	Meaning	Function	Data type	R/W
_TaskName_Active	Task Active Flag	TRUE during task execution. FALSE from the completion of task execution until the end of the task period.	BOOL	R
_TaskName_LastExecTime	Last Task Execution Time	Gives the last execution time of the task.	TIME	R
_TaskName_MaxExecTime	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

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

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

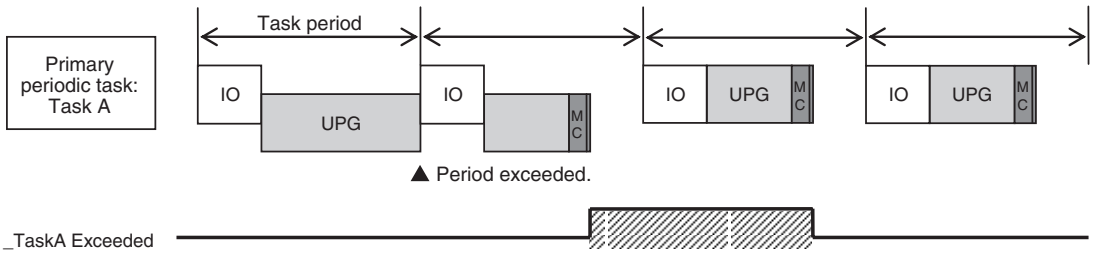
Flag Operation

● Task Active Flag (_TaskName_Active)



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

● Task Period Exceeded Flag (_TaskName_Exceeded)



5-3 Specifications and Basic Operation of Tasks for NX701

This section describes the specifications and basic operation of tasks for the NX701 with a multi-core processor.

5-3-1 Specifications of Tasks for NX701 CPU Units

The specifications of tasks are given in the following table.

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

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

Precautions for Correct Use

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

5-3-2 Guidelines for Separating Tasks for NX701

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

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

5-3-3 Basic Operation of Tasks for NX701 CPU Units

With a multi-core processor, the NX701 CPU Units can execute the primary periodic task and the priority-5 periodic task in parallel. The order in which tasks are executed depends on the execution priority that is set for each task.

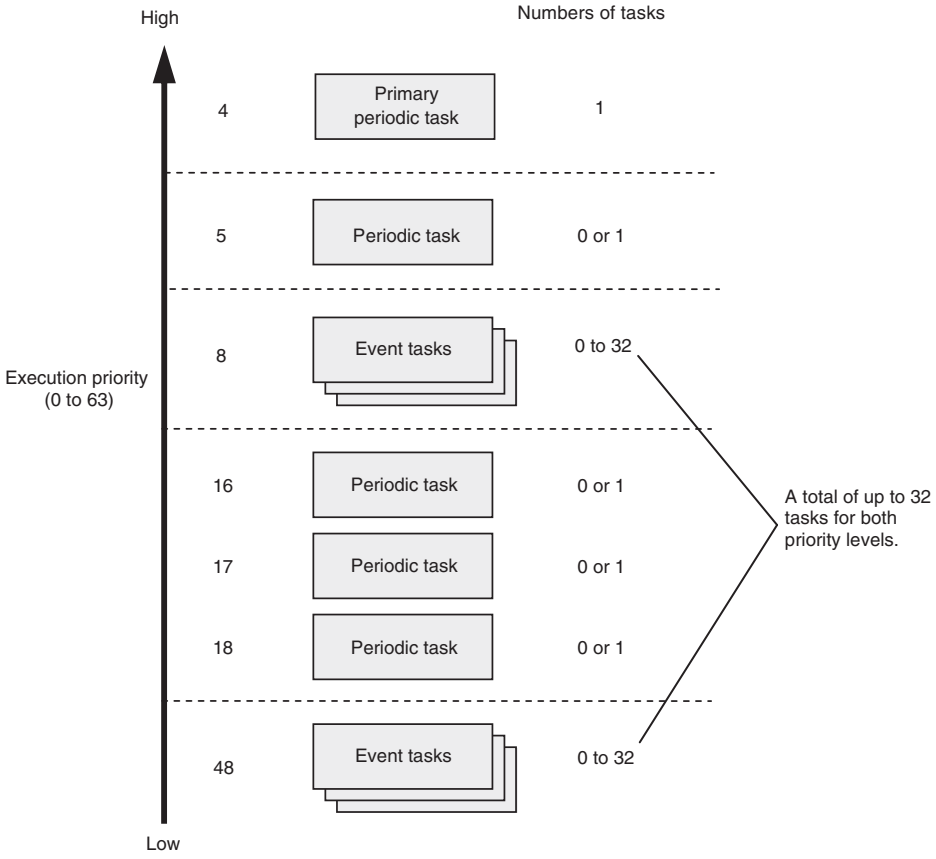
Additional Information

With an NX701 CPU Unit, you can execute multiple tasks, the tag data link service, and system services in parallel.

Task Execution Priority

The type of the task determines its execution priority. The CPU Unit execute the task with the highest execution priority first.
If the execution condition is met for another task, Tb, that has a higher execution priority while task Ta execution is in progress, the NX-series CPU Unit will assign Tb to the available core for processing on a priority basis.
The execution priority for each task type is given in the following table. The smaller the value of the execution priority, the higher the priority.

Task	Execution priority	Tasks with the same execution priority
Primary periodic task	4	---
Periodic task	5, 16, 17, or 18	You cannot set the same execution priority for more than one task.
Event task	8 or 48	You can set the same execution priority for more than one event task. Refer to *5-3-5 Event Task Execution Timing for NX701 CPU Units* on page 5-25 for the order of execution.



Task Periods for the Primary Periodic Task and Periodic Tasks

The CPU Unit repeatedly and cyclically execute 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

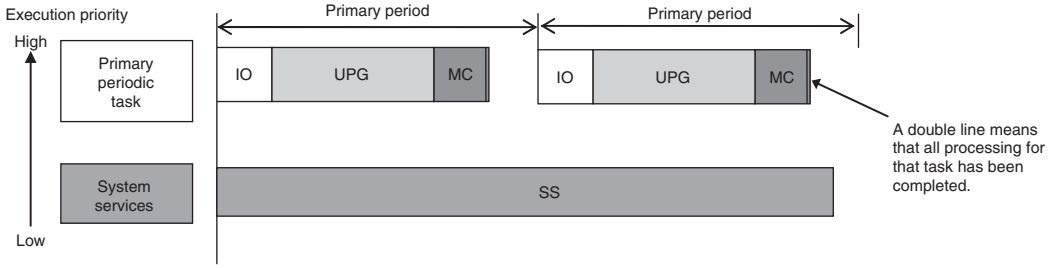
This section gives a few examples of the execution order for the primary periodic task and periodic tasks.

Refer to *5-3-5 Event Task Execution Timing for NX701 CPU Units* on page 5-25 for the order of execution of event tasks.

● Projects with Only the Primary Periodic Task

The primary periodic task is executed every primary period.

The system service shown in this figure refers to non-task related processing, such as communications processing, that is performed by the CPU Unit. Refer to *Processing Performed in System Services* on page 5-98 for details on the system services.



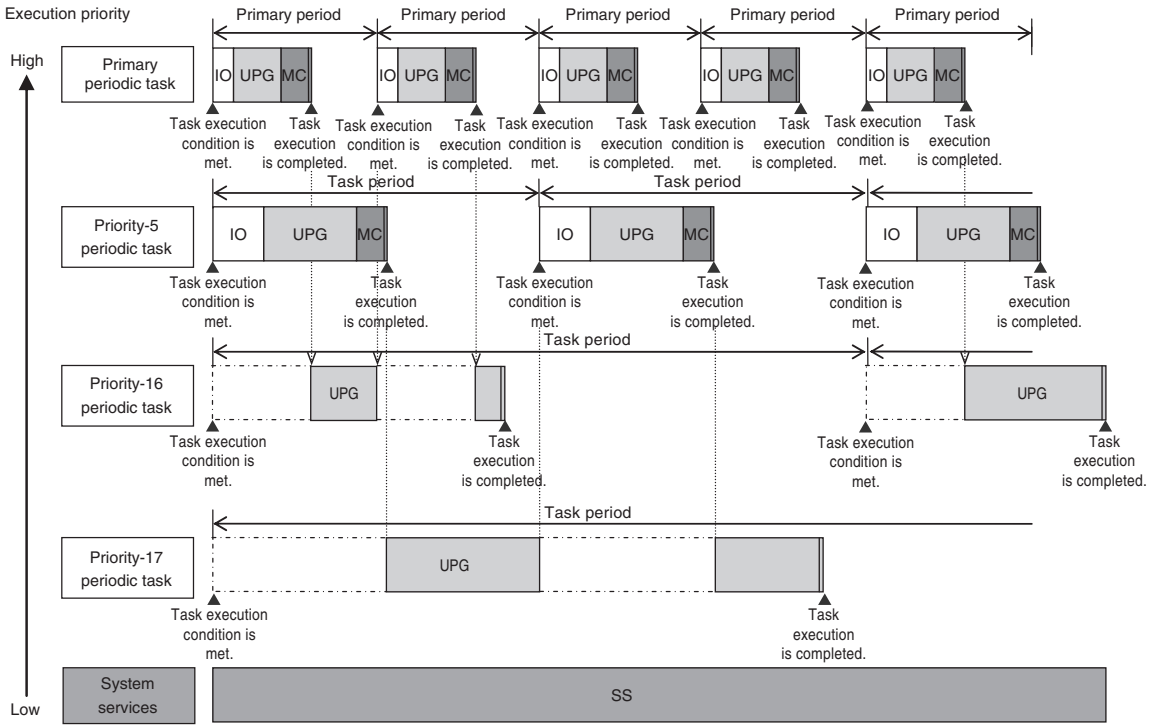
IO: Indicates I/O refreshing (data outputs and then data inputs).
UPG: User program execution.
MC: Indicates motion control.
SS: Indicates system services.

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

- The tasks are classified into the following two groups and executed.

Group	Type of task
Group 1	Primary periodic task Priority-16 periodic task
Group 2	Periodic task (priority 5, 17, or 18) Event task (priority 8 or 48)

- Each task in the same group is preferentially executed in descending order of execution priority.
- The primary periodic task has the highest execution priority, so it is always executed in the primary period.
- The priority-5 periodic task is executed in parallel with the primary periodic task.
- The priority-16 periodic task is executed after execution of the primary periodic task is completed.
- The priority-17 periodic task has a lower execution priority than the priority-5 periodic task, so it is executed when the priority-5 periodic task is not being executed.
- In this example, the task period for the priority-5 periodic task is set to twice the primary period. Also, the task period for the priority-16 periodic task is set to four times the primary period. This means that the timing at which execution of a task starts coincides with that of the primary periodic task once every two primary periods for the priority-5 periodic task and once every four primary periods for the priority-16 periodic task.
- The system services are executed at the required time without being affected by the tasks.



Precautions for Correct Use

If you have multiple tasks that read and write to the same variables, make sure to use exclusive control of variables between the tasks. Otherwise, a task other than the one currently in execution may change the variable values.
Refer to *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details.

Tasks and Operating Modes

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

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

Precautions for Correct Use

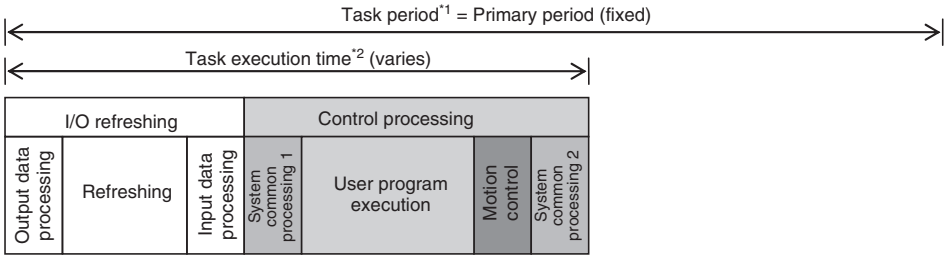
- Even if the execution condition for an event task is already met when you change the operating mode to RUN mode, the event task will not be executed. An event task is executed only when its execution condition changes from not met to met during RUN mode.
- Even in RUN mode, an event task is not executed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

The Processing Performed in Each Task

● Primary Periodic Task

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

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



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

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

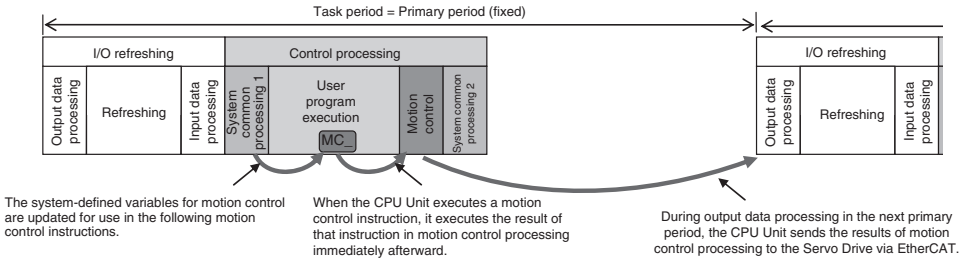
Processing		Processing contents
I/O refreshing	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Whether the condition expression for event task execution is met or not is determined. • Input refresh data is loaded from Input Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1		• Processing for exclusive control of variables in tasks 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.

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

*1. The Axis Current Values (Position, Velocity, and Torque) and Servo 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 CPU Unit outputs the results to the Servo Drives during I/O refreshing in the next primary periodic task.

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



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

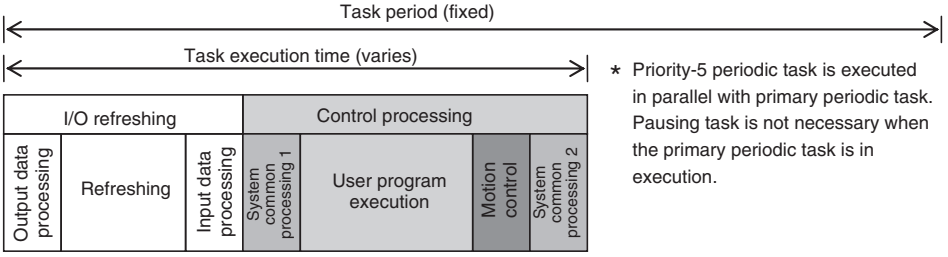
Refer to *5-12-3 System Input and Output Response Times* on page 5-137 for details.

● Priority-5 Periodic Task

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

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

The priority-5 periodic task is available only for the NX701 CPU Units.

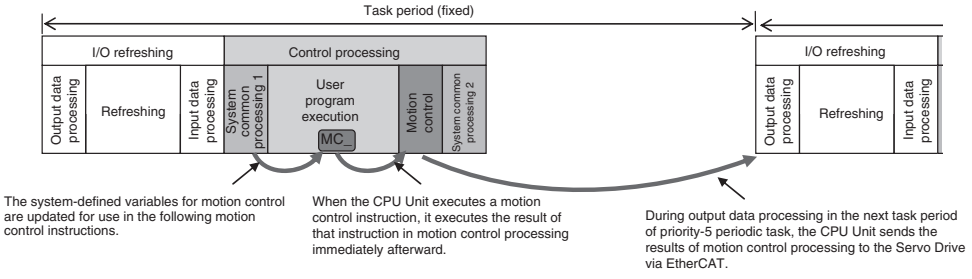


Processing		Processing contents
I/O refreshing	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Input refresh data is loaded from Input Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1		• Processing for exclusive control of variables in tasks is performed (when accessing tasks are set). • Motion input processing is performed.*1 • Data 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 priority-5 periodic task are executed. • Processing the motion outputs for I/O refreshing in the next priority-5 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 Servo 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 priority-5 periodic task, the CPU Unit executes the results from those instructions immediately afterward in motion control processing as shown below. The CPU Unit outputs the results to the Servo Drives during I/O refreshing in the next priority-5 periodic task.

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



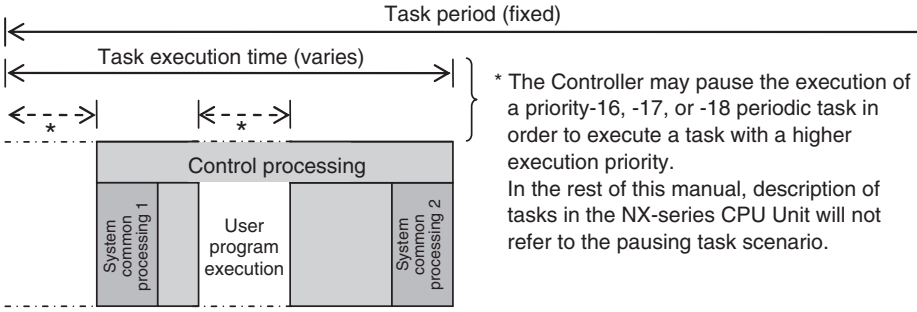
● Priority-16, Priority-17, or Priority-18 Periodic Task

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

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

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

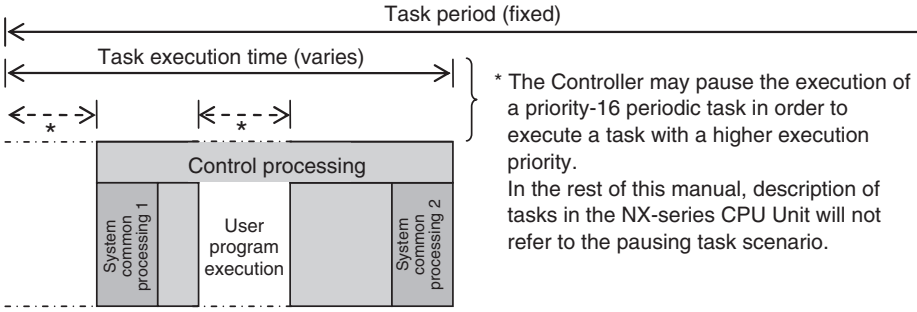
- Periodic Tasks That Do Not Control I/O



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.

- Priority-16 Periodic Task That Controls I/O



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

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

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

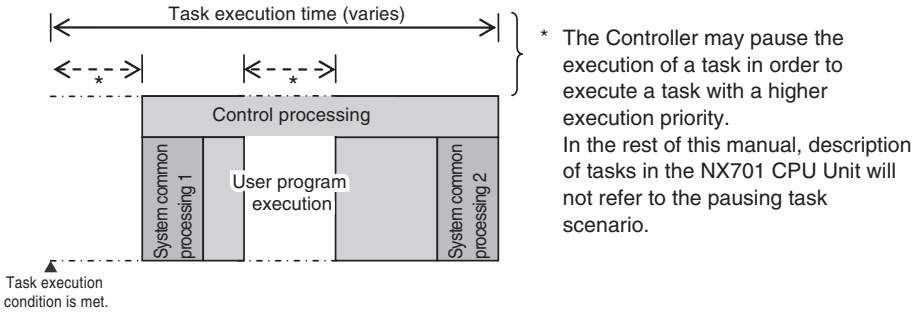
*2. This reflects the execution results of the user program for the EtherCAT slaves for which you set the priority-16 periodic task in the I/O Control Task Settings in the output refresh data. The output refresh data will be output in process data communications during I/O refreshing in the primary periodic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

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

● Event Tasks

An event task is executed only once when the specified execution condition is met. You can use 0 to 32 event tasks.

The processing details for event tasks are shown in the following figure.



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

*1. Refer to *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details on exclusive control.

5-3-4 Event Task Execution Conditions for NX701 CPU Units

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 5-26 for the timing of when the value of the specified variable is checked to see if the specified condition expression is met.

Precautions for Safe Use

If the following variables are specified for a condition expression when the execution condition is a condition expression for a variable, event tasks may not be executed when conditions are met or event tasks may be executed when conditions are not met.
- Structure members whose data size is 16 bits or more, except for system-defined variables for motion control
- Array elements whose data size is 16 bits or more

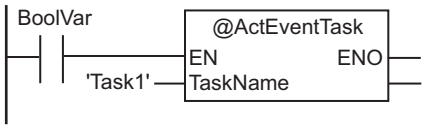
When the above variables are specified and the match evaluation is performed, perform either of the followings.
- Copy the above variables to the internal variables with a basic data type other than a data size of 64 bits, and access the copied internal variables.
- Use the settings for exclusive control of variables in tasks, and set the primary periodic task as a refreshing task.

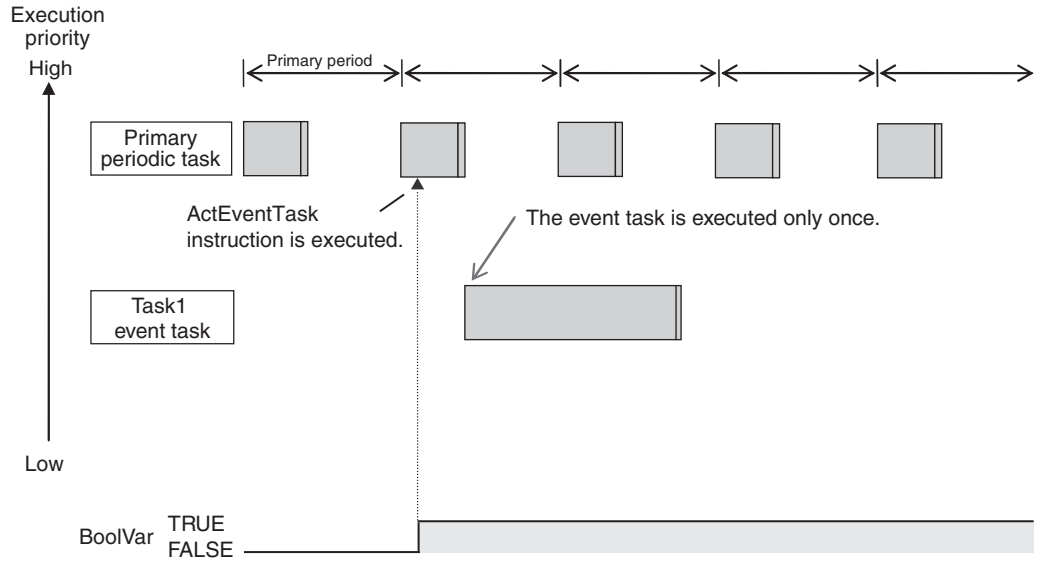
Executing Event Tasks for the ActEventTask Instruction

When the ActEventTask (Execute Event Task) instruction is executed in the user program, the specified event task is executed once. Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for the detailed specifications of the ActEventTask instruction.
Using the ActEventTask instruction to execute event tasks makes it easy to see which event tasks are executed. Also, this method is also effective when the execution condition for the event task may change before meeting the condition expression for the variable is determined.

● Example of User Programming Using the ActEventTask Instruction

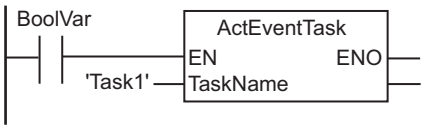
Example 1: Executing an Event Task Only Once When the Value of a Variable Changes
In the following example, the upward differentiation option is used for the ActEventTask instruction. This causes the Task1 event task to be executed only once when the *BoolVar* BOOL variable changes to TRUE.

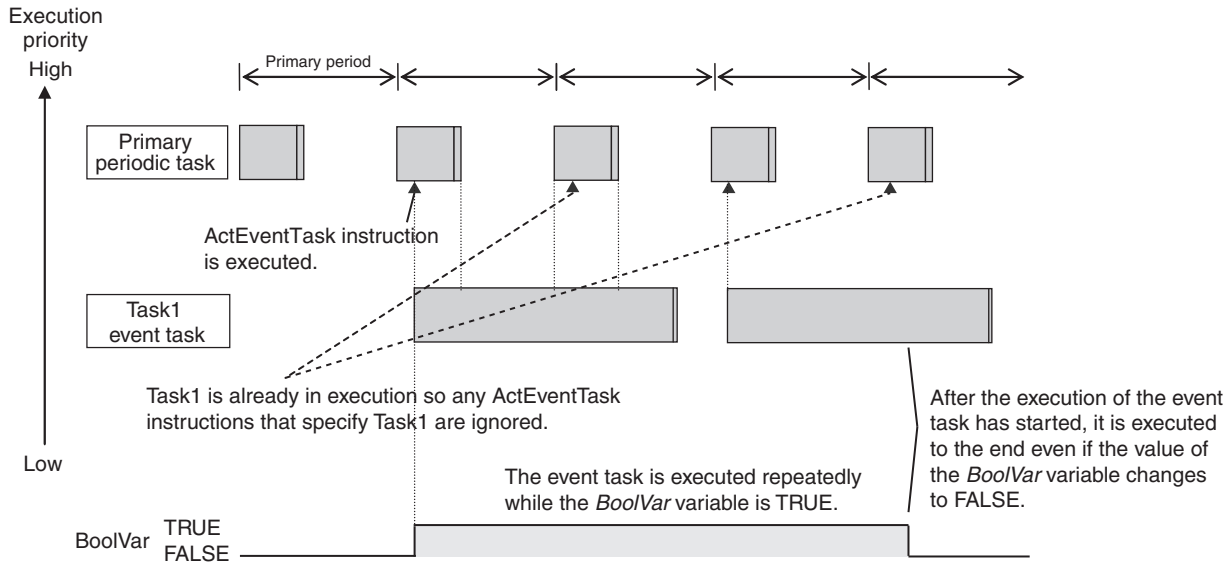




Example 2: Executing an Event Task Repeatedly While the Value of a Variable Matches a Specified Value

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





Executing Event Tasks When Condition Expressions for Variables Are Met

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

● Variables for Which You Can Specify Condition Expressions

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

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

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

● Data Types of Variables for Condition Expressions

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

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

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

● Condition Expressions That You Can Specify

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

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

● Valid Range of Comparison Constants

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

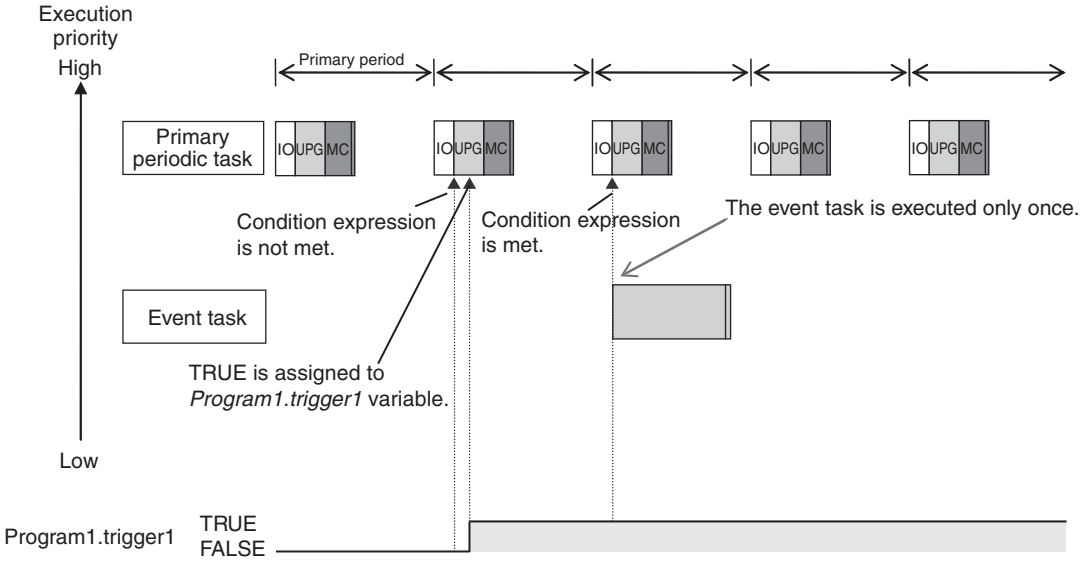
Refer to *Basic Data Types and Derivative Data Types* on page 6-32 for the valid range of values for each data type.

For example, if the variable that you specify for the condition expression is a BYTE variable, the valid range of comparison constant values is from BYTE#16#00 to BYTE#16#FF.

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

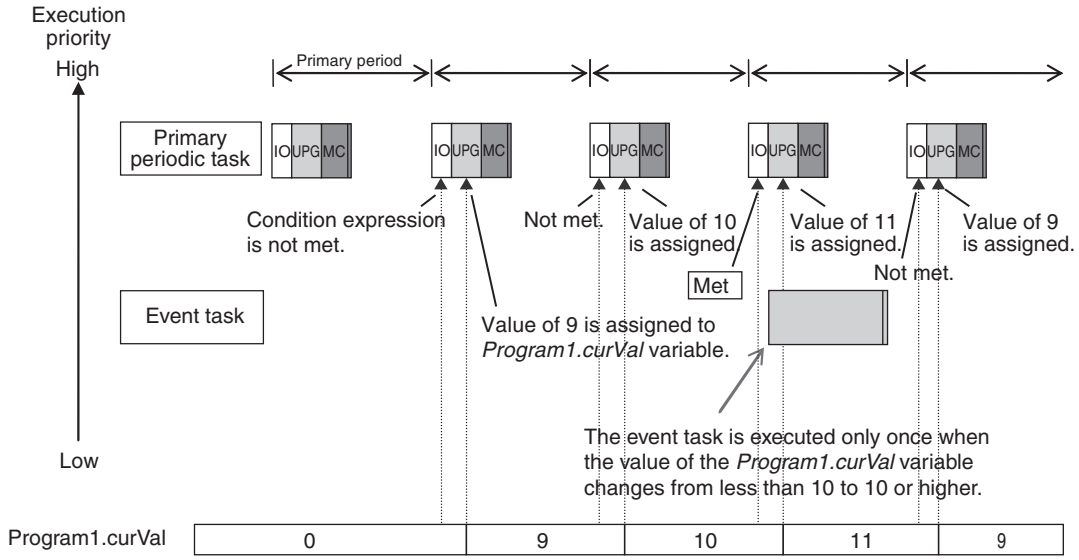
Example 1: Execution Condition for Event Task Set to a Change to TRUE of the *Program1.trigger1* Boolean Variable

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



Example 2: Execution Condition for an Event Task Set to When *Program1.curVal* (INIT variable) ≥ 10

The event task is executed only once when the value of *Program1.curVal* changes from less than 10 to 10 or higher.



Precautions for Correct Use

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

5-3-5 Event Task Execution Timing for NX701 CPU Units

The execution priority of event tasks is 8 or 48. With a multi-core processor, the NX701 CPU Units execute event tasks for which the execution condition is met according to the task execution priority. Depending on the user program, however, the CPU Unit may execute an event task in parallel with the primary periodic task, periodic tasks, or other event tasks with different execution priorities.

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

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

Refer to *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 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.

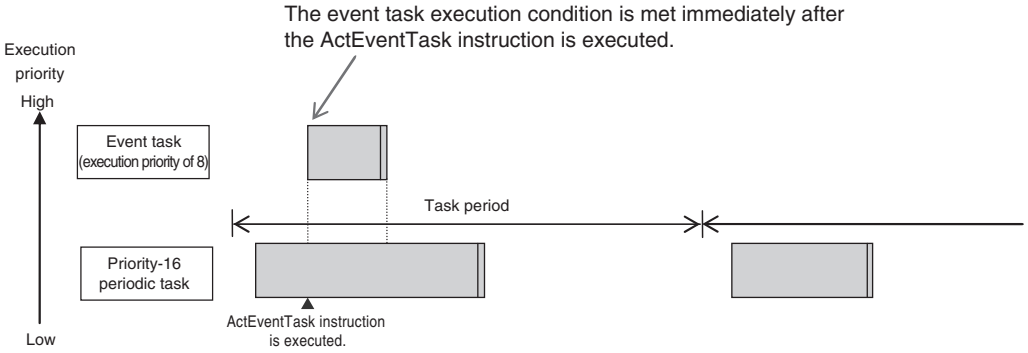
Differences in Execution Timing Based on the Execution Conditions of Event Tasks

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.

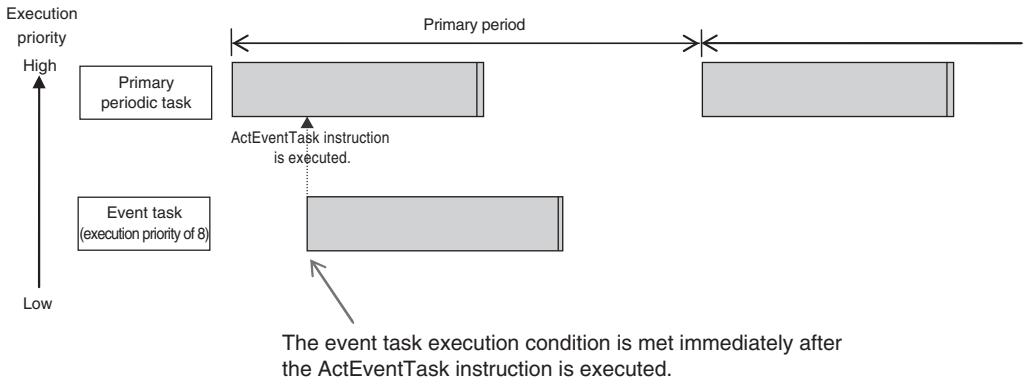
● Execution Timing When the Execution Condition Is an ActEventTask Instruction

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 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



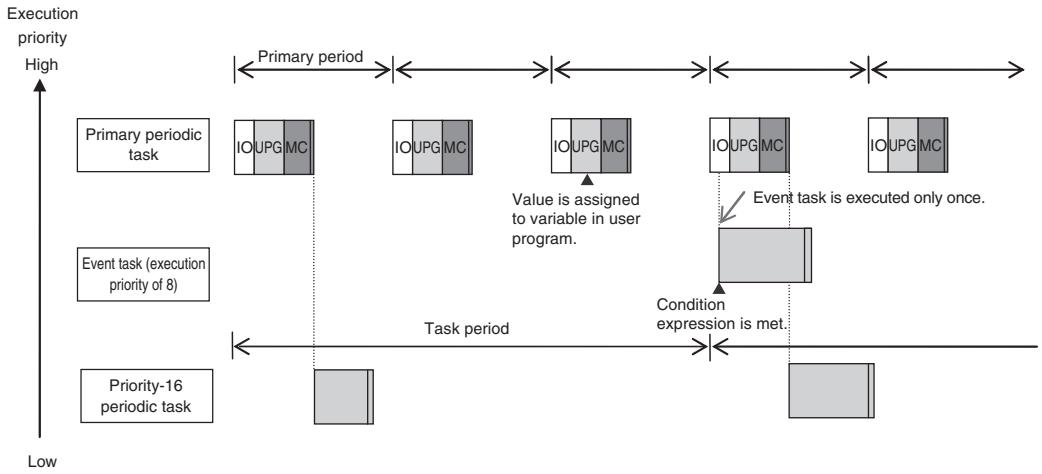
Example 2: Executing an Event Task with an Execution Priority Lower Than the Task That Executes an Instruction



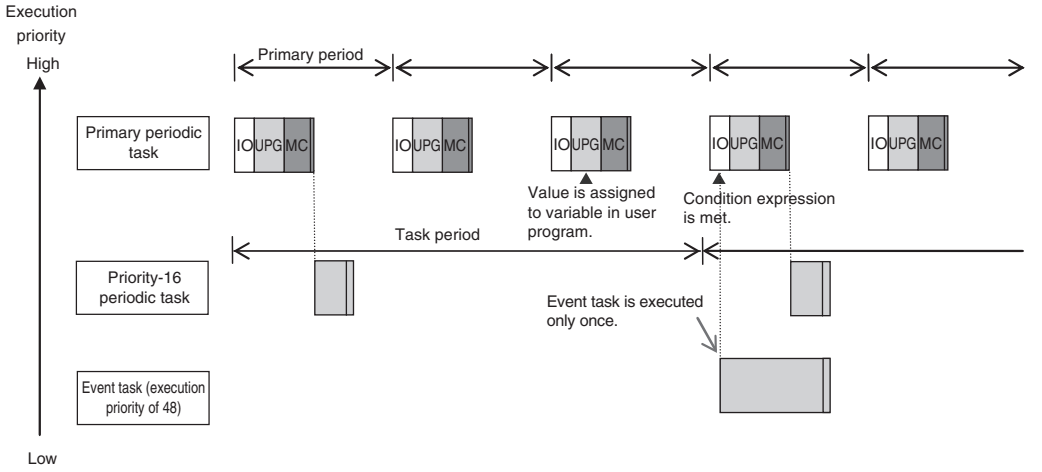
● Execution Timing When the Execution Condition Is a Condition Expression for a Variable

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 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



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



Precautions for Correct Use

- For the NX701 CPU Units, the timing at which the execution condition for an event task is met is the same regardless of whether the condition expression match is triggered by I/O refreshing in the primary periodic task, or by execution of a program that is assigned to the primary periodic task.

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

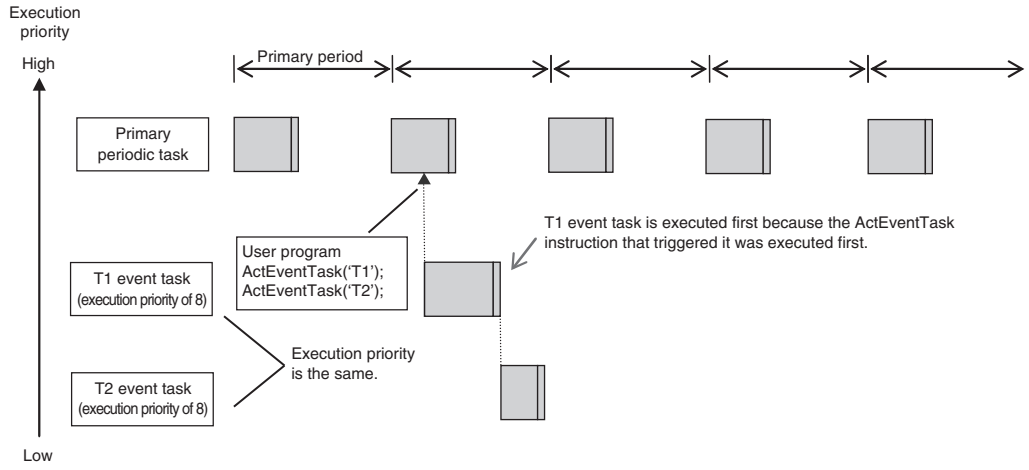
- In order for an event task to be executed, the condition expression must be met in the evaluation after the previous evaluation where the condition expression was not met. This means that even if the status of the condition expression changes from not met to met, if the condition returns to not met before the next evaluation, the event task will not be executed.
- For an NX701 CPU Unit, specify an internal variable defined in the program that is assigned to the primary periodic task if you specify an internal variable with a data size of 64 bits or more to the condition expression. If an internal variable that is not defined in the primary periodic task is specified, the concurrency of variable values may not be ensured. As the result, the match evaluation may not be correctly performed.

Execution Timing for Event Tasks with the Same Execution Priority

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

Example 1: When Two ActEventTask Instructions Are Executed

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



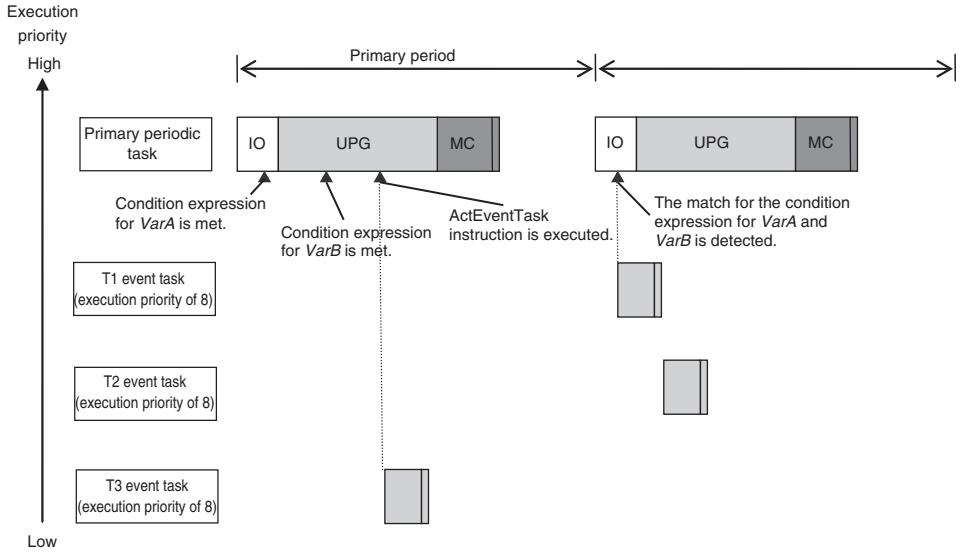
Example 2: When Both Condition Expressions for Variables and the ActEventTask Instruction Are Used

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

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

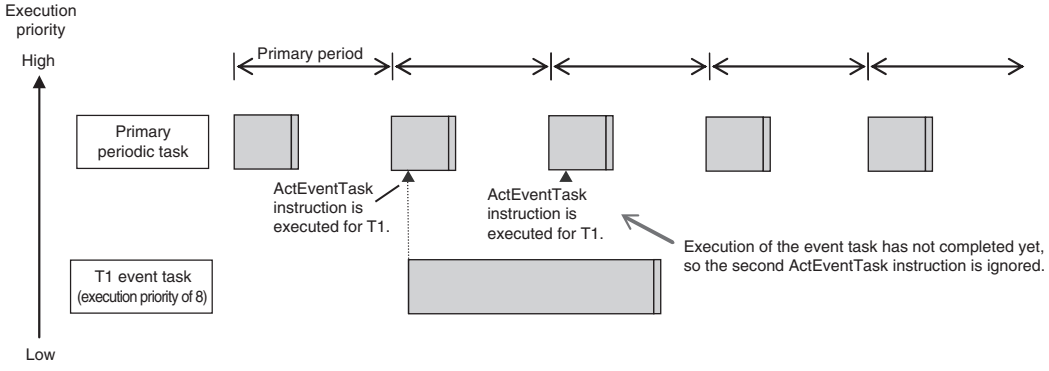
The operation would proceed as described below if the condition expression for *VarA* was met during I/O refreshing, the ActEventTask instruction was executed in the user program, and the condition expression for *VarB* was met during execution of the user program all in the same primary period.

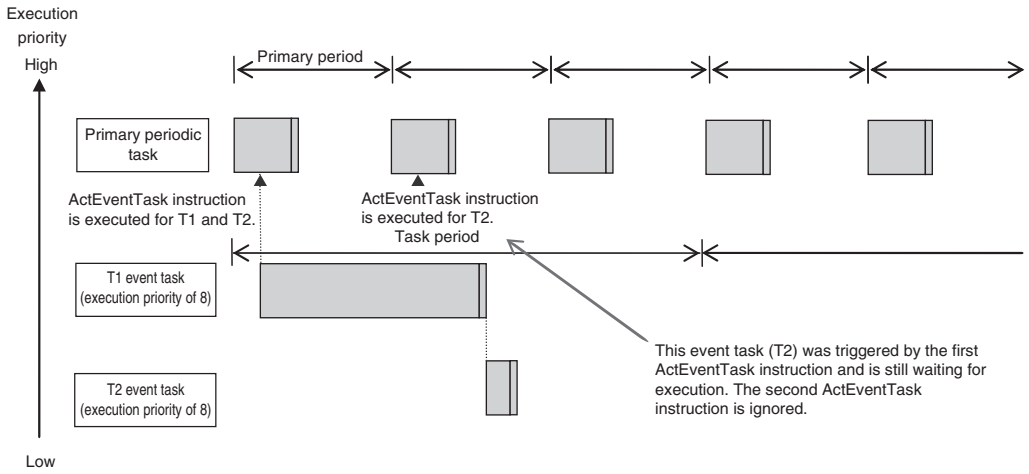
1. The condition expression for *VarA* is met during I/O refreshing.
2. The condition expression for *VarB* is met during execution of the user program.
3. At this point, T1 and T2 are not executed because the condition expressions are not yet evaluated.
4. The ActEventTask instruction is executed in 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.




Execution priority
High
Low
Primary period
Primary periodic task
ActEventTask instruction is executed for T1 and T2.
ActEventTask instruction is executed for T2.
Task period
T1 event task (execution priority of 8)
T2 event task (execution priority of 8)
This event task (T2) was triggered by the first ActEventTask instruction and is still waiting for execution. The second ActEventTask instruction is ignored.

5-4 Specifications and Basic Operation of Tasks for NX502 CPU Units

This section describes the specifications and basic operation of tasks for the NX502 CPU Units with a multi-core processor.

5-4-1 Specifications of Tasks for NX502 CPU Units

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	250 μs to 8 ms (in 250-μs increments)
Task periods of periodic tasks	• Priority 16, 17, or 18 1 ms to 100 ms (in 250-μs increments) Set the task period of each periodic task to an integer multiple of the task period of the primary periodic task. You cannot select any combination of task periods whose least common multiple exceeds 600 ms.

*1. There can be no more than one task with each of the following execution priorities: 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.

5-4-2 Guidelines for Separating Tasks for NX502 CPU Units

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.

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

5-4-3 Basic Operation of Tasks for NX502 CPU Units

The NX502 CPU Unit can execute multiple tasks and non-task services in parallel. The order in which tasks are executed depends on the execution priority that is set for each task.

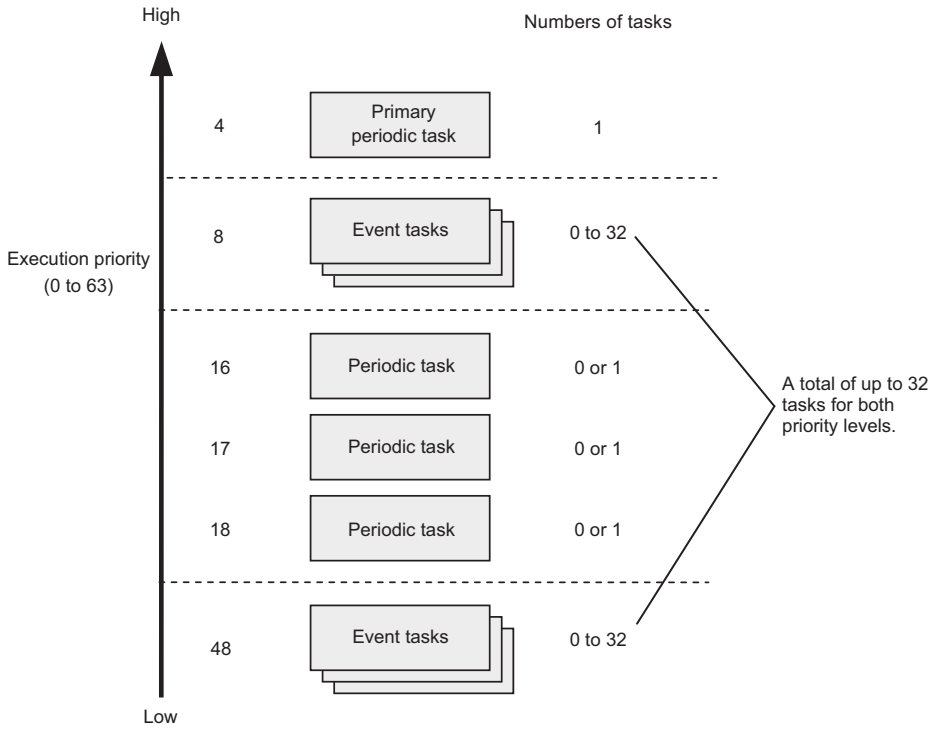
Additional Information

- With an NX502 CPU Unit, you can execute multiple tasks, the tag data link service, and system services in parallel.

Task Execution Priority

The type of the task determines its execution priority. The CPU Unit executes the task with the highest execution priority first.
If the execution condition is met for another task, Tb, that has a higher execution priority while task Ta execution is in progress, the CPU Unit will assign Tb to the available core for processing on a priority basis.
The execution priority for each task type is given in the following table. The smaller the value of the execution priority, the higher the priority.

Task	Execution priority	Tasks with the same execution priority
Primary periodic task	4	---
Periodic task	16, 17, or 18	You cannot set the same execution priority for more than one task.
Event task	8 or 48	You can set the same execution priority for more than one event task. Refer to *5-4-5 Event Task Execution Timing for NX502 CPU Units* on page 5-45 for the order of execution.



Task Periods for the Primary Periodic Task and Periodic Tasks

The CPU Unit repeatedly and cyclically execute 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.

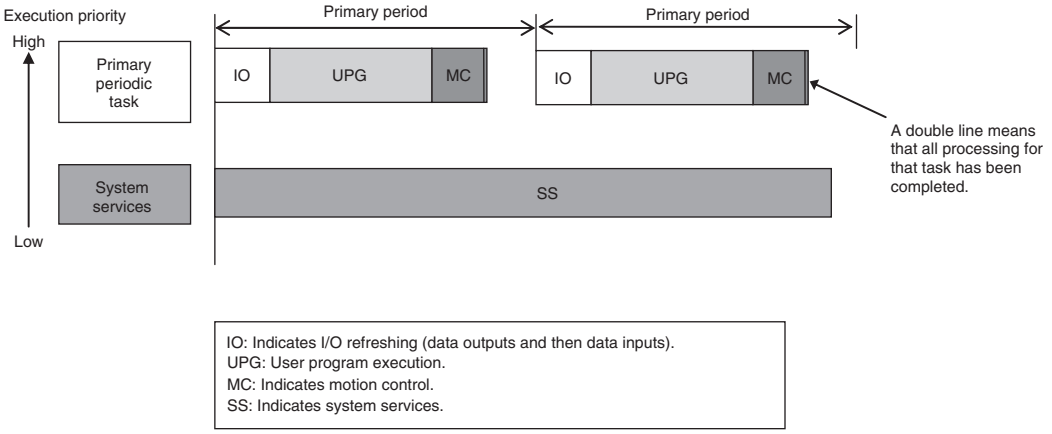
Examples of Execution Order for Tasks

This section gives a few examples of the execution order for the primary periodic task and periodic tasks.
Refer to *5-4-5 Event Task Execution Timing for NX502 CPU Units* on page 5-45 for the order of execution of event tasks.

● Projects with Only the Primary Periodic Task

The primary periodic task is executed every primary period.

The system service shown in this figure refers to non-task related processing, such as communications processing, that is performed by the CPU Unit. Refer to *Processing Performed in System Services* on page 5-98 for details on the system services.

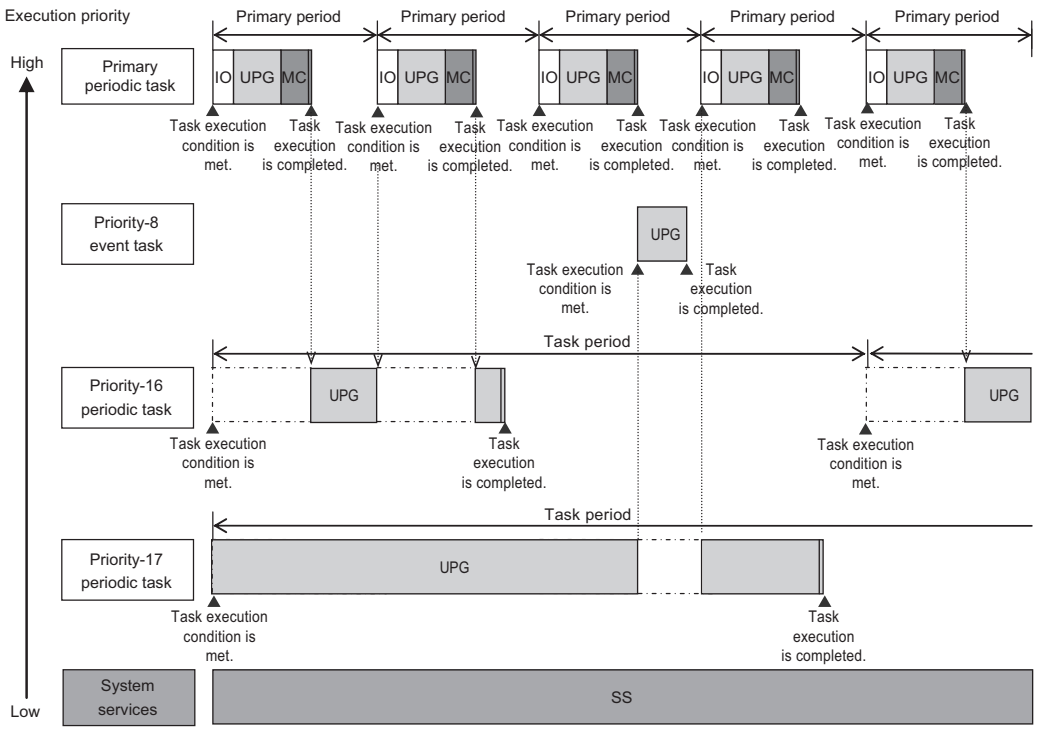


● Project with the Primary Periodic Task, Priority-16 Periodic Task, Priority-17 Periodic Task, and Priority-8 Event Task

- The tasks are classified into the following two groups and executed.

Group	Type of task
Group 1	Primary periodic task Priority-16 periodic task
Group 2	Periodic task (priority 17 or 18) Event task (priority 8 or 48)

- Each task in the same group is preferentially executed in descending order of execution priority.
- The primary periodic task has the highest execution priority, so it is always executed in the primary period.
- The priority-16 periodic task has a lower execution priority than the primary periodic task, so it is executed when the primary periodic task is not being executed.
- The priority-17 periodic task has a lower execution priority than the priority-8 event task, so it is executed when the priority-8 event task is not being executed.
- In this example, the task period for the priority-17 periodic task is set to four times the primary period. This means that execution of the priority-17 periodic task will start at the same time once every four primary periods.
- 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 *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details.

Tasks and Operating Modes

The relationship between the operating modes and tasks of the CPU Unit 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

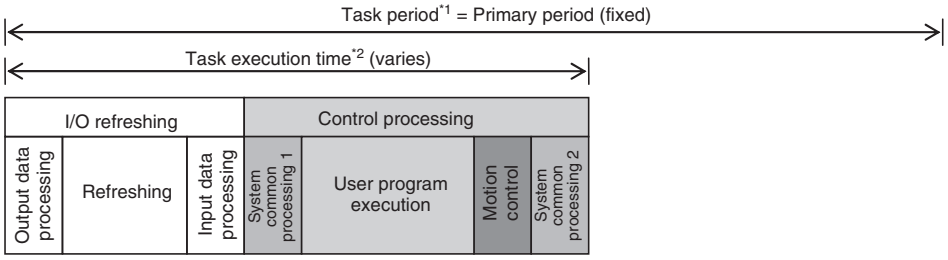
- Even if the execution condition for an event task is already met when you change the operating mode to RUN mode, the event task will not be executed. An event task is executed only when its execution condition changes from not met to met during RUN mode.
- Even in RUN mode, an event task is not executed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

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

Processing		Processing contents
I/O refreshing	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Whether the condition expression for event task execution is met or not is determined. • Input refresh data is loaded from Input Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1		• Processing for exclusive control of variables in tasks 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.

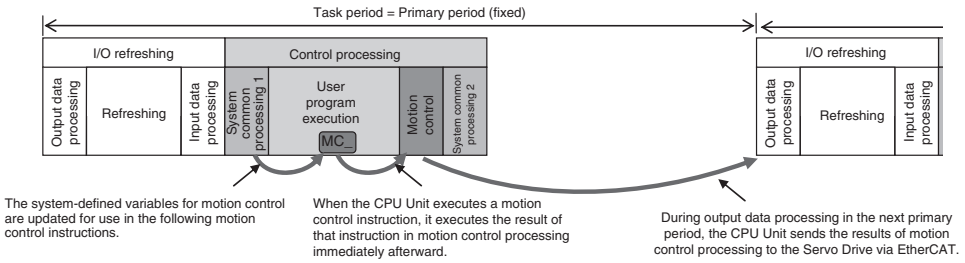
Processing	Processing contents
System common processing 2	• Processing for exclusive control of variables in tasks is performed (when refreshing tasks are set). • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the *variable access time* that is set in the **Task Settings**).*3 • 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 Servo Drive status in the system-defined variables for motion control are updated.

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

*3. For tag data link communications used with an NX-series EtherNet/IP Unit, the setting for variable access time is not required.

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 *5-12-3 System Input and Output Response Times* on page 5-137 for details.

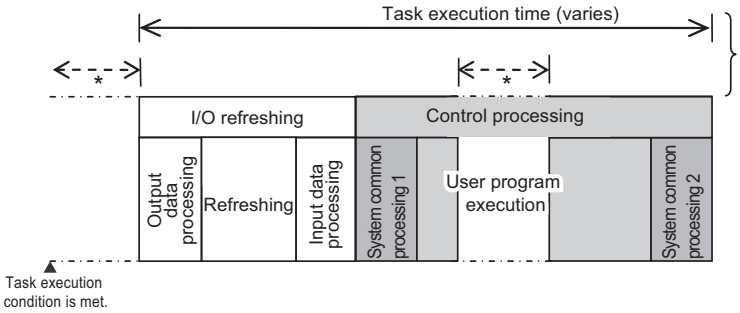
● Priority-16, Priority-17, or Priority-18 Periodic Task

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.

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

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

- Periodic Tasks That Do Not Control I/O



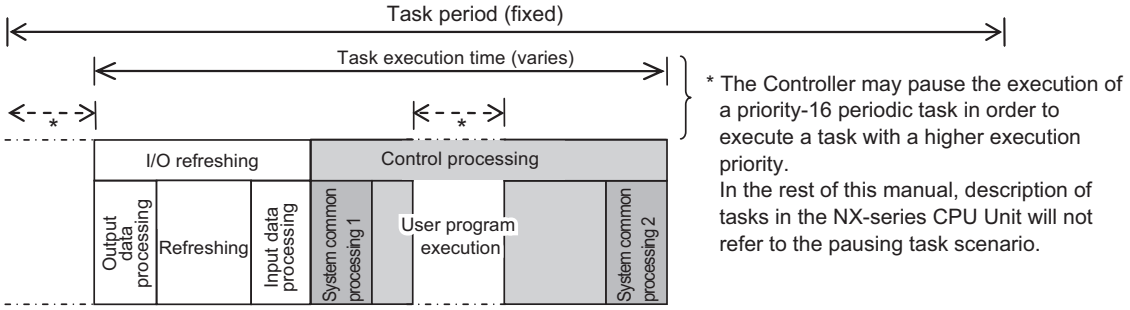
* The Controller may pause the execution of a task in order to execute a task with a higher execution priority.
In the rest of this manual, description of tasks in the NX502 CPU Unit will not refer to the pausing task scenario.

Processing		Processing
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). • 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**).*1 • 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. For tag data link communications used with an NX-series EtherNet/IP Unit, the setting for variable access time is not required.

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

- Priority-16 Periodic Task That Controls I/O



Processing		Processing contents
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	• Reflecting the input refresh data for EtherCAT.*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**).*3 • If there is processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed. • Reflecting the execution results in the output refresh data for EtherCAT.*2	

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

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

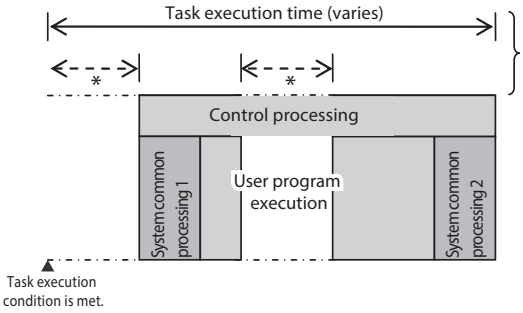
*3. For tag data link communications used with an NX-series EtherNet/IP Unit, the setting for variable access time is not required.

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.



* The Controller may pause the execution of a task in order to execute a task with a higher execution priority.
In the rest of this manual, description of tasks in the NX502 CPU Unit will not refer to the pausing task scenario.

Processing	Processing Contents
System common processing 1	• Processing for exclusive control of variables in tasks 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 *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details on exclusive control.

5-4-4 Event Task Execution Conditions for NX502 CPU Units

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 5-26 for the timing of when the value of the specified variable is checked to see if the specified condition expression is met.

Precautions for Safe Use

If the following variables are specified for a condition expression when the execution condition of event tasks is the same as the condition expression for a variable, event tasks may not be executed when conditions are met, or event tasks may be executed when conditions are not met.

- Structure members whose data size is 16 bits or more, except for system-defined variables for motion control
- Array elements whose data size is 16 bits or more.

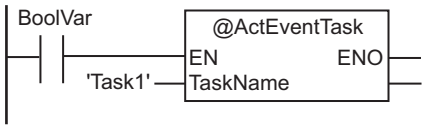
When the above variables are specified and the match evaluation is performed, perform either of the followings.

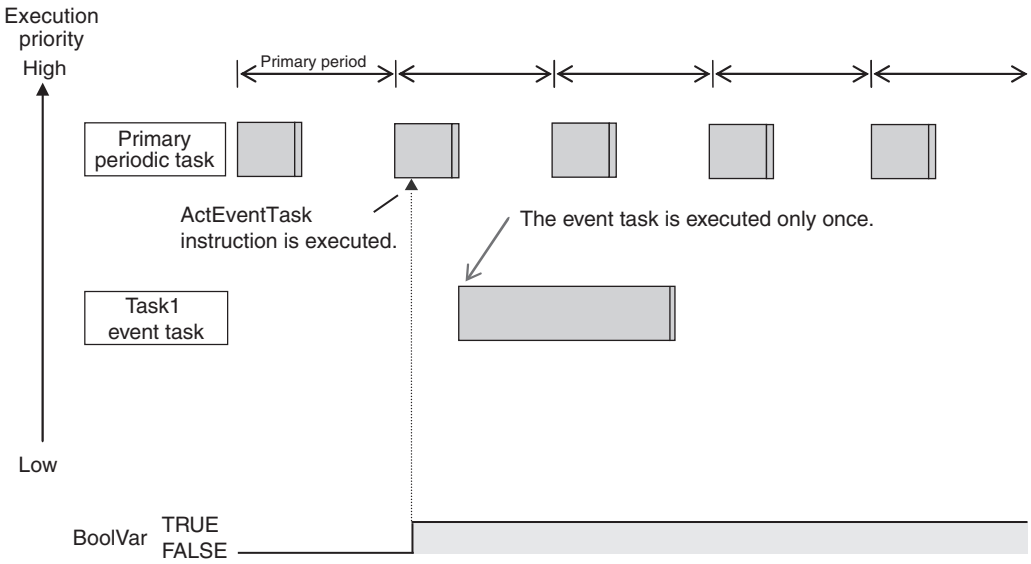
- Copy the above variables to the internal variables with a basic data type other than a data size of 64 bits, and access the copied internal variables.
- Use the settings for exclusive control of variables in tasks, and set the primary periodic task as a refreshing task.

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.

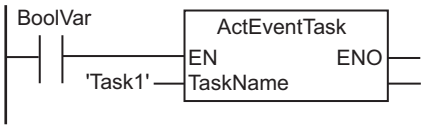


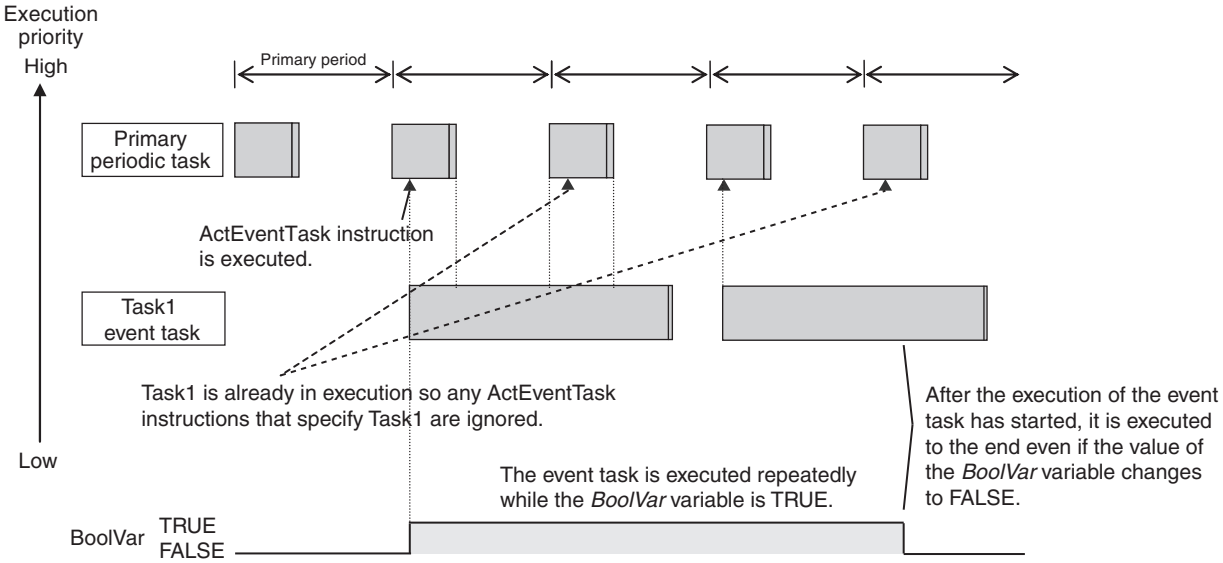


Example 2: Executing an Event Task Repeatedly While the Value of a Variable Matches a Specified Value

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 for the ActEventTask Instruction

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

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

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.

Type of variables		Specification
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.

Data type classification	Data type		Whether it can be specified
Basic data type	**Boolean type, bit string type, integer type, real type**		Allowed.
	Duration, date, time of day, date and time, or text string data		Not supported.
Data type specifications	**Array specifications**	**Arrays**	Not supported.
		Element	Possible. *1
Derivative	**Structures**	**Structures**	Not supported.
		Members	Possible. *2
	Union	**Unions**	Not supported.
		Members	Possible. *2
	Enumerated		Allowed.

*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 FALSE to TRUE
	Change to FALSE TRUE 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

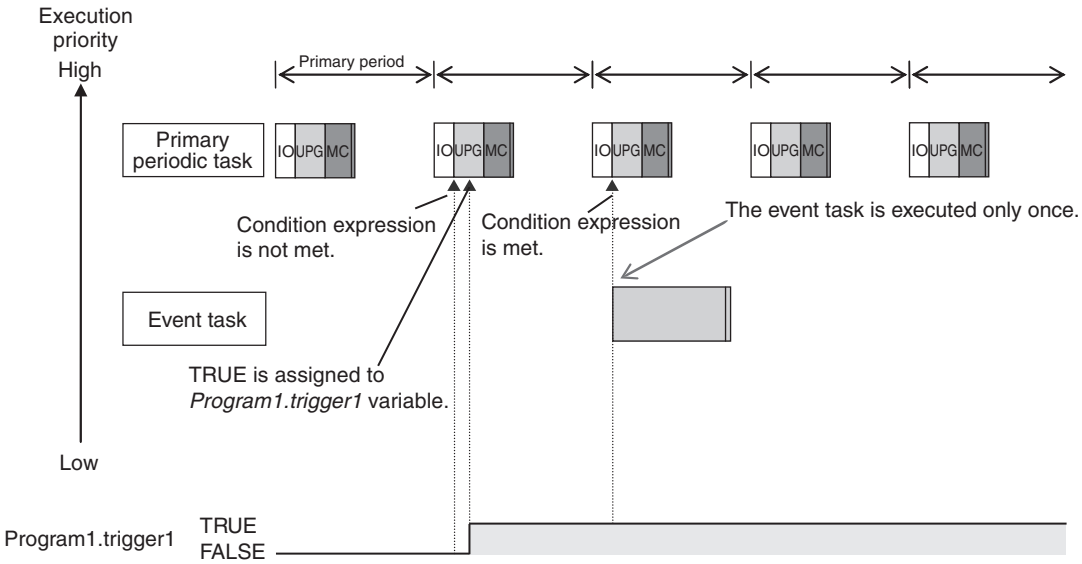
If the variable that you specify for a condition expression is bit string data, integer data, or real data, you must set a comparison constant to compare to the value of the variable. The valid range of

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 6-32 for the valid range of values for each data type.
For example, if the variable that you specify for the condition expression is a BYTE variable, the valid range of comparison constant values is from BYTE#16#00 to BYTE#16#FF.

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

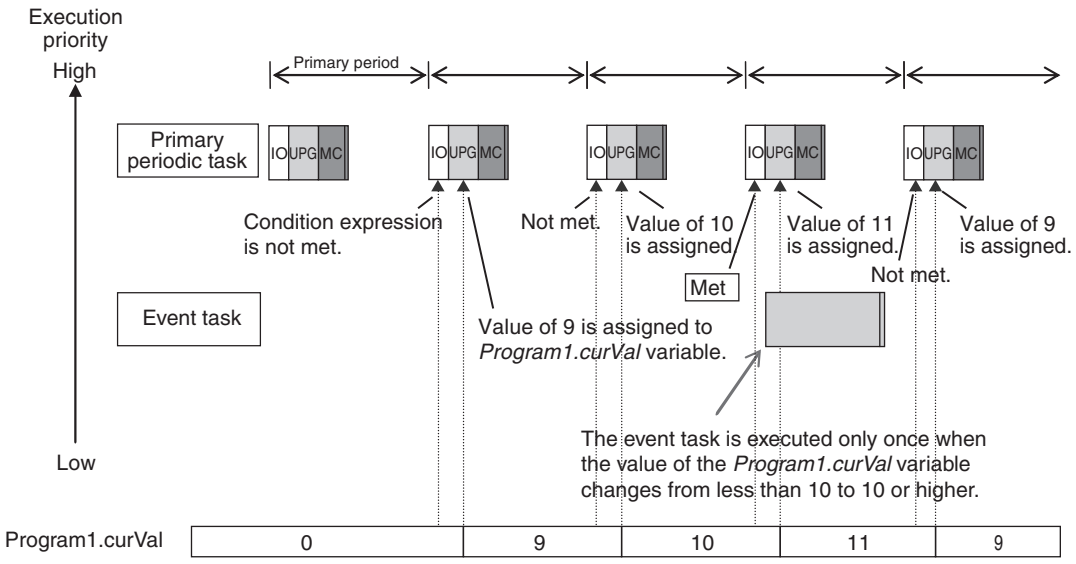
Example 1: Execution Condition for Event Task Set to a Change to TRUE of the *Program1.trigger1* Boolean Variable

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



Example 2: Execution Condition for an Event Task Set to When *Program1.curVal* (INIT variable) ≥ 10

The event task is executed only once when the value of *Program1.curVal* changes from less than 10 to 10 or higher.



Precautions for Correct Use

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

5-4-5 Event Task Execution Timing for NX502 CPU Units

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 tasks and periodic tasks.

If there are 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 *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 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.

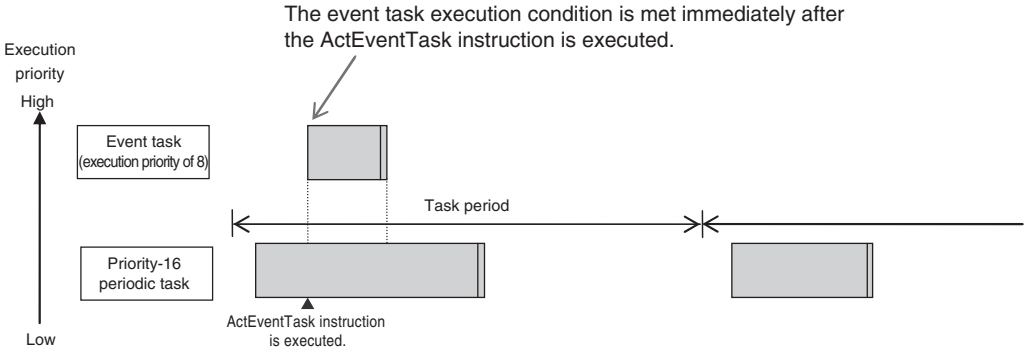
Differences in Execution Timing Based on the Execution Conditions of Event Tasks

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.

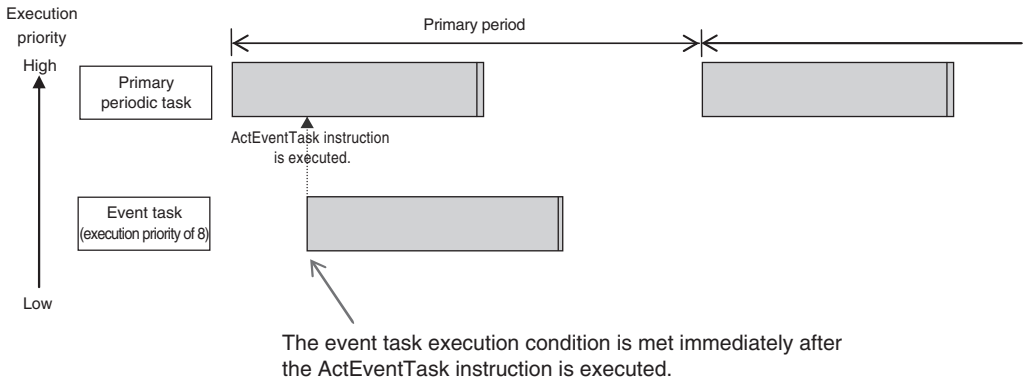
● Execution Timing When the Execution Condition Is an ActEventTask Instruction

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 CPU Units execute event tasks for which the execution condition is met according to the task execution priority.

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



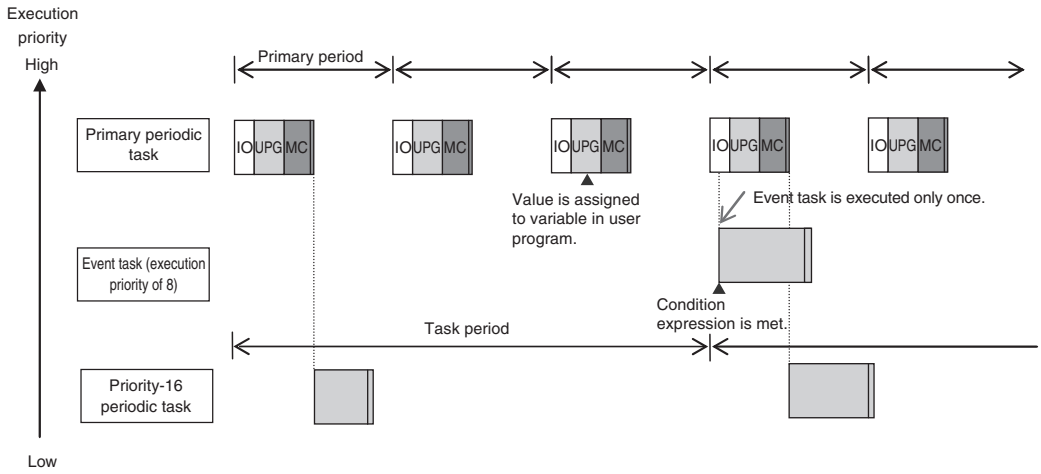
Example 2: Executing an Event Task with an Execution Priority Lower Than the Task That Executes an Instruction



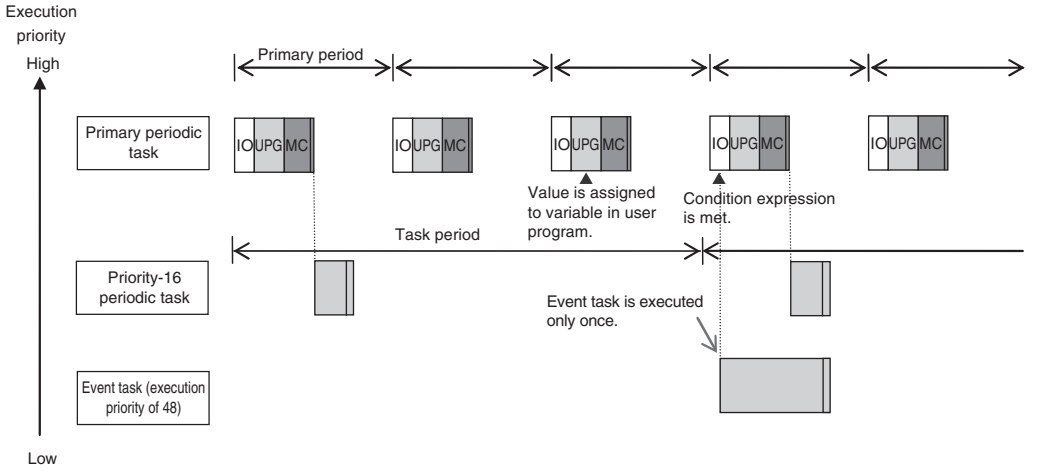
● Execution Timing When the Execution Condition Is a Condition Expression for a Variable

The condition expression is evaluated for a match inside the primary periodic task. The execution condition for an event task is met when it is evaluated to match the condition expression. The CPU Units execute event tasks for which the execution condition is met according to the task execution priority.

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



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



Precautions for Correct Use

- For the NX502 CPU Units, the timing at which the execution condition for an event task is met is the same regardless of whether the condition expression match is triggered by I/O refreshing in the primary periodic task, or by execution of a program that is assigned to the primary periodic task.

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

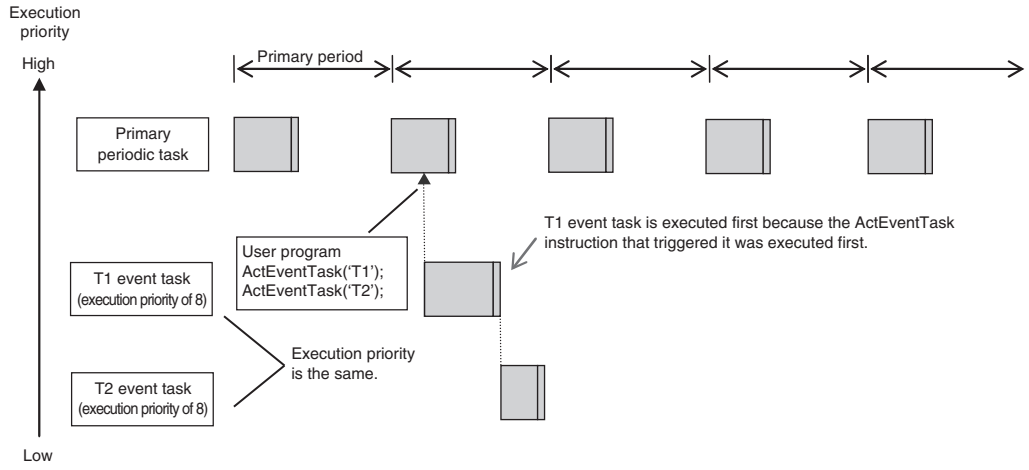
- In order for an event task to be executed, the condition expression must be met in the evaluation after the previous evaluation where the condition expression was not met. This means that even if the status of the condition expression changes from not met to met, if the condition returns to not met before the next evaluation, the event task will not be executed.
- For an NX502 CPU Unit, specify an internal variable defined in the program that is assigned to the primary periodic task if you specify an internal variable with a data size of 64 bits or more to the condition expression. If an internal variable that is not defined in the primary periodic task is specified, the concurrency of variable values may not be ensured. As the result, the match evaluation may not be correctly performed.

Execution Timing for Event Tasks with the Same Execution Priority

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

Example 1: When Two ActEventTask Instructions Are Executed

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



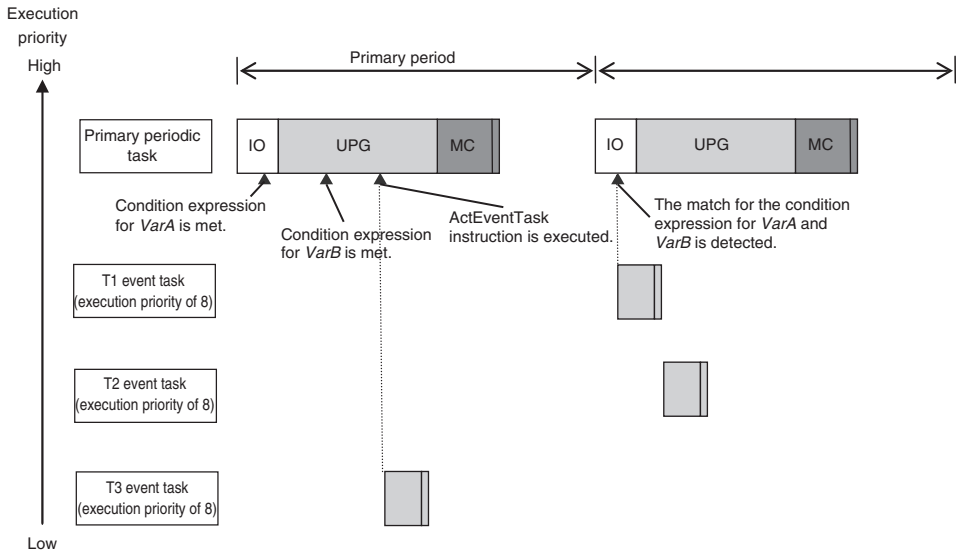
Example 2: When Both Condition Expressions for Variables and the ActEventTask Instruction Are Used

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

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

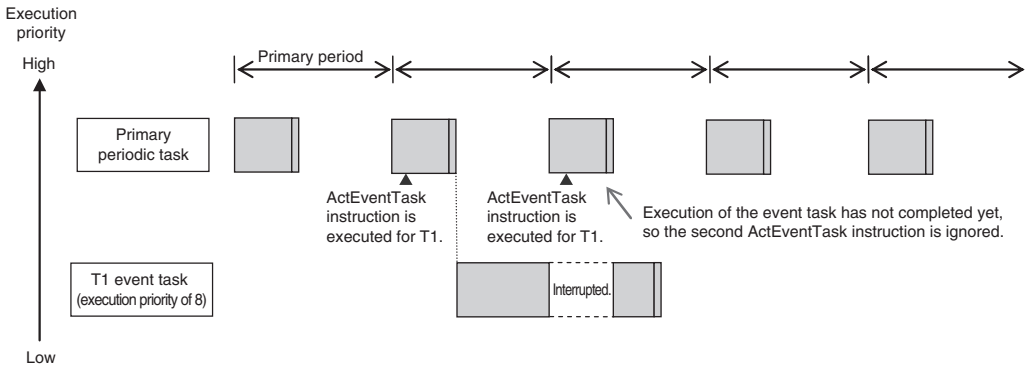
The operation would proceed as described below if the condition expression for *VarA* was met during I/O refreshing, the ActEventTask instruction was executed in the user program, and the condition expression for *VarB* was met during execution of the user program all in the same primary period.

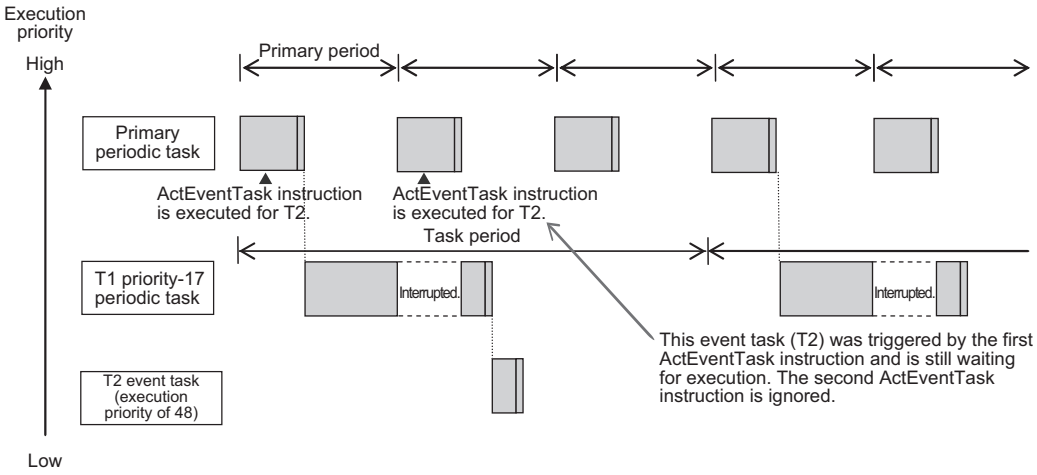
1. The condition expression for *VarA* is met during I/O refreshing.
2. The condition expression for *VarB* is met during execution of the user program.
3. At this point, T1 and T2 are not executed because the condition expressions are not yet evaluated.
4. The ActEventTask instruction is executed in 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.



5-4-6 Operation When Execution Condition Is Met Again Before Execution of the Event Task Is Completed

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




Execution priority
High
Low
Primary period
Primary periodic task
ActEventTask instruction is executed for T2.
ActEventTask instruction is executed for T2.
Task period
T1 priority-17 periodic task
Interrupted.
Interrupted.
T2 event task (execution priority of 48)
This event task (T2) was triggered by the first ActEventTask instruction and is still waiting for execution. The second ActEventTask instruction is ignored.

5-5 Specifications and Basic Operation of Tasks for NX102 CPU Units and NX1P2 CPU Units

This section describes the specifications and basic operation of tasks for the NX102 CPU Units and NX1P2 CPU Units with a multi-core processor.

5-5-1 Specifications of Tasks for NX102 CPU Units and NX1P2 CPU Units

The specifications of tasks are given in the following table.

Item	Specification
Type of task	• Primary periodic task • Periodic task (priority 17 or 18) • Event task (priority 8 or 48)
Numbers of tasks	• Primary periodic task: 1 • Periodic tasks: 0 to 2 tasks *1 • Event tasks: 0 to 32 tasks *2
Number of programs per task	128 max.
Task period of the primary periodic task	• NX102-□□□□ CPU Units: 1 ms to 32 ms (in 250-μs increments) • NX1P2-1□□□□□ CPU Units: 2 ms to 8 ms (in 250-μs increments) • NX1P2-90□□□□ CPU Units: 2 ms to 8 ms (in 250-μs increments) • NX1P2-9B□□□□ CPU Units: 4 ms to 8 ms (in 250-μs increments)

Task periods of periodic tasks	• Priority 17 or 18
	  • NX102-□□□□ CPU Units: 1 ms to 32 ms (in 250-μs increments)
	  • NX1P2-1□□□□□ CPU Units: 2 ms to 100 ms (in 250-μs increments)
	  • NX1P2-90□□□□ CPU Units: 2 ms to 8 ms (in 250-μs increments)
	  • NX1P2-9B□□□□ CPU Units: 4 ms to 8 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: 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.

5-5-2 Guidelines for Separating Tasks for NX102 CPU Units and NX1P2 CPU Units

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-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.

5-5-3 Basic Operation of Tasks for NX102 CPU Units and NX1P2 CPU Units

The NX102 CPU Units and NX1P2 CPU Units cannot execute more than one task at the same time. The order in which tasks are executed depends on the execution priority that is set for each task.

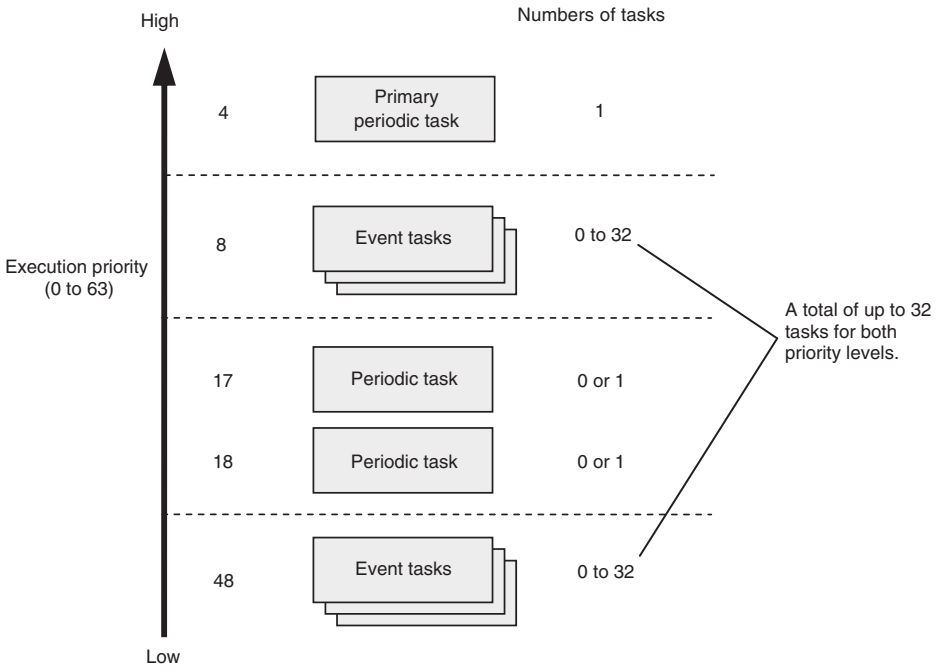
Additional Information

- With an NX102 CPU Unit, you can execute the communications bridge service, tag data link service and system services without being affected by the task execution.
- With an NX1P2 CPU Unit, you can execute the tag data link service, option board service, and system services without being affected by the task execution.

Task Execution Priority

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

Task	Execution priority	Tasks with the same execution priority
Primary periodic task	4	---
Periodic task	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 *5-5-5 Event Task Execution Timing for NX102 CPU Units and NX1P2 CPU Units* on page 5-62 for the order of execution.



Task Periods for the Primary Periodic Task and Periodic Tasks

The CPU Unit repeatedly and cyclically executes the primary periodic task and periodic tasks.
The task periods for periodic tasks must be assigned as integer multiples of the task period of the primary periodic task (called the primary period). Therefore, execution of both tasks will start at the same time every few cycles.
For example, if the primary period is set to 2 ms and the task period of the priority-17 periodic task is set to 8 ms, the execution timing of the primary periodic task and the priority-17 periodic task is synchronized after each four executions of the primary periodic task.

Additional Information

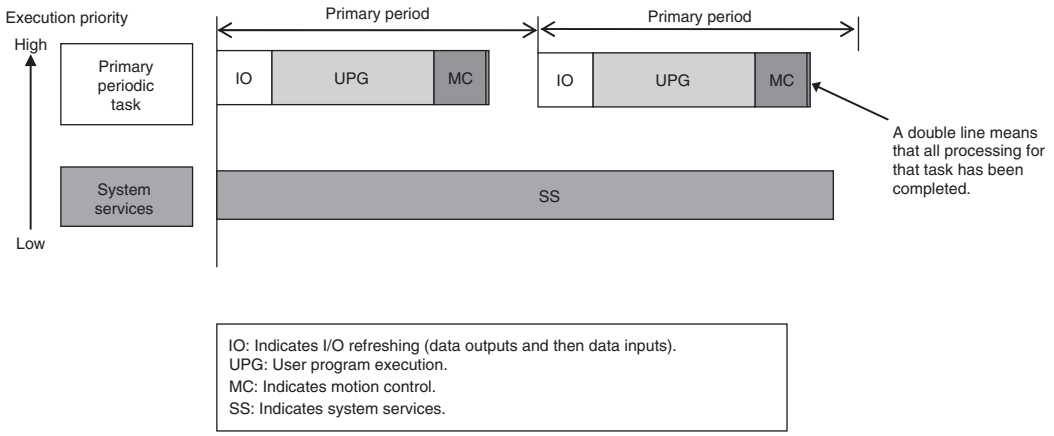
An event task is not executed periodically. Instead, it is executed only once when the specified execution condition is met. Therefore, execution of an event task depends on when its execution condition is met and on its execution priority.

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 *5-5-5 Event Task Execution Timing for NX102 CPU Units and NX1P2 CPU Units* on page 5-62 for the order of execution of event tasks.

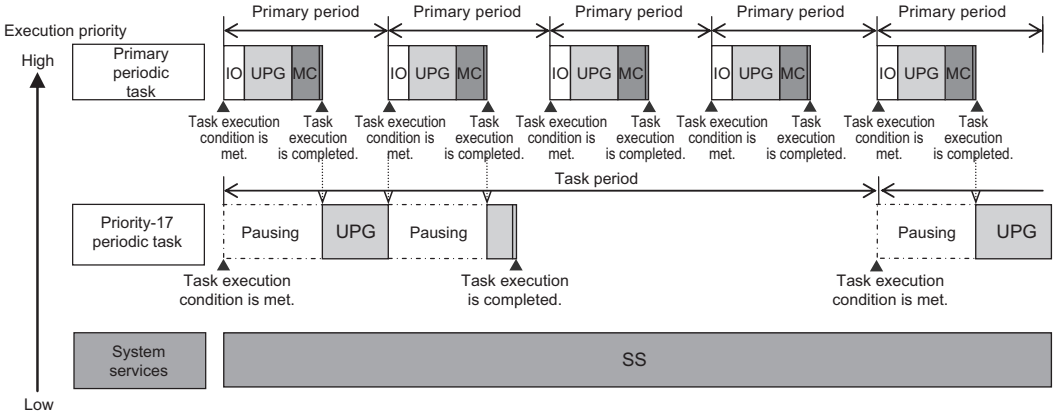
● Projects with Only the Primary Periodic Task

The primary periodic task is executed every primary period.
The system service shown in this figure refers to non-task related processing, such as communications processing, that is performed by the CPU Unit. Refer to *Processing Performed in System Services* on page 5-98 for details on the system services.



● Project with the Primary 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-17 periodic task has a lower execution priority than the primary periodic task, so it is executed when the primary periodic task is not being executed.
- In this example, the task period for the priority-17 periodic task is set to four times the primary period. This means that execution of the priority-17 periodic task will start at the same time once every four primary periods.
- 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 *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details.

Tasks and Operating Modes

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

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

Precautions for Correct Use

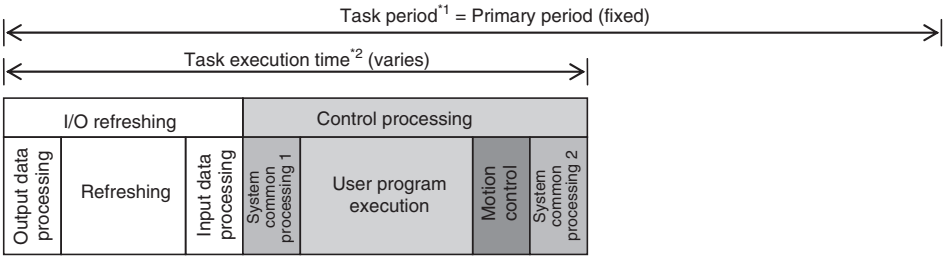
- Even if the execution condition for an event task is already met when you change the operating mode to RUN mode, the event task will not be executed. An event task is executed only when its execution condition changes from not met to met during RUN mode.
- Even in RUN mode, an event task is not executed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

The Processing Performed in Each Task

● Primary Periodic Task

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

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



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

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

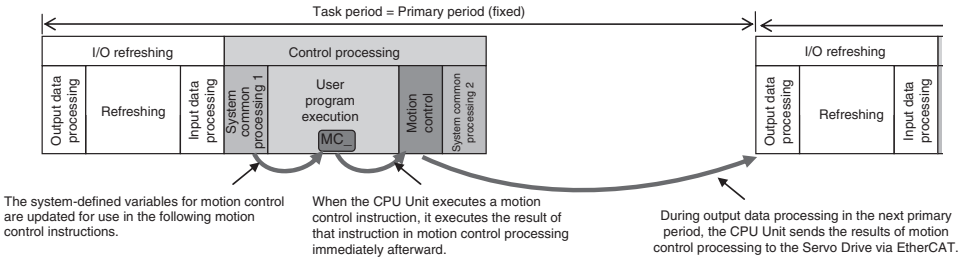
Processing		Processing contents
I/O refreshing	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Whether the condition expression for event task execution is met or not is determined. • Input refresh data is loaded from Input Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1		• Processing for exclusive control of variables in tasks 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.

Processing	Processing contents
Motion control*2	• The motion control commands from the motion control instructions in the user program assigned to the primary 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 Servo Drive status in the system-defined variables for motion control are updated.

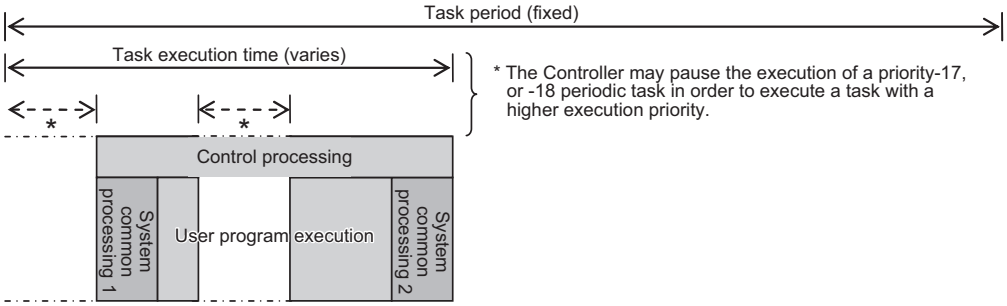
*2. When there are motion control instructions in user program execution in the primary periodic task, the CPU Unit executes the results from those instructions immediately afterward in motion control processing as shown below. The CPU Unit outputs the results to the Servo Drives during I/O refreshing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

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



● 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 2 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.

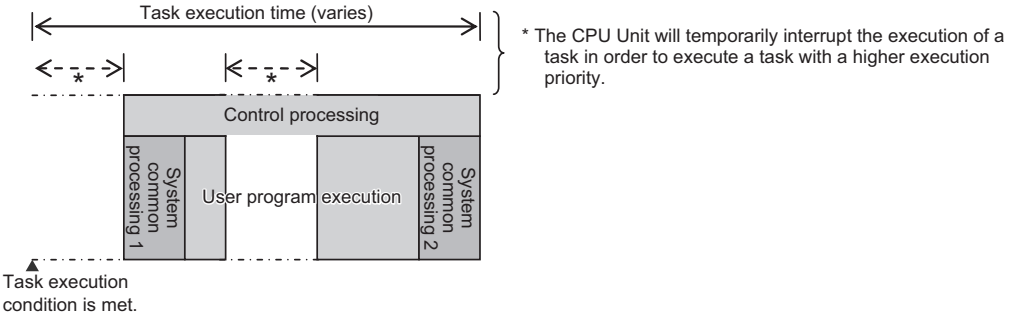
Processing	Processing contents
System common processing 2	• Processing for exclusive control of variables in tasks is performed (when refreshing tasks are set). • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the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

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 *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details on exclusive control.

5-5-4 Event Task Execution Conditions for NX102 CPU Units and NX1P2 CPU Units

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 condition	Event task execution timing	Reason for use
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 5-26 for the timing of when the value of the specified variable is checked to see if the specified condition expression is met.

Precautions for Safe Use

If the following variables are specified for a condition expression when the execution condition is a condition expression for a variable, event tasks may not be executed when conditions are met or event tasks may be executed when conditions are not met.

- Structure members whose data size is 16 bits or more, except for system-defined variables for motion control
- Array elements whose data size is 16 bits or more

When the above variables are specified and the match evaluation is performed, perform either of the followings.

- Copy the above variables to the internal variables with a basic data type other than a data size of 64 bits, and access the copied internal variables.
- Use the settings for exclusive control of variables in tasks, and set the primary periodic task as a refreshing task.

Executing Event Tasks for the ActEventTask Instruction

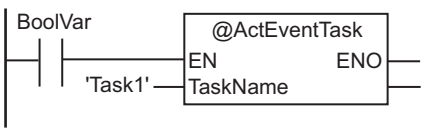
When the ActEventTask (Execute Event Task) instruction is executed in the user program, the specified event task is executed once. Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for the detailed specifications of the ActEventTask instruction.

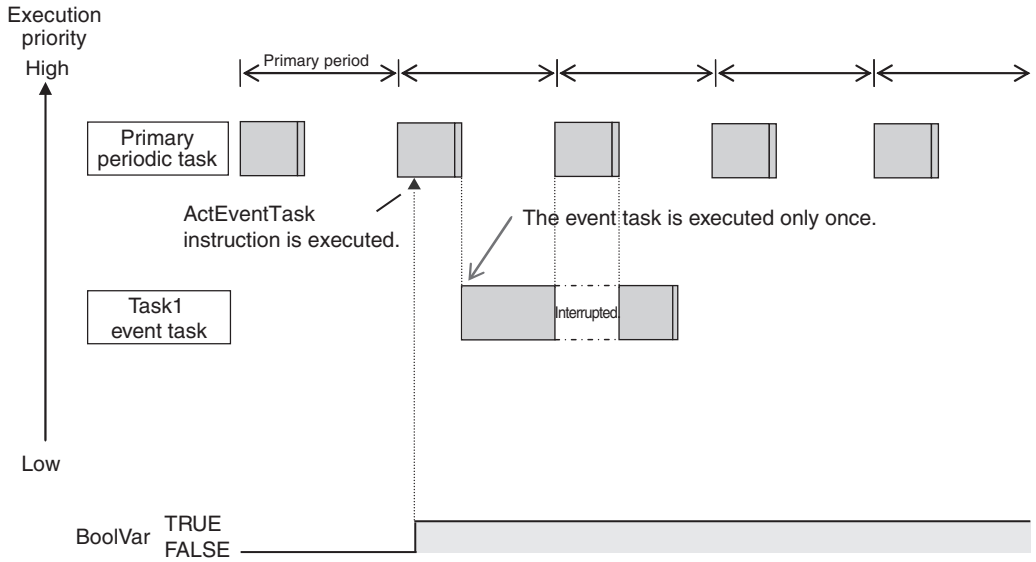
Using the ActEventTask instruction to execute event tasks makes it easy to see which event tasks are executed. Also, this method is also effective when the execution condition for the event task may change before meeting the condition expression for the variable is determined.

● Example of User Programming Using the ActEventTask Instruction

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

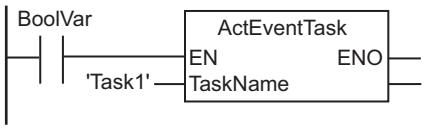
In the following example, the upward differentiation option is used for the ActEventTask instruction. This causes the Task1 event task to be executed only once when the *BoolVar* BOOL variable changes to TRUE.

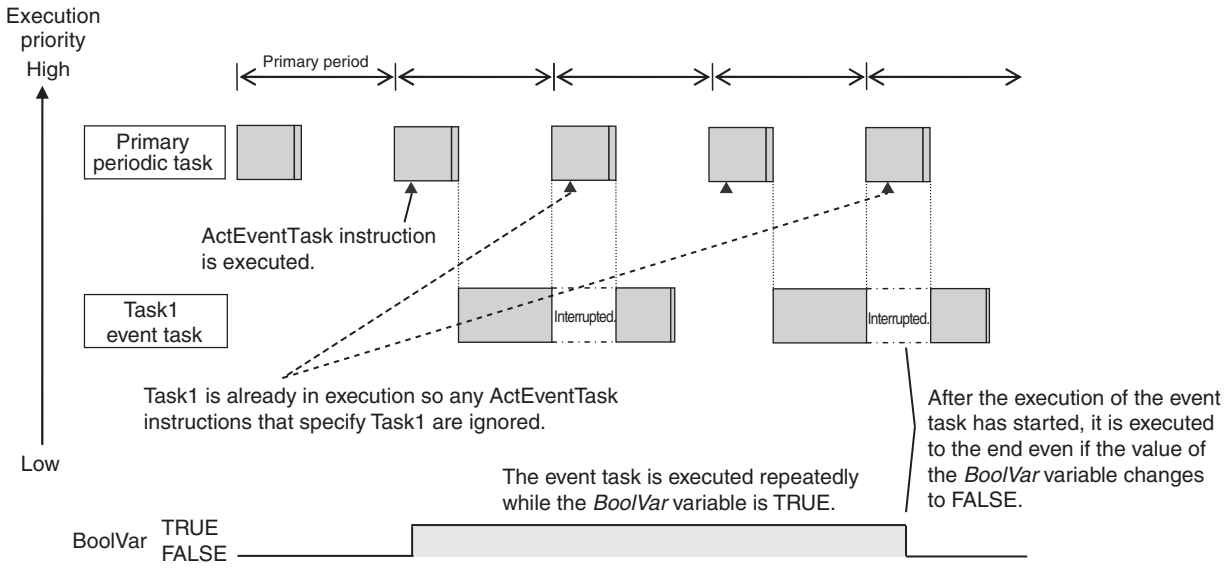




Example 2: Executing an Event Task Repeatedly While the Value of a Variable Matches a Specified Value

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





Executing Event Tasks When Condition Expressions for Variables Are Met

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

● Variables for Which You Can Specify Condition Expressions

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

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

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

● Data Types of Variables for Condition Expressions

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

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

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

● Condition Expressions That You Can Specify

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

Data type	Possible condition expressions
Boolean, Boolean array elements, Boolean structure members, and Boolean union members	Change to TRUE FALSE to TRUE
	Change to FALSE TRUE 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.

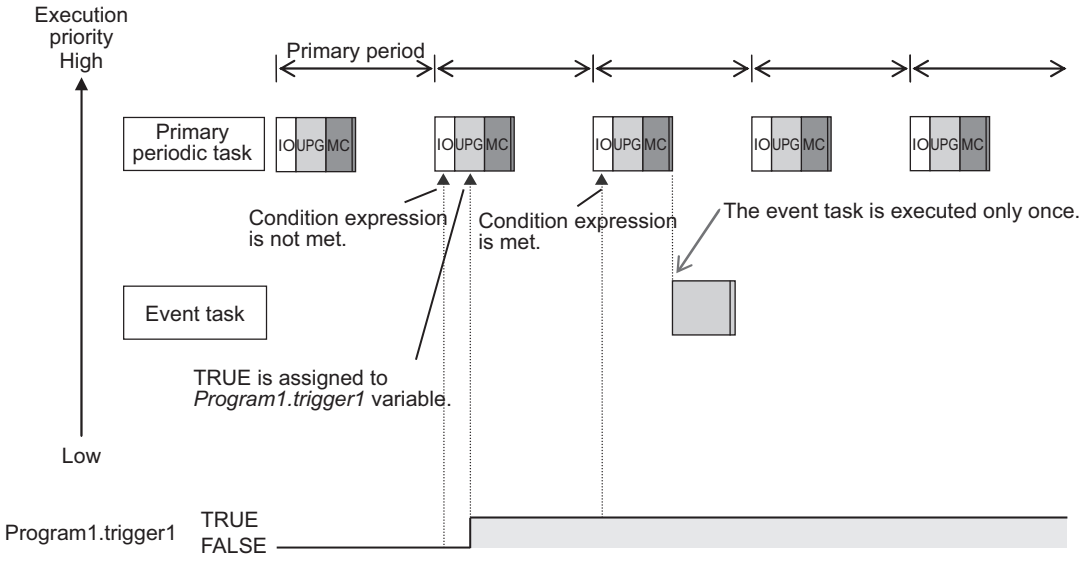
Refer to *Basic Data Types and Derivative Data Types* on page 6-32 for the valid range of values for each data type.

For example, if the variable that you specify for the condition expression is a BYTE variable, the valid range of comparison constant values is from BYTE#16#00 to BYTE#16#FF.

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

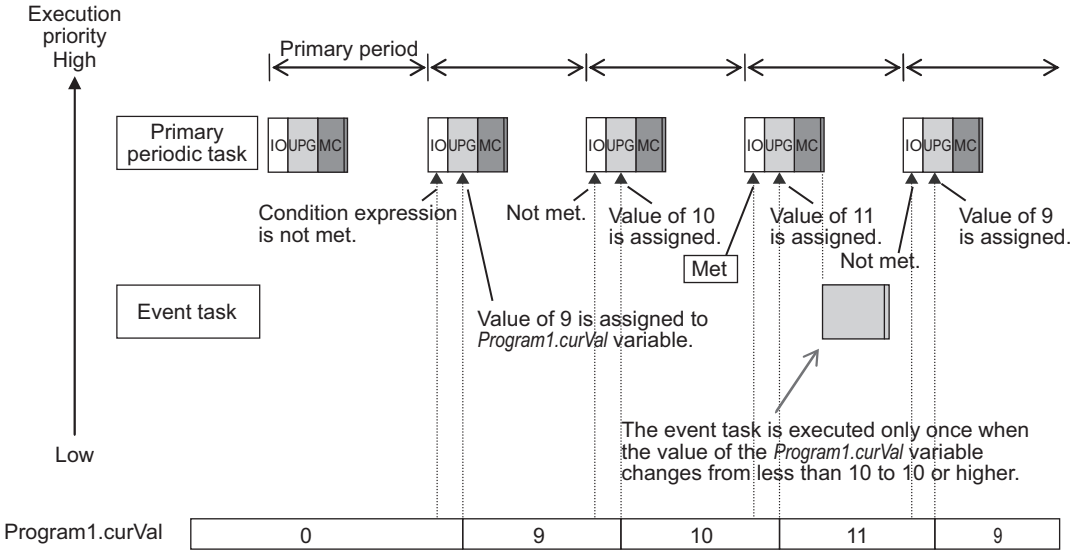
Example 1: Execution Condition for Event Task Set to a Change to TRUE of the *Program1.trigger1* Boolean Variable

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



Example 2: Execution Condition for an Event Task Set to When *Program1.curVal* (INIT variable) ≥ 10

The event task is executed only once when the value of *Program1.curVal* changes from less than 10 to 10 or higher.



Precautions for Correct Use

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

5-5-5 Event Task Execution Timing for NX102 CPU Units and NX1P2 CPU Units

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 *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 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.

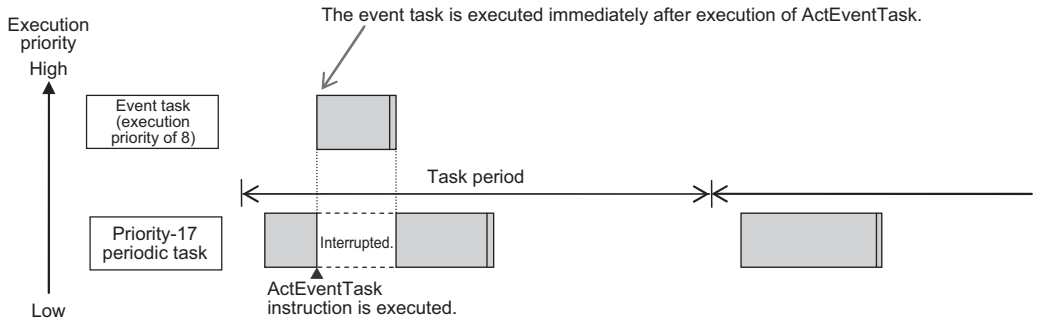
Differences in Execution Timing Based on the Execution Conditions of Event Tasks

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.

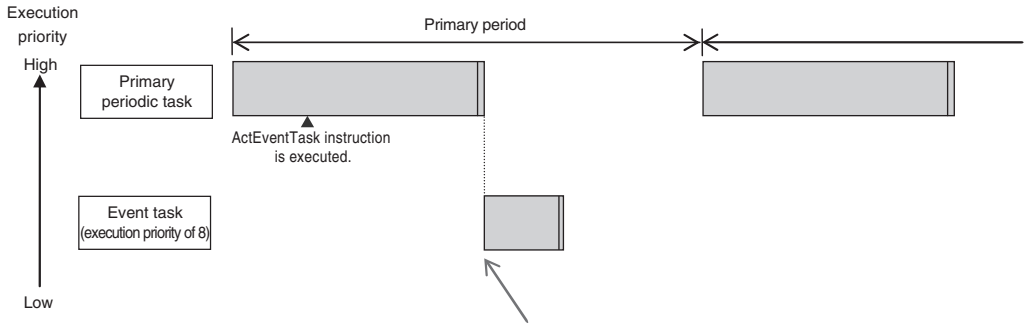
● Execution Timing When the Execution Condition Is an ActEventTask Instruction

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 NX1P2 CPU Units execute event tasks for which the execution condition is met according to the task execution priority.

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



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

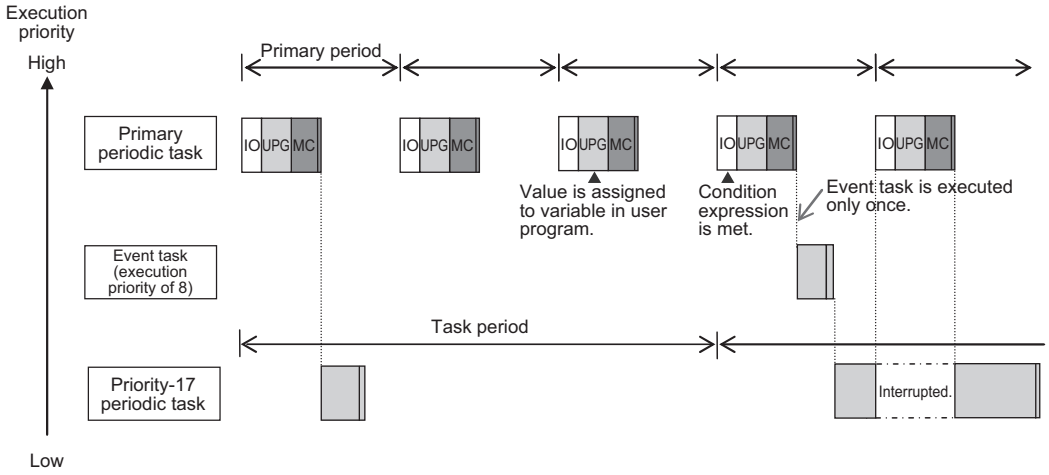


● Execution Timing When the Execution Condition Is a Condition Expression for a Variable

The condition expression is evaluated for a match inside the primary periodic task. The execution condition for an event task is met when it is evaluated to match the condition expression. The CPU Units execute event tasks for which the execution condition is met according to the task execution priority.

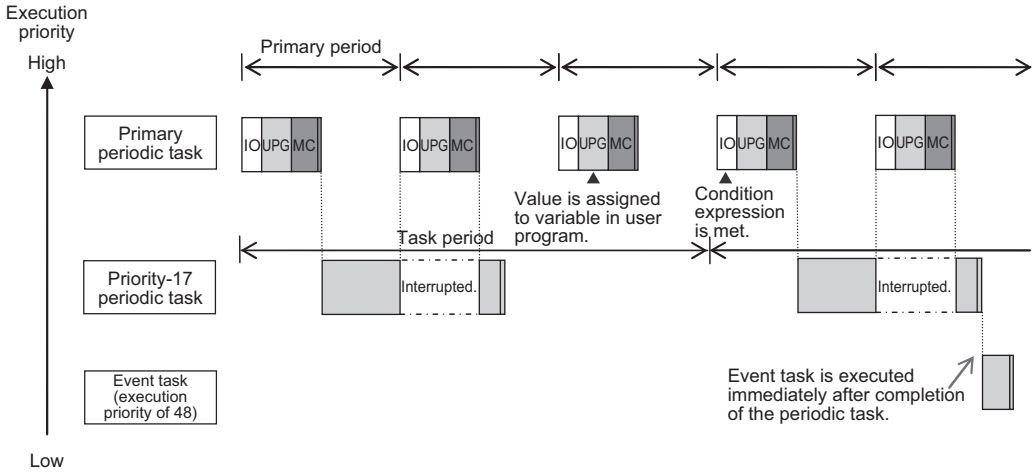
Example 1: Project with a Priority-17 Periodic Task and an Event Task Execution with a Priority of 8

The execution priority of the event task (execution priority of 8) is higher than the execution priority of the priority-17 periodic task. The priority-17 periodic task is therefore executed after the event task is executed.



Example 2: Project with a Priority-17 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-17 periodic task. The event task is therefore executed after the priority-17 periodic task is executed.



Precautions for Correct Use

- For the NX102 CPU Units and NX1P2 CPU Units, the timing at which the execution condition for an event task is met is the same regardless of whether the condition expression match is triggered by I/O refreshing in the primary periodic task, or by execution of a program that is assigned to the primary periodic task.

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

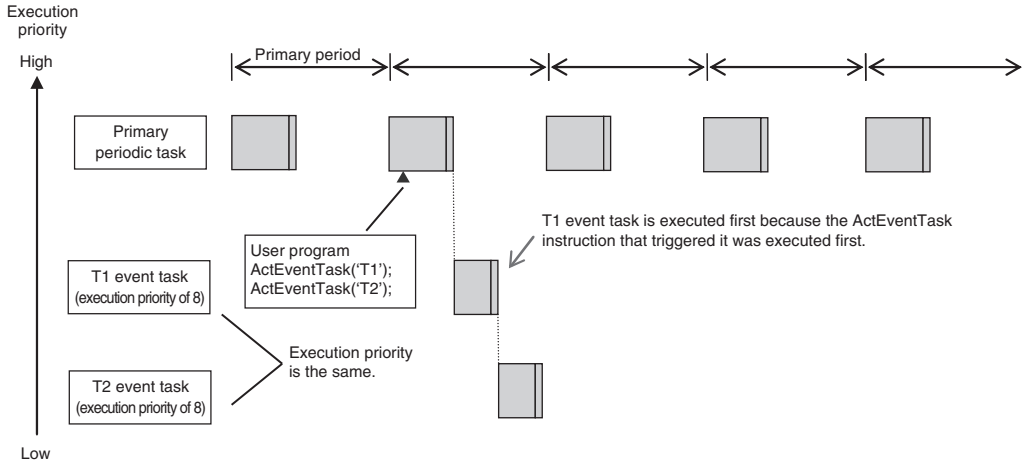
- In order for an event task to be executed, the condition expression must be met in the evaluation after the previous evaluation where the condition expression was not met. This means that even if the status of the condition expression changes from not met to met, if the condition returns to not met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

Execution Timing for Event Tasks with the Same Execution Priority

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

Example 1: When Two ActEventTask Instructions Are Executed

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



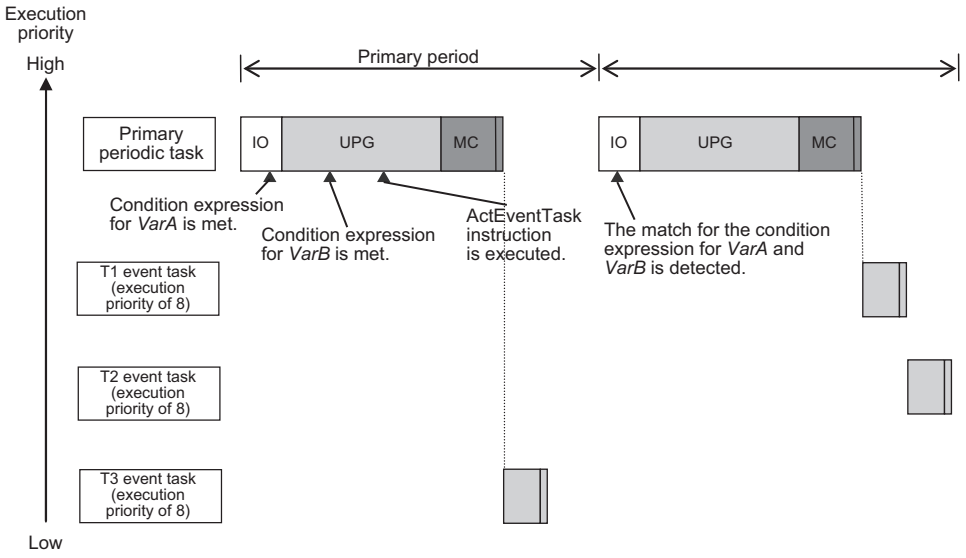
Example 2: When Both Condition Expressions for Variables and the ActEventTask Instruction Are Used

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

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

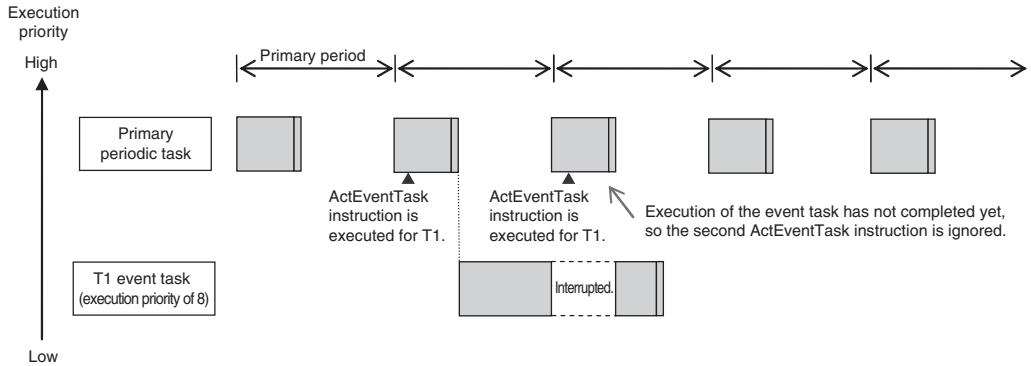
The operation would proceed as described below if the condition expression for *VarA* was met during I/O refreshing, the ActEventTask instruction was executed in the user program, and the condition expression for *VarB* was met during execution of the user program all in the same primary period.

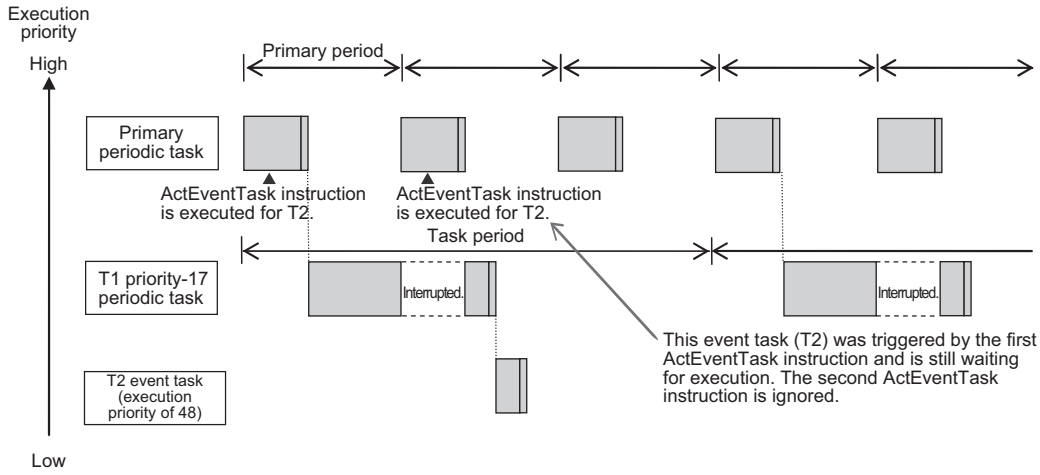
1. The condition expression for *VarA* is met during I/O refreshing.
2. The condition expression for *VarB* is met during execution of the user program.
3. At this point, T1 and T2 are not executed because the condition expressions are not yet evaluated.
4. The ActEventTask instruction is executed in 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.



5-5-6 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.




Execution priority
High
Low
Primary period
Primary periodic task
ActEventTask instruction is executed for T2.
ActEventTask instruction is executed for T2.
Task period
T1 priority-17 periodic task
Interrupted.
Interrupted.
T2 event task (execution priority of 48)
This event task (T2) was triggered by the first ActEventTask instruction and is still waiting for execution. The second ActEventTask instruction is ignored.

5-6 Specifications and Basic Operation of Tasks for NJ-series Controllers

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

5-6-1 Specifications of Tasks for NJ-series Controllers

The specifications of tasks are given in the following table.

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

*1. There can be no more than one task with each of the following execution priorities: 16, 17, and 18.
*2. There can be up to 32 tasks with each of the following priorities as long as there are no more than a total of 32 tasks with these priorities: 8 and 48.
*3. With the NJ301-□□□□, you can use this setting with unit version 1.03 or later.
You cannot use this setting with the NJ101-□□□□.

● Valid Task Periods for Periodic Tasks

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

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

5-6-2 Guidelines for Separating Tasks for NJ-series Controllers

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

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

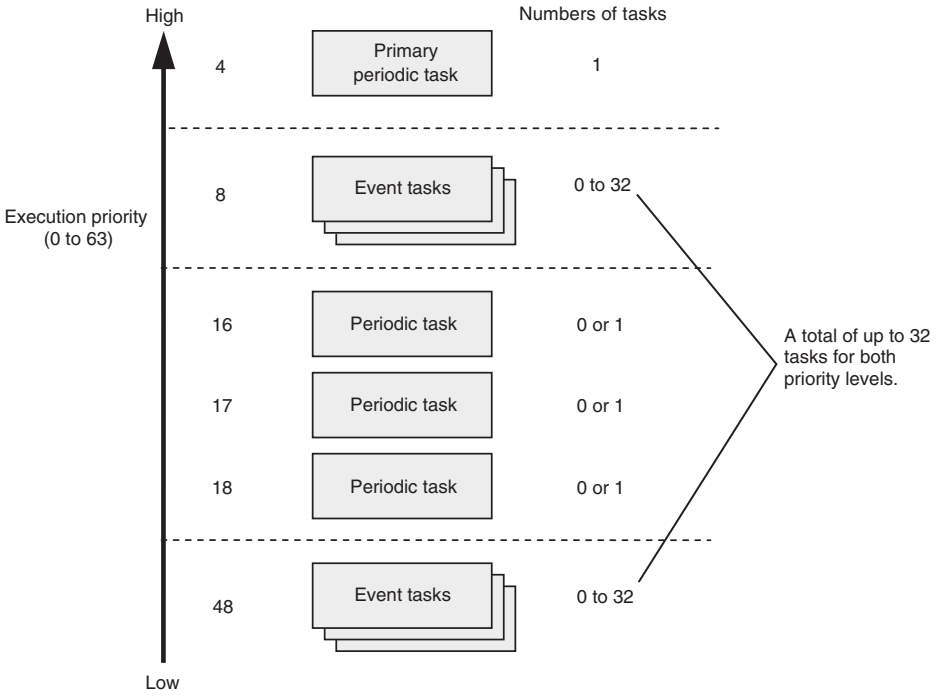
5-6-3 Basic Operation of Tasks for NJ-series Controllers

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

Task Execution Priority

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

Task	Execution priority	Tasks with the same execution priority
Primary periodic task	4	---
Periodic task	16, 17, or 18	You cannot set the same execution priority for more than one task.
Event task	8 or 48	You can set the same execution priority for more than one event task. Refer to *5-6-5 Event Task Execution Timing for NJ-series Controllers* on page 5-81 for the order of execution.



Task Periods for the Primary Periodic Task and Periodic Tasks

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

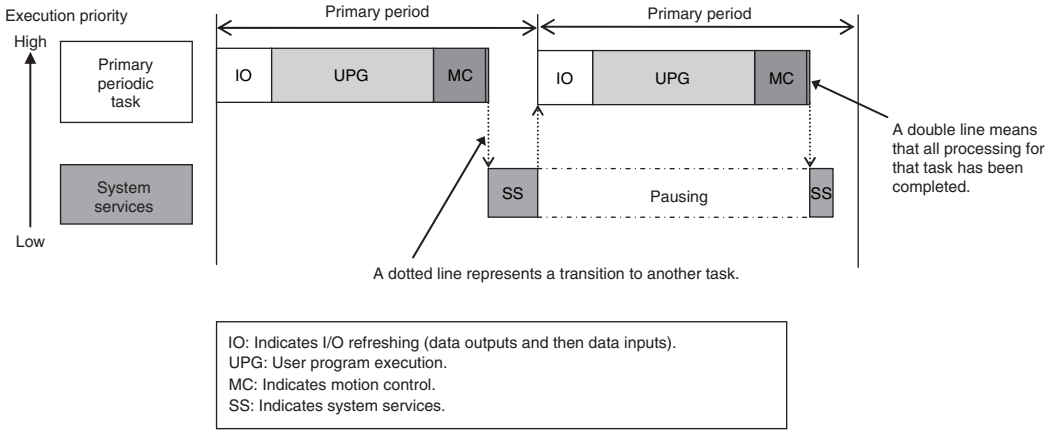
Examples of Execution Order for Tasks

This section gives a few examples of the execution order for the primary periodic task and periodic tasks.
Refer to *5-6-5 Event Task Execution Timing for NJ-series Controllers* on page 5-81 for the order of execution of event tasks.

● Projects with Only the Primary Periodic Task

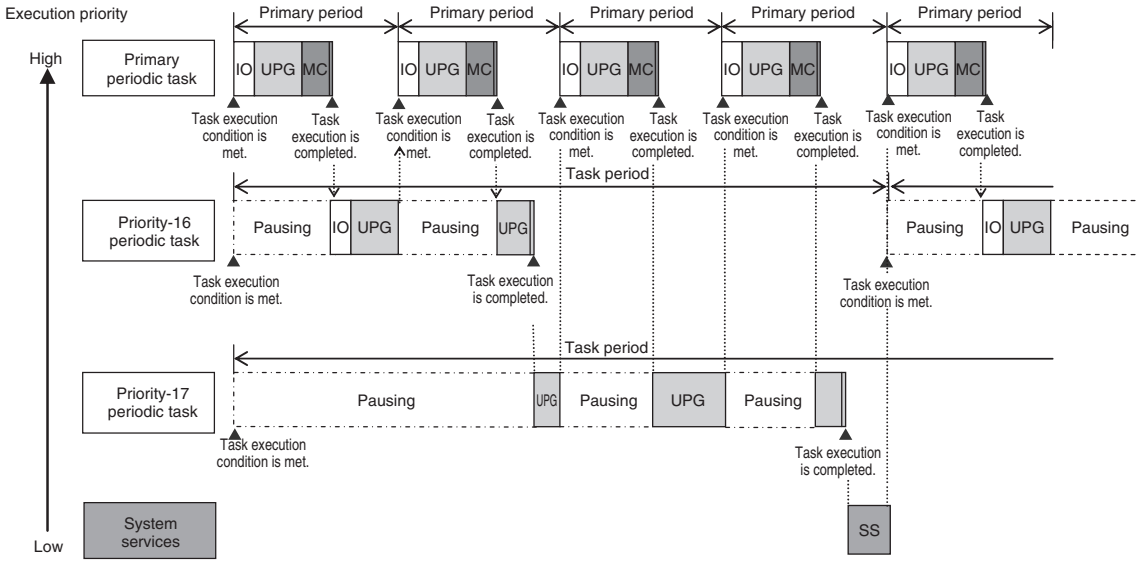
The primary periodic task is executed every primary period.
The system service shown in this figure refers to non-task related processing, such as communications processing, that is performed by the CPU Unit. System services are executed in the unused

time between execution of the tasks. Refer to *Processing Performed in System Services* on page 5-98 for details on the system services.



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

- The primary periodic task has the highest execution priority, so it is always executed in the primary period.
- The priority-16 periodic task has a lower execution priority than the primary periodic task, so it is executed when the primary periodic task is not being executed.
- The priority-17 periodic task has an even lower execution priority, so it is executed when the above two tasks are not under execution.
- In this example, the task period for the priority-16 periodic task is set to four times the primary period. This means that once ever four primary periods, execution of the primary periodic task and the priority-16 periodic task will start at the same time.
- System services are executed in the unused time between execution of the tasks.

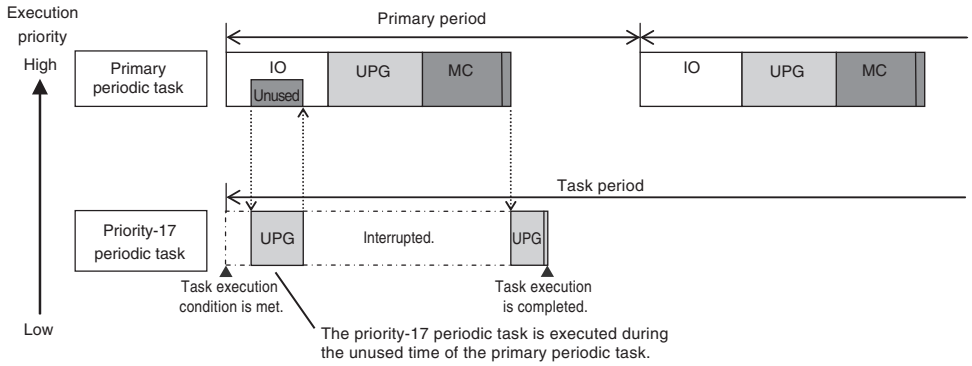


Precautions for Correct Use

If you have multiple tasks that read and write to the same variables, make sure to use exclusive control of variables between the tasks. Otherwise, a task other than the one currently in execution may change the variable values.
Refer to *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details.

Executing Other Tasks during the Unused Time of a Task

A task with a higher execution priority is never interrupted to start execution of a task with a lower execution priority. However, for NJ-series CPU Units, if unused time occurs during a task with higher priority, that time may be used to start execution of a task with lower priority. An example of this is the time spent waiting for a data transfer to be completed in an I/O refresh process. As soon as processing resumes for the task with higher priority, the task with lower priority will be interrupted. This processing order is illustrated in the following figure.



Precautions for Correct Use

Even if unused time occurs in the primary periodic task, the priority-16 periodic task is always executed after the primary periodic task is completed.
However, for NJ-series CPU Units, this restriction does not apply to the priority-17 or priority-18 periodic task.
This restriction also does not apply to I/O refreshing in the priority-16 periodic task. I/O refreshing for the priority-16 periodic task may be executed during the unused time of the primary periodic task.

Tasks and Operating Modes

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

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

Precautions for Correct Use

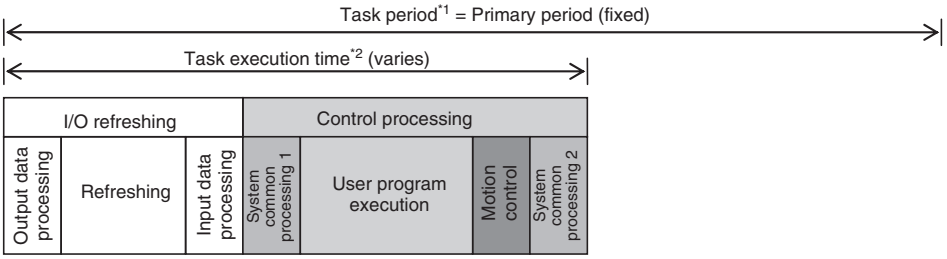
- Even if the execution condition for an event task is already met when you change the operating mode to RUN mode, the event task will not be executed. An event task is executed only when its execution condition changes from not met to met during RUN mode.
- Even in RUN mode, an event task is not executed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

The Processing Performed in Each Task

● Primary Periodic Task

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

In the specified period, this task performs system common processing, I/O refreshing, user program execution, and motion control. The primary periodic task performs motion control processing (MC).



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

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

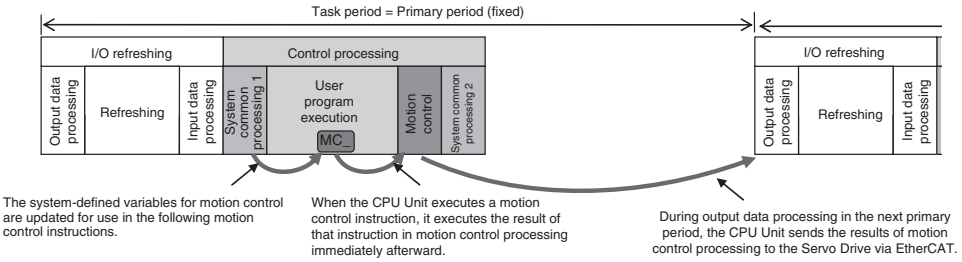
Processing		Processing contents
I/O refreshing	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Input refresh data is loaded from Input Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1		• Processing for exclusive control of variables in tasks is performed (when accessing tasks are set). • Motion input processing is performed.*1 • Data trace processing (sampling and trigger checking) is performed. • Whether the condition expression for event task execution is met or not is determined.
User program execution		• Programs assigned to tasks are executed in the order that they are assigned.
Motion control*2		• The motion control commands from the motion control instructions in the user program execution are executed. • Processing the motion outputs for I/O refreshing in the next primary periodic task.

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

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

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

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



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

Refer to *5-12-3 System Input and Output Response Times* on page 5-137 for details.

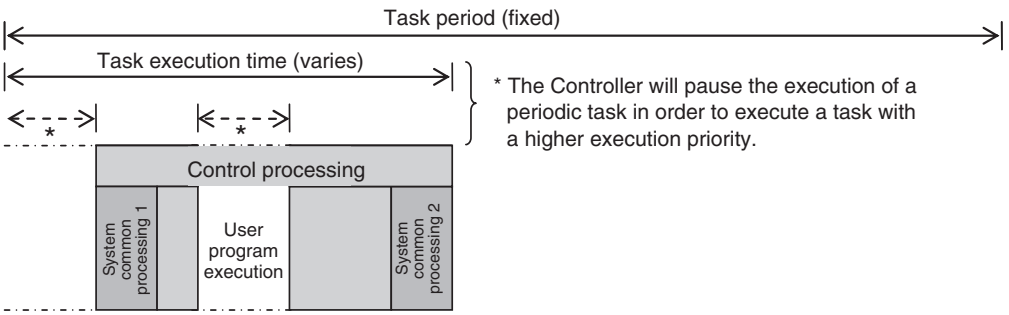
● Periodic Tasks

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

The priority-16 periodic task can also refresh I/O.

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

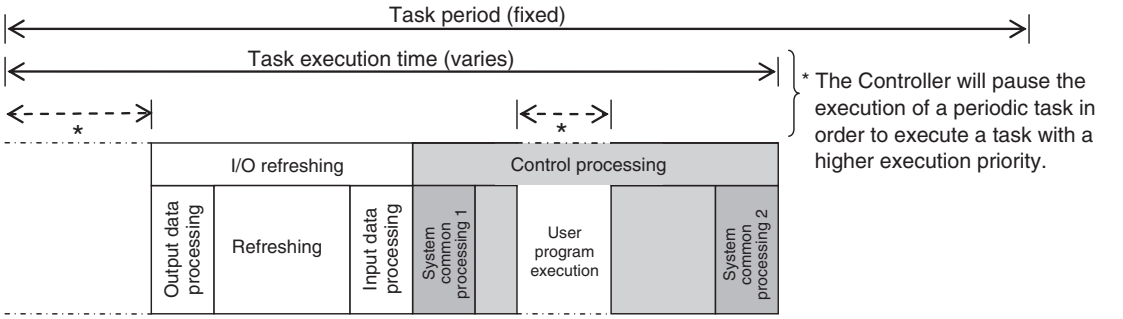
- Periodic Tasks That Do Not Control I/O



Processing	Processing contents
System common processing 1	• Processing for exclusive control of variables in tasks 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.

- Periodic Tasks That Control I/O



Processing		Processing contents
I/O refreshing*1	Output data processing	• Output refresh data is created for Output Units that refresh I/O. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
	Refreshing	• This process exchanges data with I/O.
	Input data processing	• Input refresh data is loaded from Input Units that refresh I/O. *2 • 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). • 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.

*1. I/O Refreshing for CJ-series Units is executed.

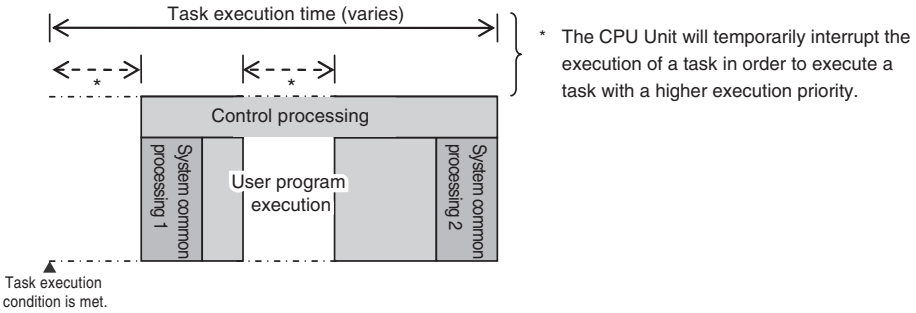
*2. If the input refresh data loaded in the priority-16 periodic task is accessed in the primary periodic task for the NJ-series CPU Unit, copy the input refresh data to access in the priority-16 periodic task to a global variable and access the values of the copied global variable in the primary periodic task. Ensure concurrency of variable values for the global variable. Refer to *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details.
If you do not perform the above processing, when an execution of the primary periodic task is started during I/O refreshing in the priority-16 periodic task, the concurrency of values of the input refresh data accessed in the primary periodic task may not be ensured.

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 *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details on exclusive control.

5-6-4 Event Task Execution Conditions for NJ-series Controllers

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

Execution condition	Event task execution timing	Reason for use
Execution with the 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 5-82 for the timing of when the value of the specified variable is checked to see if the specified condition expression is met.

Precautions for Safe Use

If the following variables are specified for a condition expression when the execution condition is a condition expression for a variable, event tasks may not be executed when conditions are met or event tasks may be executed when conditions are not met.

- Structure members whose data size is 16 bits or more, except for system-defined variables for motion control
- Array elements whose data size is 16 bits or more

When the above variables are specified and the match evaluation is performed, perform either of the followings.

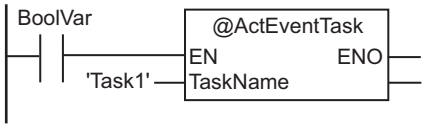
- Copy the above variables to the internal variables with a basic data type other than a data size of 64 bits, and access the copied internal variables.
- Use the settings for exclusive control of variables in tasks, and set the primary periodic task as a refreshing task.

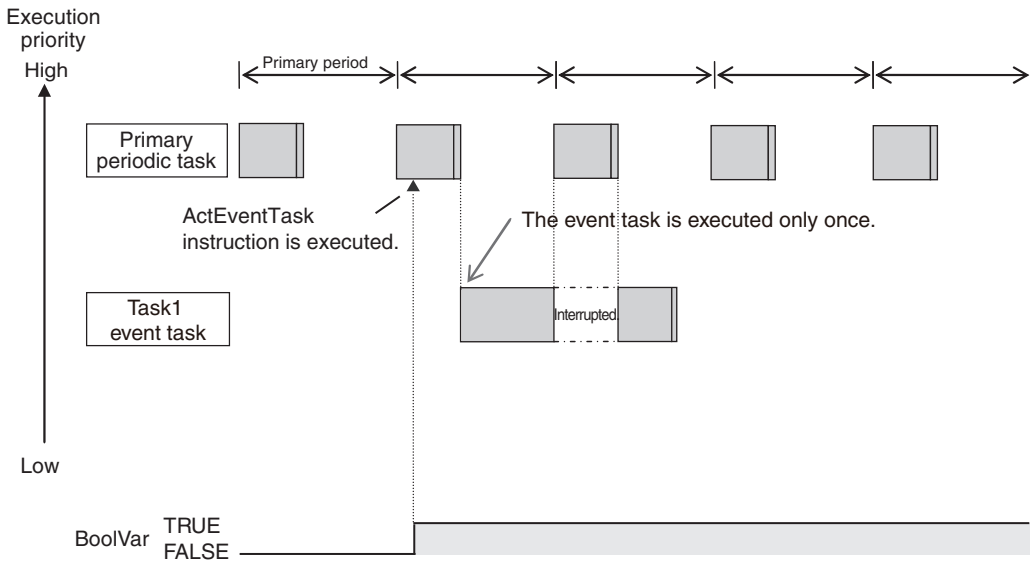
Executing Event Tasks for the ActEventTask Instruction

When the ActEventTask (Execute Event Task) instruction is executed in the user program, the specified event task is executed once. Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for the detailed specifications of the ActEventTask instruction.
Using the ActEventTask instruction to execute event tasks makes it easy to see which event tasks are executed. Also, this method is 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

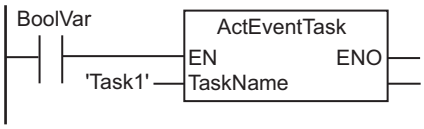
Example 1: Executing an Event Task Only Once When the Value of a Variable Changes
In the following example, the upward differentiation option is used for the ActEventTask instruction. This causes the Task1 event task to be executed only once when the *BoolVar* BOOL variable changes to TRUE.

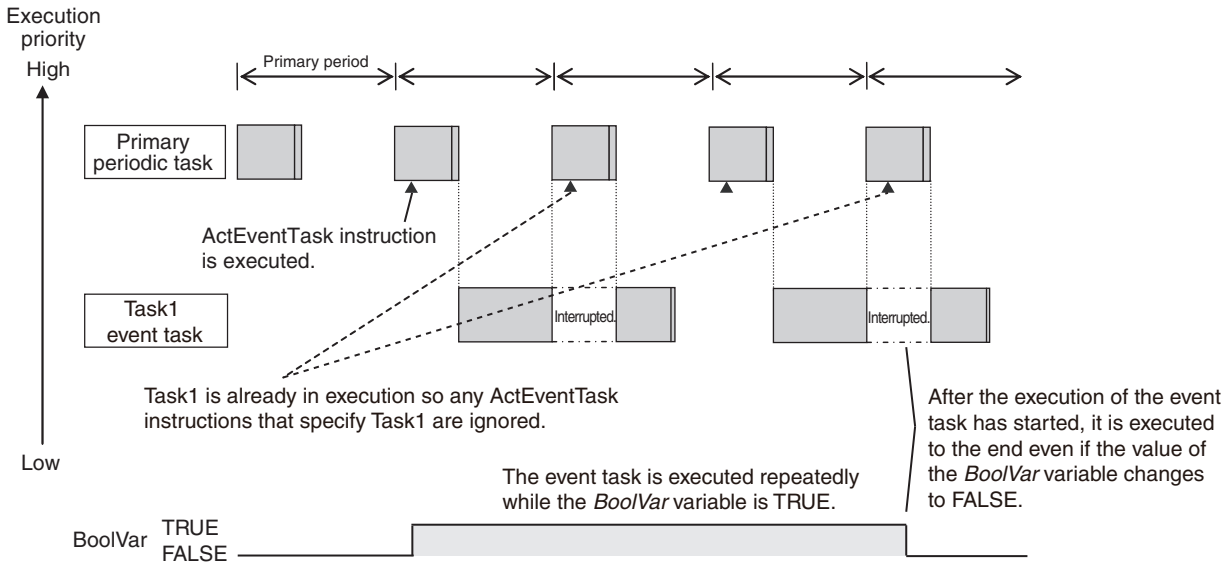




Example 2: Executing an Event Task Repeatedly While the Value of a Variable Matches a Specified Value

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.

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

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

*2. In-out variables cannot be used.

● Data Types of Variables for Condition Expressions

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

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

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

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

● Condition Expressions That You Can Specify

The condition expressions that you can specify depend on the data type of the variable that you specify for the condition expression.

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

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

● Valid Range of Comparison Constants

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

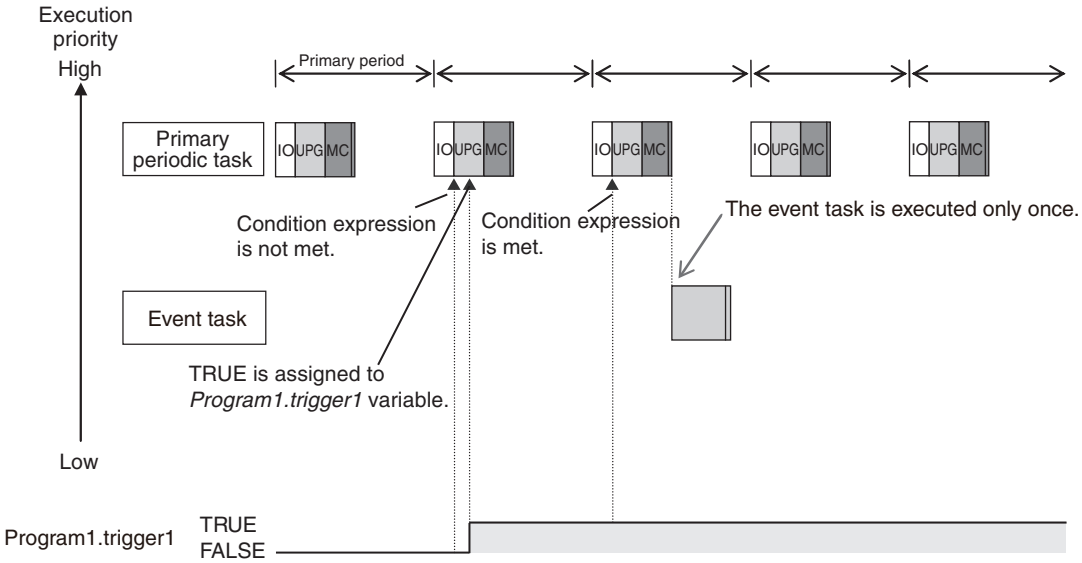
Refer to *Basic Data Types and Derivative Data Types* on page 6-32 for the valid range of values for each data type.

For example, if the variable that you specify for the condition expression is a BYTE variable, the valid range of comparison constant values is from BYTE#16#00 to BYTE#16#FF.

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

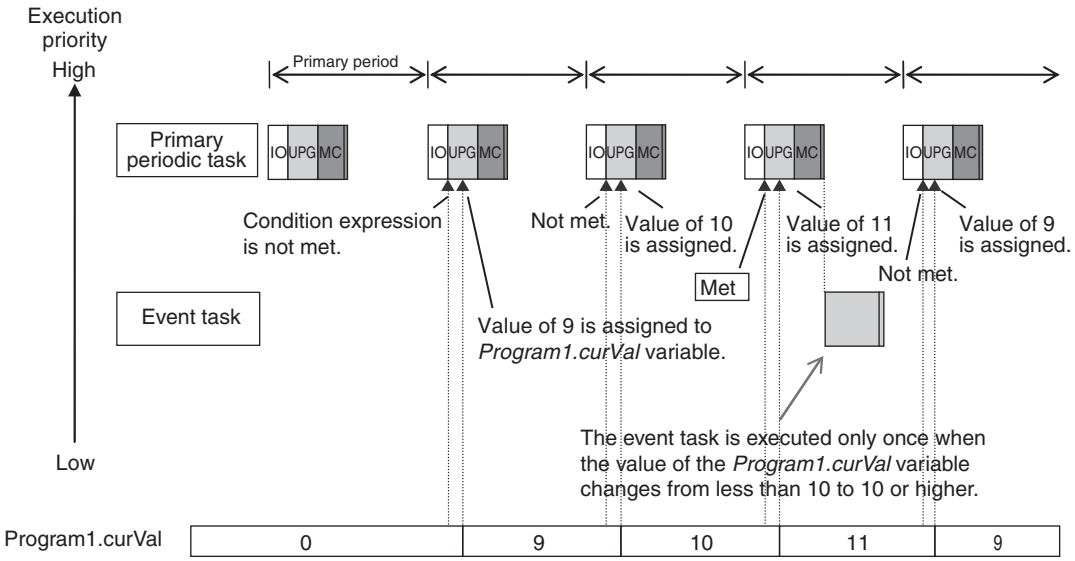
Example 1: Execution Condition for Event Task Set to a Change to TRUE of the *Program1.trigger1* Boolean Variable

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



Example 2: Execution Condition for an Event Task Set to When *Program1.curVal* (INIT variable) ≥ 10

The event task is executed only once when the value of *Program1.curVal* changes from less than 10 to 10 or higher.



Precautions for Correct Use

The CPU Unit evaluates whether the condition expression is met before the programs that are assigned to the primary periodic task are executed. Even if the specified value of the variable matches the condition expression during execution of the program in the primary periodic task, the condition is not evaluated until just before the next execution of the primary periodic task. This means that the event task will be executed after the end of the execution of the primary periodic task that follows the execution of the primary periodic task in which the value of the variable meets the condition expression.

5-6-5 Event Task Execution Timing for NJ-series Controllers

The execution priority of event tasks is 8 or 48. If the execution conditions for an event task are met while another task is in execution, the task with the higher execution priority is given priority. The task with the lower execution priority is interrupted. This is the same as with the primary periodic task and periodic tasks. The execution of an event task also depends on its execution conditions.
You can also set the same execution priority for more than one event task. You must be careful when the execution conditions are met for more than one event task that has the same execution priority.

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

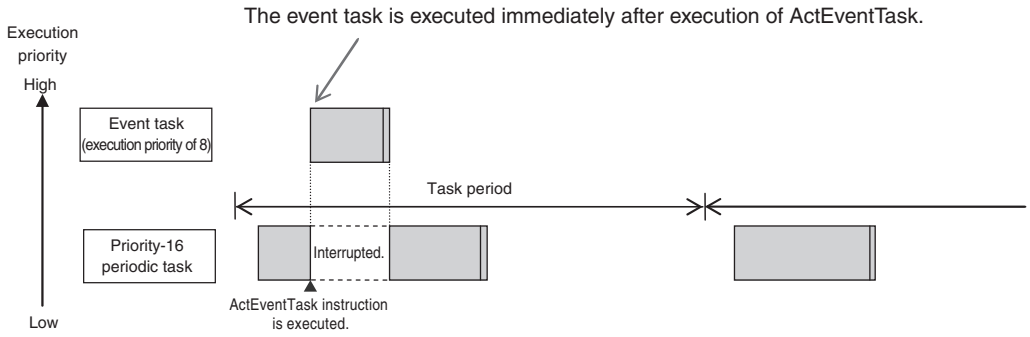
The execution of event tasks depends on whether the execution condition is triggered by an ActEventTask instruction or by when an condition expression for a variable is met.

● Execution Timing When the Execution Condition Is an ActEventTask Instruction

If the execution condition for an event task is triggered by an ActEventTask instruction, the event task will be executed immediately after the ActEventTask instruction is executed.

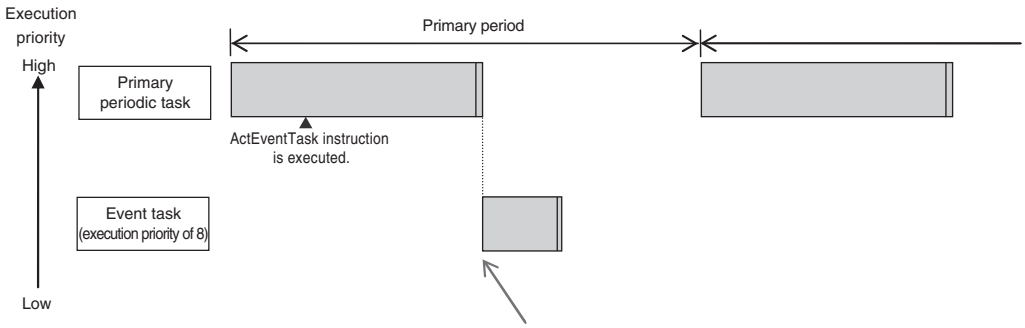
Example 1: ActEventTask Instruction Executed in Priority-16 Periodic Task and an Event Task with an Execution Priority of 8

The execution priority of the event task (execution priority of 8) is higher than the execution priority of the priority-16 periodic task. Execution of the priority-16 periodic task is therefore interrupted and the event task is executed.



Example 2: Executing an ActEventTask Instruction within the Primary Periodic Task

The event task has a lower execution priority than the primary periodic task, so the event task is executed after execution of the primary periodic task is completed.



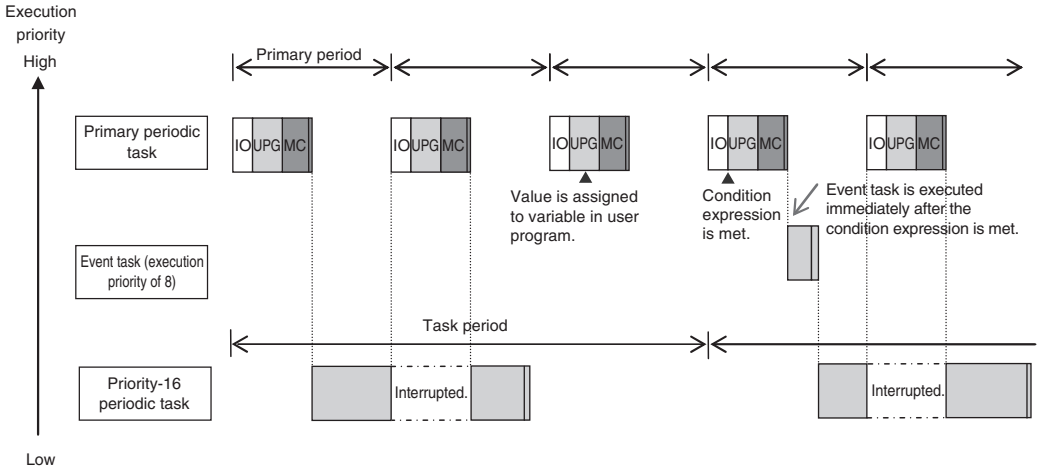
Event task is executed when the primary periodic task is completed.

● 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. This means that the event task will be executed immediately after the first execution of the primary periodic task after the specified value of the variable meets the condition expression. However, if there are tasks with a higher execution priority than the event task, those tasks will be executed first.

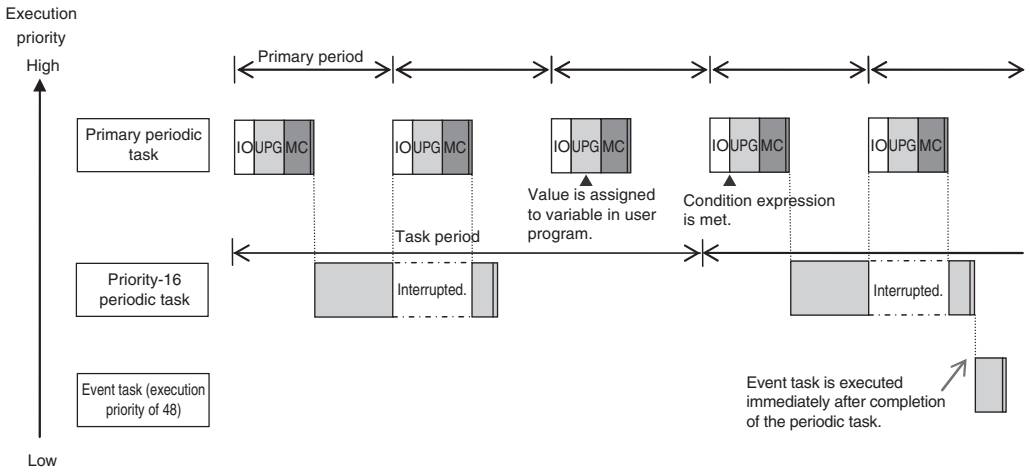
Example 1: Project with a Priority-16 Periodic Task and an Event Task Execution with a Priority of 8

The event task is executed immediately after the first execution of the primary periodic task after the condition expression is met. The execution priority of the event task (execution priority of 8) is higher than the execution priority of the priority-16 periodic task. The priority-16 periodic task is therefore executed after the event task is executed.



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

The execution priority of the event task is lower than the execution priority of the priority-16 periodic task. The event task is therefore executed after the priority-16 periodic task is executed.



Precautions for Correct Use

- The execution timing of an event task depends on how the condition expression is met. The match can be triggered by I/O refreshing in the primary periodic task, or by execution of a program that is assigned to the primary periodic task. This difference is described in the following table. This difference occurs because the condition expression is evaluated for a match by system common processing 1 inside the primary periodic task. Processing in the primary periodic task takes place in this order: I/O refreshing, system common processing 1, and user program execution.

Trigger for condition expression to match	Event task execution timing
I/O refreshing in the primary periodic task	After completion of the primary periodic task
Execution of the programs in the primary periodic task	After completion of the next execution of the primary periodic task

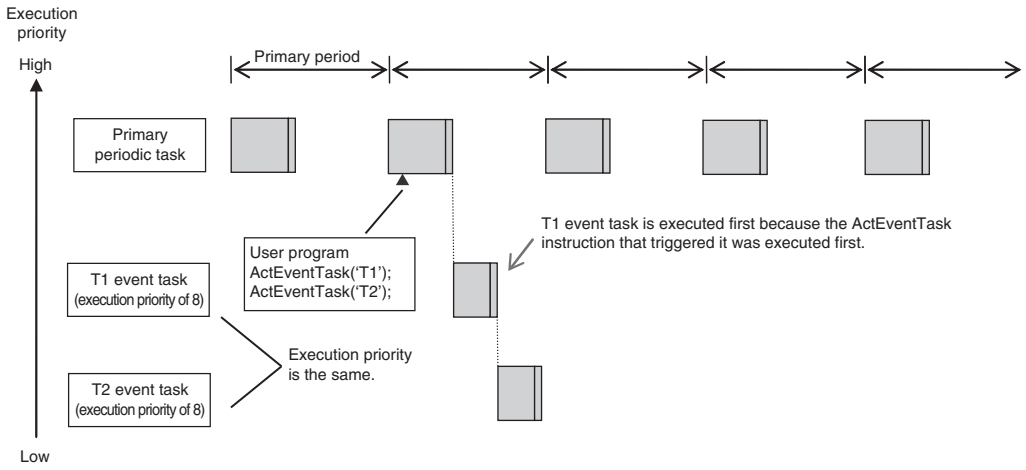
- In order for an event task to be executed, the condition expression must be met in the evaluation after the previous evaluation where the condition expression was not met. This means that even if the status of the condition expression changes from not met to met, if the condition returns to not met before the next evaluation, the event task will not be executed.
- For an NJ-series CPU Unit, specify an internal variable defined in the program that is assigned to the primary periodic task if you specify an internal variable with a data size of 64 bits or more to the condition expression. If an internal variable that is not defined in the primary periodic task is specified, the concurrency of variable values may not be ensured. As the result, the match evaluation may not be correctly performed.

Execution Timing for Event Tasks with the Same Execution Priority

You can also set the same execution priority for more than one event task. If the execution conditions are met for more than one event task that has the same execution priority, the event tasks will be executed in the order that their execution conditions are met.

Example 1: When Two ActEventTask Instructions Are Executed

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



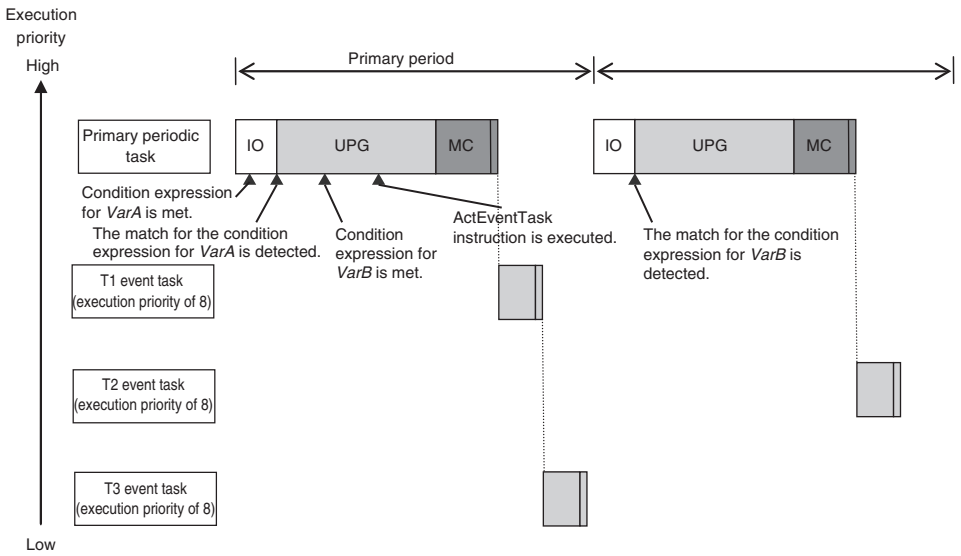
Example 2: When Both Condition Expressions for Variables and the ActEventTask Instruction Are Used

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

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

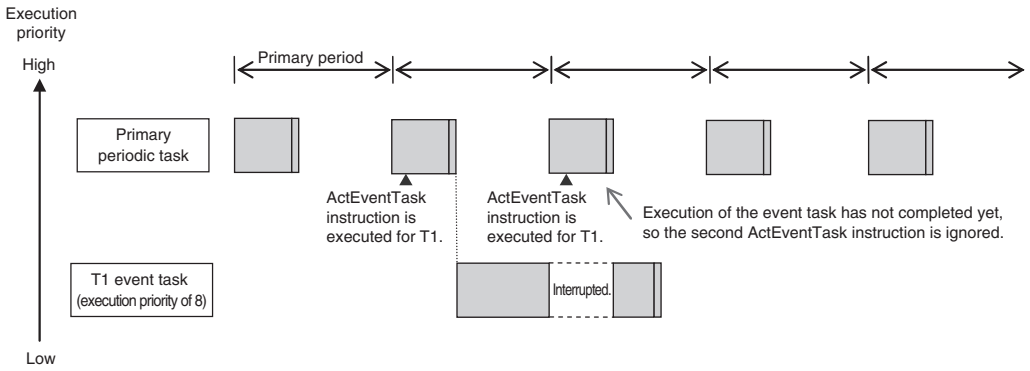
The operation would proceed as described below if the condition expression for *VarA* was met during I/O refreshing, the ActEventTask instruction was executed in the user program, and the condition expression for *VarB* was met during execution of the user program all in the same primary period.

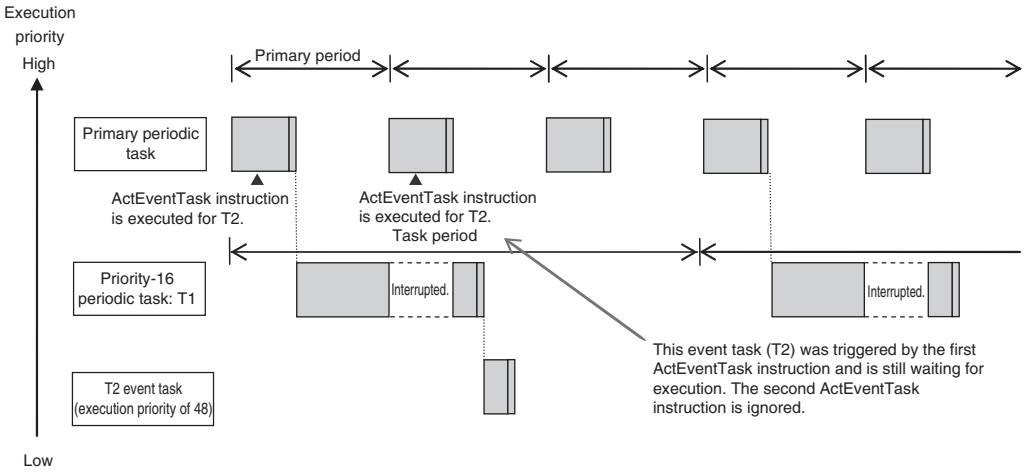
1. The condition expression for *VarA* is met during I/O refreshing.
2. In system common processing 1, the match is detected for the condition expression for *VarA* and T1 is executed. The condition expression for *VarB* is not yet met, so T2 is not executed.
3. The condition expression for *VarB* is met during execution of the user program.
4. The ActEventTask instruction is executed in the user program, so T3 is executed.
5. In system common processing 1 in the next primary period, the match is detected for the condition expression for *VarB* and T2 is executed.



5-6-6 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.





5-7 Services Other Than Tasks

The NJ/NX-series CPU Unit performs processing other than the primary periodic task, periodic tasks, and event tasks. This section describes the services other than tasks for the NX701 CPU Units, NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units.

NX701 CPU Units

For the NX701 CPU Unit, the processing includes the tag data link service and system services. The processing that is performed, the execution priority, and the execution timing for these two services are given in the following table.

Item	Tag data link service	System services
Execution priority	The tag data link service is executed in parallel with the task execution and system services.	System services are executed in parallel with the task execution and tag data link service.
Processing	Data is exchanged by using tags with other controllers and devices on an EtherNet/IP network. You can use the built-in EtherNet/IP port in the CPU Unit. *1	System services include services with low execution priority, such as the USB port service and SD Memory Card service.
Execution timing	The execution interval and the time that is required for each execution depend on the model of the CPU Unit and on the tag data link settings. *2	System services are executed at the required time if a request comes from the hardware or from outside of the CPU Unit.

*1. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for details on the processing that is performed for tag data links.

*2. Refer to *5-7-2 Processing Performed in and Execution Timing of the Tag Data Link Service* on page 5-92 for details on the execution interval and the time that is required for execution of tag data links.

NX502 CPU Units

For the NX502 CPU Unit, the processing includes the tag data link service and system services. The processing that is performed, the execution priority, and the execution timing for these two services are given in the following table.

Item	Tag data link service	System services
Execution priority	The tag data link service is executed in parallel with the task execution. It is prioritized than system services.	System services are executed in parallel with the task execution.
Processing	Data is exchanged by using tags with other controllers and devices on an EtherNet/IP network. You can use the built-in EtherNet/IP port in the CPU Unit. *1	System services include services with low execution priority, such as the SD Memory Card service.
Execution timing	The execution interval and the time that is required for each execution depend on the model of the CPU Unit and on the tag data link settings. *2	System services are executed at the required time if a request comes from the hardware or from outside of the CPU Unit.

*1. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for details on the processing that is performed for tag data links.

*2. Refer to *5-7-2 Processing Performed in and Execution Timing of the Tag Data Link Service* on page 5-92 for details on the execution interval and the time that is required for execution of tag data links.

NX102 CPU Units

For the NX102 CPU Units, the following three services are provided: the communications bridge service, the tag data link service and system services. The processing that is performed, the execution priority, and the execution timing for these three services are given in the following table.

Item	Communications bridge service*1	Tag data link service	System services
Execution priority	The communications bridge service is executed in parallel with the task execution and system services. It is prioritized than the tag data link service and system services.	Tag data link service is executed in parallel with the task execution. However, during execution of the communications bridge services, the tag data link service is not executed. It is prioritized than system services.	System services are executed in parallel with the task execution. However, during execution of the communications bridge services and tag data link service, system services are not executed.
Processing	The processing to relay CIP Safety communications between CIP Safety on EtherNet/IP devices and the Safety CPU Unit that is mounted to the CPU Unit, is executed in the communications bridge service.	Data is exchanged by using tags with other controllers and devices on an EtherNet/IP network. You can use the built-in EtherNet/IP port in the CPU Unit. *2	System services include services with low execution priority, such as the SD Memory Card service.
Execution timing	The communications bridge service is executed when the frames of the CIP Safety communications are sent and received by the built-in EtherNet/IP port.	The execution interval and the time that is required for each execution depend on the model of the CPU Unit and on the tag data link settings. *3	System services are executed depending on the execution priority if a request comes from the hardware or from outside of the CPU Unit.

*1. The service is executed for an NX102 CPU Unit with unit version 1.31 or later.
*2. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for details on the processing that is performed for tag data links.
*3. Refer to *5-7-2 Processing Performed in and Execution Timing of the Tag Data Link Service* on page 5-92 for details on the execution interval and the time that is required for execution of tag data links.

NX1P2 CPU Units

For the NX1P2 CPU Unit, the processing includes the tag data link service, option board service, and system services. The processing that is performed, the execution priority, and the execution timing for these three services are given in the following table.

Item	Tag data link service	Option board service	System services
Execution priority	The tag data link service is executed in parallel with the task execution. It is prioritized than the option board service and system services.	The option board service is executed in parallel with the task execution. However, during execution of the tag data link service, the option board service is not executed. It is prioritized than system services.	System services are executed in parallel with the task execution. However, during execution of the tag data link service or option board service, system services are not executed.
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 CPU Unit. *1	I/O refreshing for Analog I/O Option Boards is executed.	System services include services with low execution priority, such as the SD Memory Card service.

Item	Tag data link service	Option board service	System services
Execution timing	The execution interval and the time that is required for each execution depend on the model of the CPU Unit and on the tag data link settings. *2	The option board service is executed depending on the execution priority if a request comes from the Option Board.	System services are executed depending on the execution priority if a request comes from the hardware or from outside of the CPU Unit.

*1. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for details on the processing that is performed for tag data links.

*2. Refer to *5-7-2 Processing Performed in and Execution Timing of the Tag Data Link Service* on page 5-92 for details on the execution interval and the time that is required for execution of tag data links.

NJ-series CPU Units

For the NJ-series CPU Unit, the processing includes the tag data link service and system services. The processing that is performed, the execution priority, and the execution timing for these two services are given in the following table.

Item	Tag data link service	System services
Execution priority	The execution priority is between the execution priorities of the priority-16 periodic task and the priority-17 periodic task.	The execution priority is lower than that of any of the tasks.
Processing	Data is exchanged by using tags with other controllers and devices on an EtherNet/IP network. You can use the built-in EtherNet/IP port in the CPU Unit or a CJ-series CJ1W-EIP21 EtherNet/IP Unit to connect to other Controllers or devices. *1	System services include services with low execution priority, such as the USB port service and SD Memory Card service.
Execution timing	The execution interval and the time that is required for each execution depend on the model of the CPU Unit and on the tag data link settings. *2	The system services are executed in the unused time between execution of the tasks and tag data link service. *3

*1. For details on the processing that is performed for tag data links, refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* or the *CJ-series EtherNet/IP Unit Operation Manual for NJ-series CPU Unit (Cat. No. W495)*.
You can use CJ-series Units only on the NJ-series CPU Units.

*2. Refer to *5-7-2 Processing Performed in and Execution Timing of the Tag Data Link Service* on page 5-92 for details on the execution interval and the time that is required for execution of tag data links.

*3. If sufficient time cannot be obtained to execute the system services with an NJ-series CPU Unit, the processing of tasks with an execution priority of 17 or higher will be interrupted to allocate sufficient time. You can set the time for execution of the system services in the *System Service Monitoring Settings* on the Sysmac Studio.

5-7-1 Execution Priorities and Execution Orders of Services Other Than Tasks

This section provides examples of the execution priorities and execution orders of the communications bridge service, tag data link service, option board service, and system services.

Execution Priorities of Services Other Than Tasks

● NX701 CPU Units

The tag data link service is not affected by the task execution priority and system services.
System services are not affected by the task execution priority and tag data link service.

NX502 CPU Units

The tag data link service and system services are not affected by the task execution priority. The tag data link service and system services are executed in descending order of execution priority.

NX102 CPU Units

The communications bridge service, tag data link service and system services are not affected by the task execution priority. The communications bridge service, tag data link service and system services are executed in descending order of execution priority.

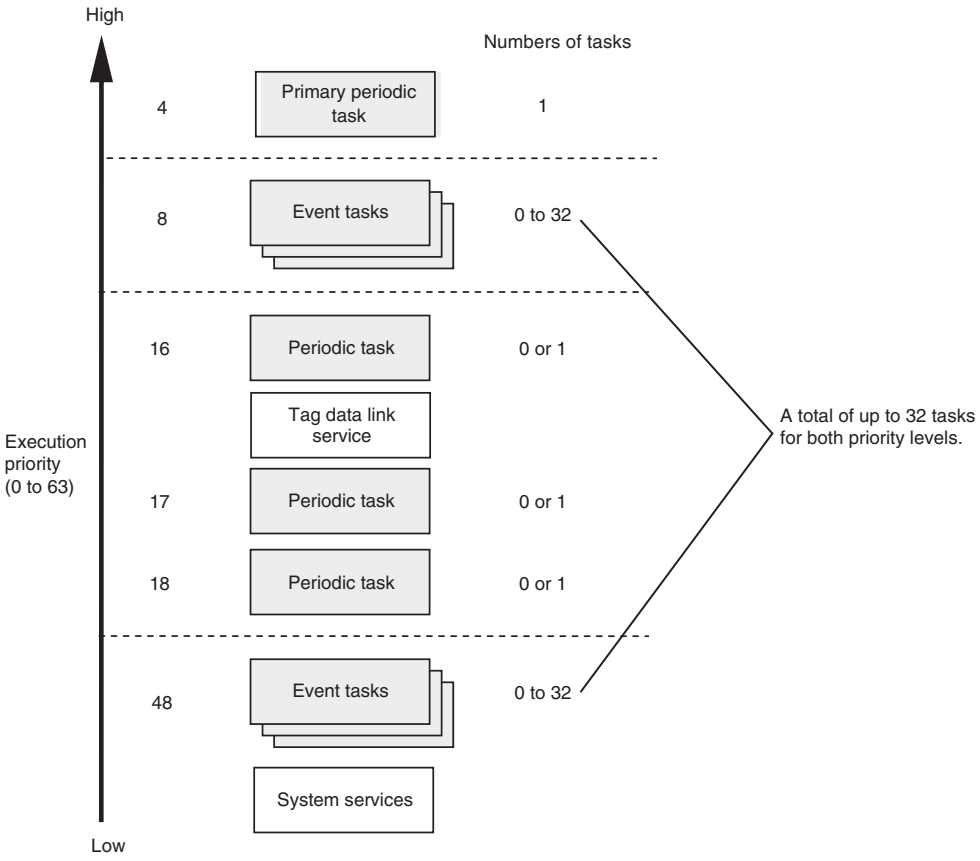
Note The communications bridge service is executed by an NX102 CPU Unit with unit version 1.31 or later.

NX1P2 CPU Units

The tag data link service, option board service and system services are not affected by the task execution priority. The tag data link service, option board service and system services are executed in descending order of execution priority.

NJ-series CPU Units

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.

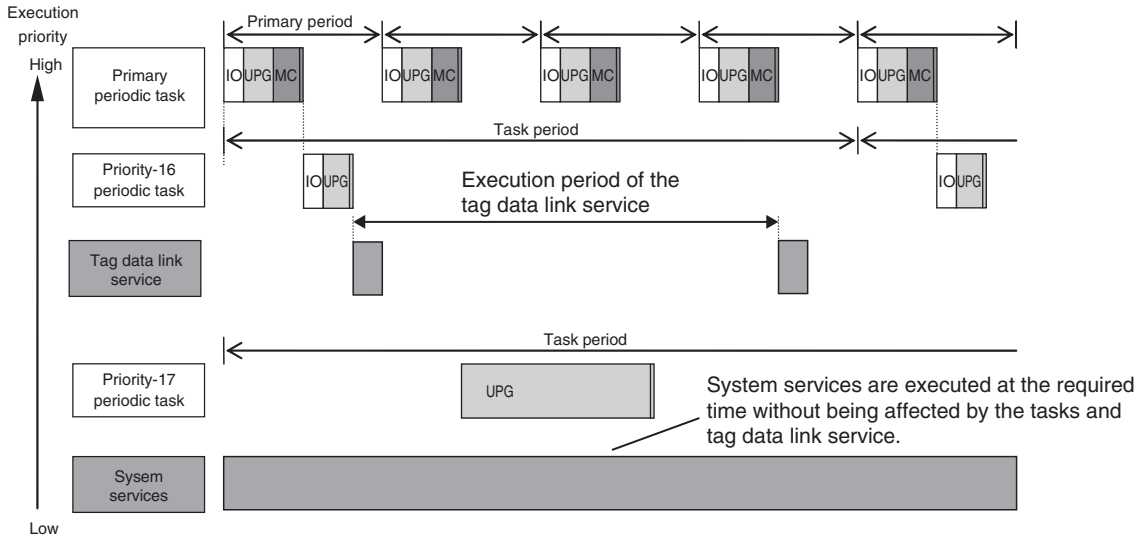


Examples of the Order of Execution for Services Other Than Tasks

As an example, the order of execution for the primary periodic task, priority-16 periodic task, priority-17 periodic task, tag data link service, option board service, and system services is shown below.

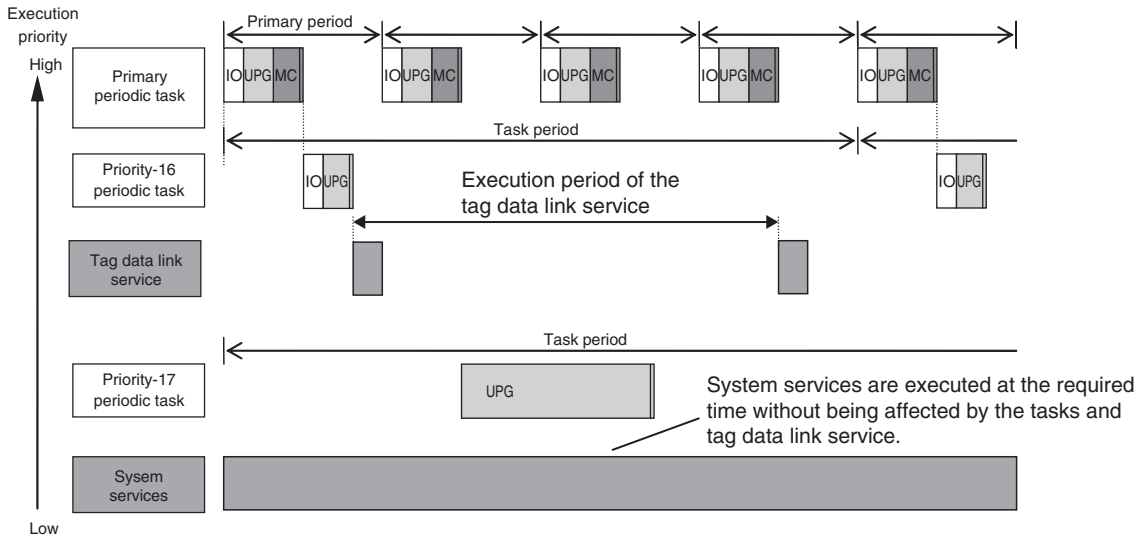
● NX701 CPU Units

- The tag data link service is executed without being affected by the tasks and system services.
- The system services are executed at the required time without being affected by the tasks and tag data link service.



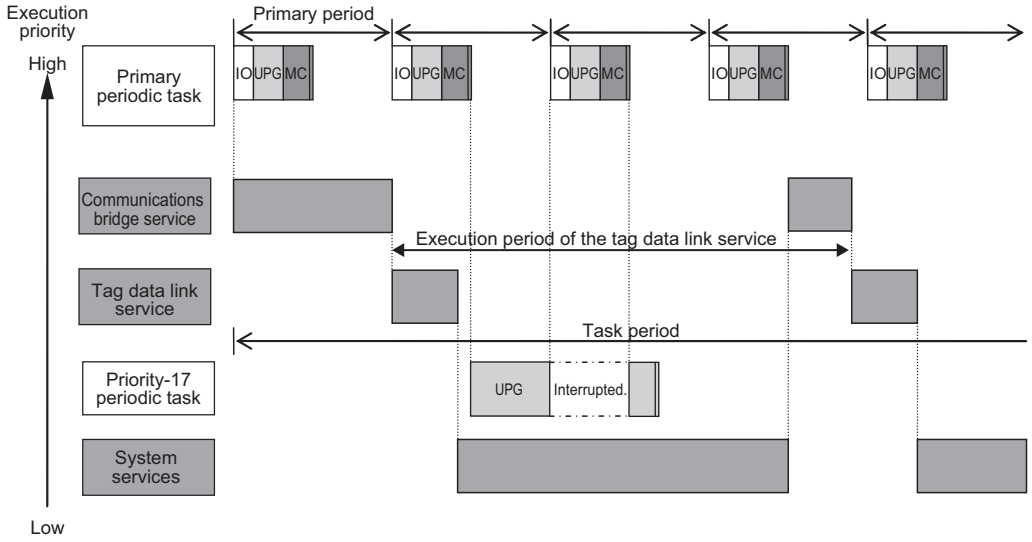
● NX502 CPU Units

- You can execute the tag data link service and system services in parallel with the execution of tasks.
- The order of execution priority is tag data link service and then system services.



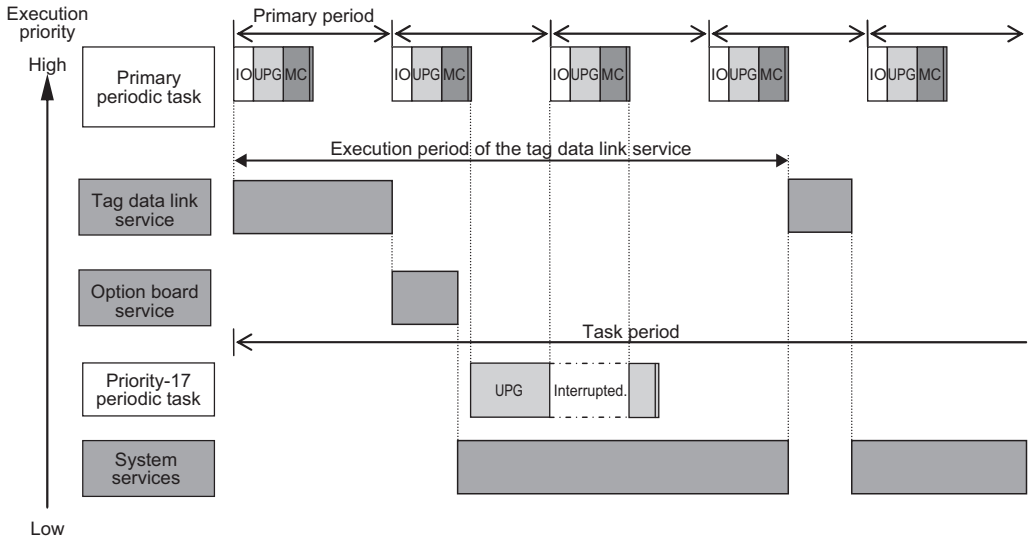
● NX102 CPU Units

- You can execute the communications bridge service, tag data link service and system services in parallel with the execution of tasks.
- The order of execution priority is communications bridge service, tag data link service and then system services.



● NX1P2 CPU Units

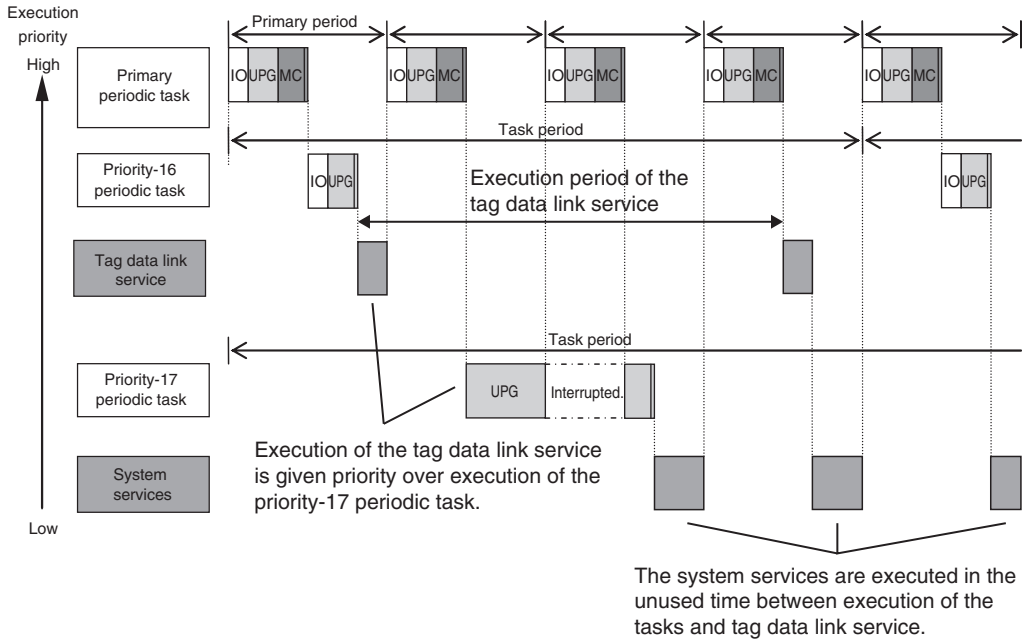
- You can execute the tag data link service, option board service or system services in parallel with the execution of tasks.
- The order of execution priority is tag data link service, option board service and then system services.



● NJ-series CPU Units

- Execution of the tag data link service is given priority over execution of the priority-17 periodic task. However, execution of the primary periodic task and priority-16 periodic task is given even higher priority.

- The system services are executed in the unused time between execution of all of the tasks and tag data link service.



The execution interval and the execution time for one execution depend on the model of the CPU Unit and on the tag data link settings. Refer to *5-7-2 Processing Performed in and Execution Timing of the Tag Data Link Service* on page 5-92 for details.

5-7-2 Processing Performed in and Execution Timing of the Tag Data Link Service

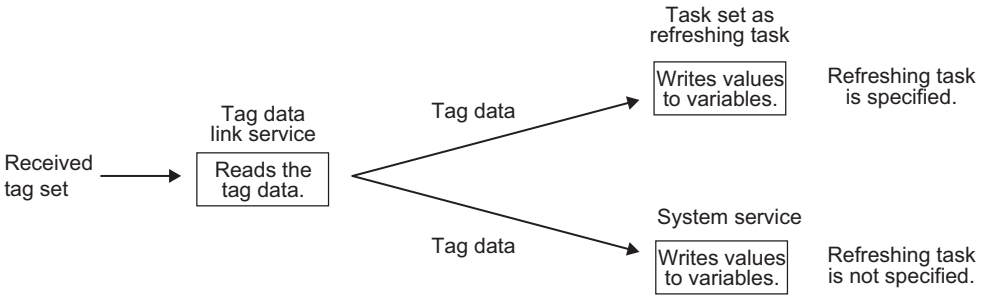
This section describes the processing that is performed in and the execution timing of the tag data link service.

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 ports on the CPU Unit is used.

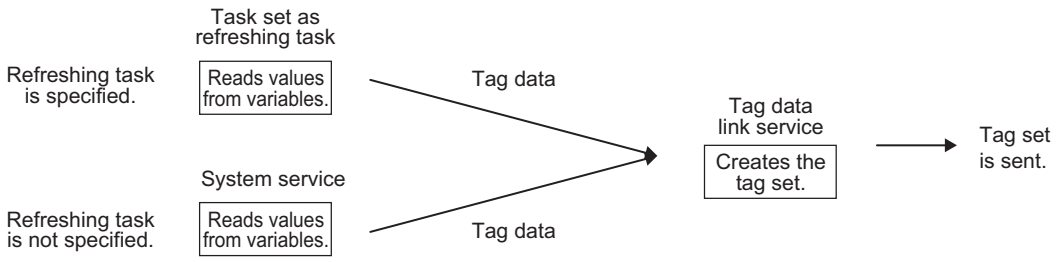
● Flow of Tag Data Reception Processing

- The tag data link service reads the tag data from the received tag sets.
- If a refreshing task is set, the task writes the values of the tag data to the variables that are assigned to the tags.
- If a refreshing task is not set, a system service writes the values of the tag data to the variables that are assigned to the tags.



● Flow of Tag Data Transmission Processing

- If a refreshing task is set, the task reads the values of the tag data from the variables that are assigned to the tags.
- If a refreshing task is not set, a system service reads the values of the tag data from the variables that are assigned to the tags.
- Then, in the tag data link service, the tag set is created from the tag data and the tag set is sent.



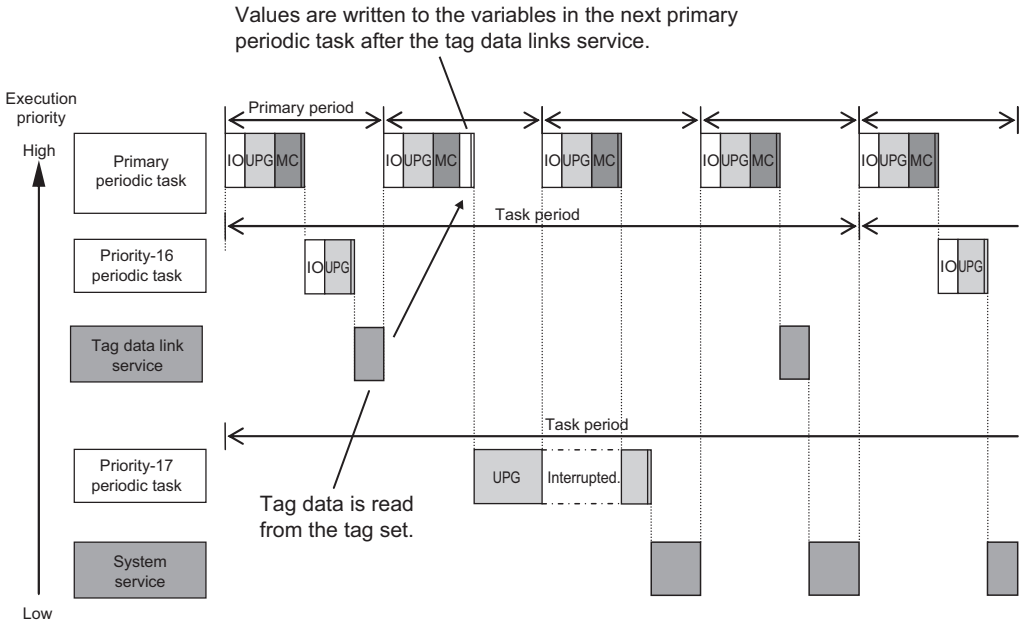
Additional Information

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

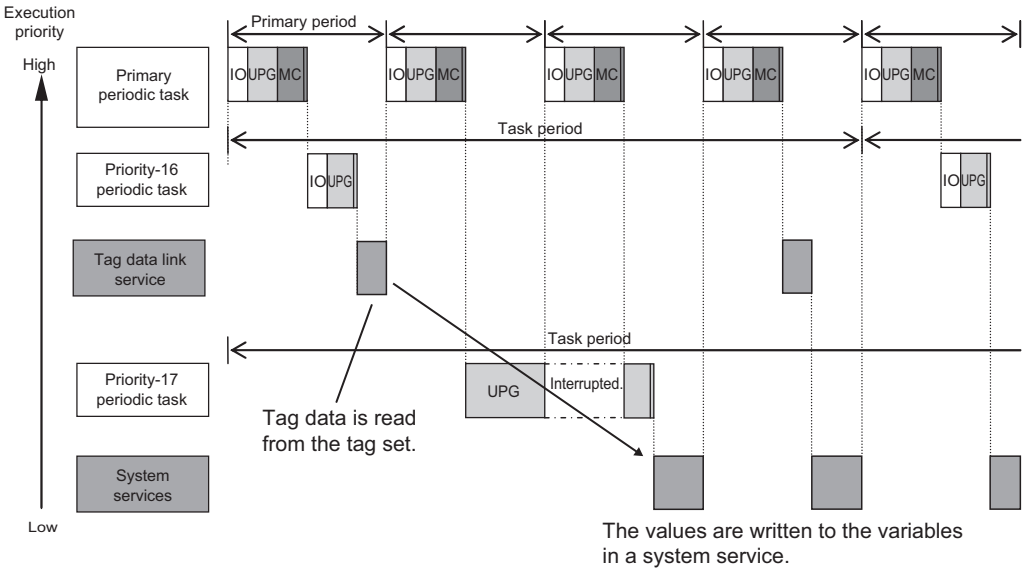
The process to write values to and read values from variables is different depending on whether a refreshing task is set. If a refreshing task is set, the values are read and written in that task. If a refreshing task is not set, the values are read and written in a system service. This means there is a difference in the execution timing of processing tag data links.

The difference in the execution timing when data is received for tag data links with an NJ-series CPU Unit is shown below.

- When a Refreshing Task Is Specified
  In this example, the primary periodic task is set as the refreshing task. Values are written to the variables in the next primary periodic task after the tag data link service.

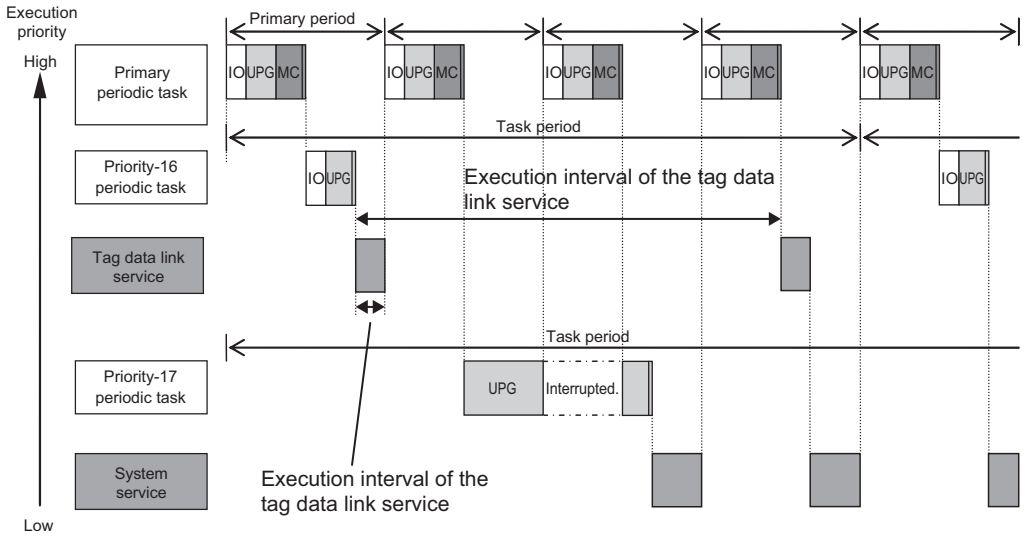


- When a Refreshing Task Is Not Specified
  The values are written to the variables in a system service.



Execution Timing of the Tag Data Link Service

The execution interval and the execution time depend on the model of the CPU Unit and on the tag data link settings. Guidelines are provided below. For an NJ-series CPU Unit, however, if a task with a higher execution priority than the tag data link service is executed, the execution interval and the execution time will be longer than the following values.



● Using the Built-in EtherNet/IP Port on the CPU Unit

CPU Unit	Tag data link settings			Execution interval (ms)	Execution time (ms)
	Number of tags	Number of connections	RPI [ms]		
NJ501-□□□□	8	1	1	1	0.02 to 0.04
		4	2	1	0.02 to 0.10
		32	50	1	0.02 to 0.10
NJ301-□□□□	8	1	1	1	0.03 to 0.06
		4	2	1	0.03 to 0.15
		32	50	1	0.03 to 0.15
NJ101-□□□□	8	1	1	1	0.04 to 0.09
		4	2	1	0.04 to 0.28
		32	50	1	0.04 to 0.28

Additional Information

For an NX-series CPU Unit, the priority-17 and priority-18 periodic tasks and the system service are not affected by execution of the tag data link service.

● Using the CJ1W-EIP21 EtherNet/IP Unit

CPU Unit	Tag data link settings				Execution interval (ms)	Execution time (ms)
	Number of tags	Number of connections	RPI [ms]	Send/ receive		
NJ501-□□□□	8	1	1	Send	3	0.10
				Receive	3	0.10
		4	2	Send	4	0.22
				Receive	4	0.13
		32	50	Send	12	0.68
				Receive	6 to 35	0.23 to 0.66
NJ301-□□□□	8	1	1	Send	3	0.16
				Receive	3	0.16
		4	2	Send	3	0.21
				Receive	4	0.33
		32	50	Send	15	1.05
				Receive	8 to 35	0.29 to 1.85
NJ101-□□□□	8	1	1	Send	3	0.21
				Receive	3	0.21
		4	2	Send	6	0.46
				Receive	13	0.80
		32	50	Send	12 to 50	2.97 to 4.09
				Receive	25	2.36

Version Information

Execution of processing for tag data links depends on the unit version of the CPU Unit and the Sysmac Studio version as given below.

Unit version of CPU Unit	Sysmac Studio version	
	1.03 or lower	1.04 or higher
1.03 or later	Built-in EtherNet/IP port service in system services	Tag data link service*1
1.02 or earlier	Built-in EtherNet/IP port service in system services	

*1. If a unit version of 1.02 or earlier is set on the Sysmac Studio, the tag data links are processed in the built-in EtherNet/IP port service in the system services.

5-7-3 Processing Performed in and Execution Timing of the Option Board Service

Processing Performed in Option Board Service

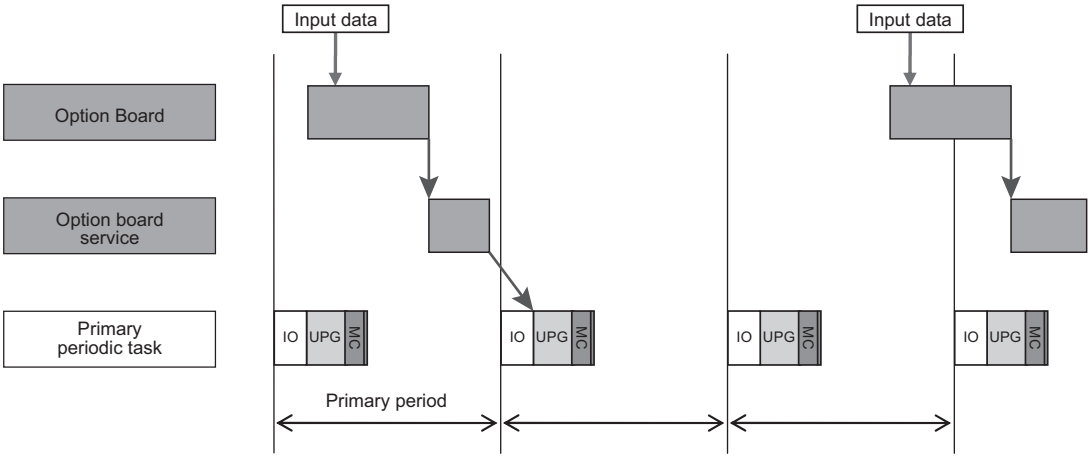
For the NX1P2 CPU Units, I/O refreshing for Analog I/O Option Boards is executed in the processing performed in the option board service.

Execution Timing of the Option Board Service

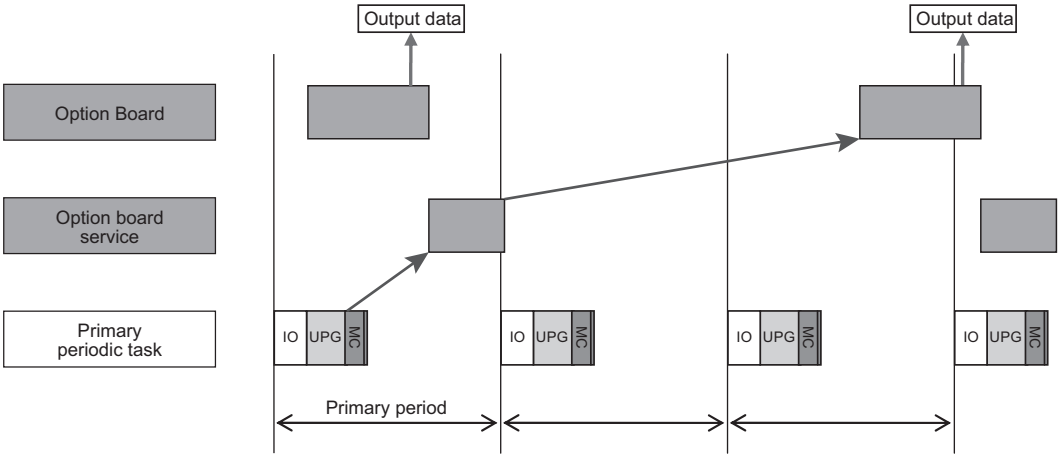
For the NX1P2 CPU Units, the option board service is executed without being affected by the tasks. However, the tag data link service, option board service and system services are not executed in parallel.

The option board service is executed depend on the execution priority if a request comes from the Option Board. The internal processing of an Option Board, the option board service for an NX1P2 CPU Unit, and the task execution are performed asynchronously.

When you input the data, the input data is imported in the internal processing of an Option Board and processed in the option board service for an NX1P2 CPU Unit. When the processing in the option board service is completed, you can use the input data in user program execution in the next primary periodic task.



When you output data, the output data is processed in the option board service after user program execution is completed. When the processing in the option board service is completed, the output data is output in the next internal processing of an Option Board.



5-7-4 Processing Performed in and Execution Timing of the Communications Bridge Service

Processing Performed in Communications Bridge Service

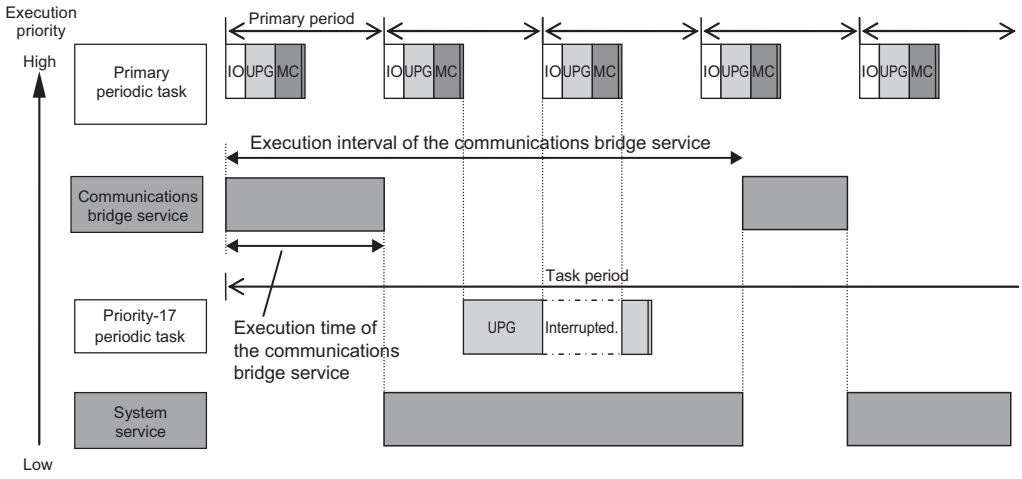
For the NX102 CPU Units, the processing to relay CIP Safety communications between CIP Safety on EtherNet/IP devices and the Safety CPU Unit that is mounted to the CPU Unit, is executed in the communications bridge service.

Note The communications bridge service is executed by an NX102 CPU Unit with unit version 1.31 or later.

Execution Timing of the Communications Bridge Service

For the NX102 CPU Units, the communications bridge service is executed without being affected by the tasks. However, the communications bridge service, tag data link service and system services are not executed in parallel.

The communications bridge service is executed when the frames of the CIP Safety communications are sent and received by the built-in EtherNet/IP port.



The execution interval and execution time for the communications bridge service depend on the settings of CIP Safety communications.

Refer to the *NX-series Safety Control Unit User's Manual (Cat. No. Z930-E1-12 or later)* for information on the CIP Safety communications.

5-7-5 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
USB port service*1	• Processing of service requests from the Sysmac Studio or host computers
Built-in EtherNet/IP port service	• Processing of message service requests, such as CIP commands, from the Sysmac Studio, an HMI, host computers, or other Controllers • Execution of communications instructions for CIP and socket communications
Built-in EtherCAT port service	• Execution of EtherCAT message communications
Communications processing for a Serial Communications Option Board*2	• Execution of communications processing for a Serial Communications Option Board
Service for CJ-series Special Units*3	• Event servicing for CJ-series Special Units • Execution of communications instructions (CIP)
SD Memory Card service	• Access from FTP client • SD Memory Card operations from the Sysmac Studio • Execution of SD Memory Card instructions
Self-diagnosis	• Hardware error detection

*1. The NX701 CPU Units with hardware revision A or later, NX502, NX102, and NX1P2 CPU Units do not provide a USB port.
*2. The communications processing for a Serial Communications Option Board is executed only by an NX1P2 CPU Unit.
*3. The CPU Unit exchanges data between CJ-series Special Units and the memory words that are allocated to them during *I/O refreshing*.
You can use CJ-series Special Units only with NJ-series CPU Units.

Execution Timing of the System Services

For the NX701 CPU Units, the system services are executed at the required time without being affected by the task and tag data link service. It is designed to always secure sufficient time for system service execution.

For the NX502 CPU Units, the system services are executed without being affected by the tasks. However, during execution of the tag data link service, system services are not executed.

For the NX102 CPU Units, the system services are executed without being affected by the tasks. However, during execution of the communications bridge service or tag data link service, system services are not executed.

For the NX1P2 CPU Units, the system services are executed without being affected by the tasks. However, during execution of the tag data link service or option board service, system services are not executed.

For NJ-series CPU Units, the system services are executed in the unused time between execution of the tasks and tag data link service.
If sufficient time cannot be obtained to execute the system services, the processing of tasks with an execution priority of 17 or higher will be interrupted to allocate sufficient time.
You can set the time for execution of the system services in the *System Service Monitoring Settings* on the Sysmac Studio.

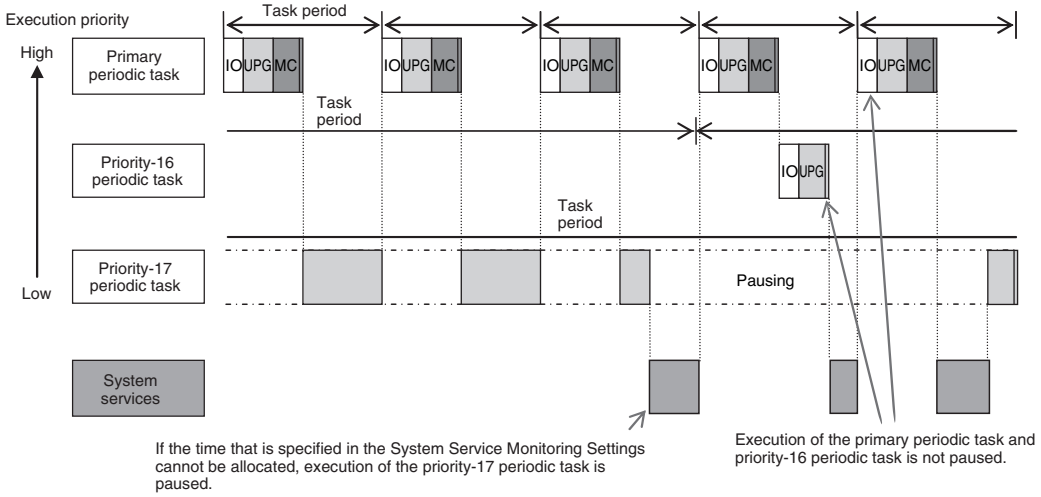
There is no priority in the processing of system services. All of the processing is executed in parallel with time slicing.

● Execution Timing When Sufficient Execution Time for the System Services Cannot Be Obtained

With an NJ-series CPU Unit, if there is not enough unused time between execution of the tasks and tag data link service to execute the system services, the execution of tasks with an execution priority of 17 or higher will be interrupted to allocate sufficient time.

If the system service execution time cannot be obtained even if execution of tasks with an execution priority of 17 or higher is temporarily interrupted, an Insufficient System Service Time Error occurs and user program execution stops.

You can set the time for execution of the system services in the System Service Monitoring Settings on the Sysmac Studio.



● System Service Monitoring Settings

The System Service Monitoring Settings are used to monitor whether sufficient system service execution time can be secured with NJ-series CPU Units. If the execution of all tasks cannot be completed within the system service execution interval, execution is monitored to see if at least the ratio of execution that is set for the System Service Execution Time Ratio is completed.

You can set the System Service Monitoring Settings in the Basic Settings Display of the Operation Settings Tab Page on the Sysmac Studio.

Access point	Setting group	Setting [unit]	Description	Set value	Default	Update timing	Changes in RUN mode
Operation Settings, Operation Settings Tab, Basic Settings	System Service Monitoring Settings	System Service Execution Interval [ms]	Sets the interval of system service execution.	10 ms to 1 s	10 ms	When downloaded to CPU Unit	Not allowed.
		System Service Execution Time Ratio [%]	Sets the ratio for monitoring system service execution.	5% to 50%	10%	When downloaded to CPU Unit	Not allowed.

Additional Information

NX-series CPU Units are designed to always secure sufficient time for system service execution, so the System Service Monitoring Settings are not provided. Also an Insufficient System Service Time Error will not occur.

Precautions for Correct Use

- For NJ-series CPU Units, set the System Service Monitoring Settings to the minimum values that are required to meet the response performance of the system services so that sufficient time can be allocated to the system services and task execution. The System Service Monitoring Settings are used to monitor whether the specified system service execution time can be obtained. System services will not necessarily be executed for the specified time.
- For NJ-series CPU Units, design the tasks so that sufficient time can be allocated to execution of the system services. Refer to *5-12 Task Design Methods and I/O Response Times* on page 5-133 for the setting procedures for tasks.
- To increase the system service execution time with an NJ-series CPU Unit, increase the task period or take other steps to increase the unused time between task execution.
- With an NJ-series CPU Unit, if the time that is specified in the System Service Monitoring Settings cannot be allocated to the system service execution time even if execution of tasks with an execution priority of 17 or higher is interrupted, an Insufficient System Service Time Error occurs and user program execution stops.
- With an NJ-series CPU Unit, if sufficient system service execution time cannot be allocated and execution of tasks with an execution priority of 17 or higher is interrupted, a Task Period Exceeded error will occur for the tasks that are interrupted. Design the tasks so that the execution of tasks with an execution priority of 17 or higher is completed within the task periods even if the execution time of the system services satisfies the System Service Monitoring Settings.

5-8 Assignment and Settings Related to Tasks

This section describes the assignment and setting related to tasks.

5-8-1 Assigning I/O Refreshing to Tasks

I/O refreshing of the EtherCAT slaves, NX Units on the CPU Unit, X Bus Units, I/O built in the CPU Unit, Option Boards in the CPU Unit, and CJ-series Units is assigned to the tasks. Unit of assignment and tasks to which assignment is possible are different depending on the target for I/O refreshing. Unit of assignment refers to a target or a group of targets for I/O refreshing assigned to one task. For example, when the unit of assignment is Slave Terminal, you can assign I/O refreshing to only one task even if more than one NX Unit is connected to a Communications Coupler Unit.
If you want to perform input and output operations in tasks to which I/O refreshing is not assigned, refer to *Input and Output Operations in Tasks to Which I/O Refreshing Is Not Assigned* on page 5-104.
The following table shows the relationship among the target for I/O refreshing, the assignable task, and the unit of assignment.

I/O refreshing target	Assignable task	Unit of assignment
Communications Coupler Unit with an NX-series Safety Control Unit on the Slave Terminal	Primary periodic task or priority-5 and priority-16 periodic tasks *1*2*3	Slave Terminal
Communications Coupler Unit with an NX Unit assigned to an axis on the Slave Terminal	Primary periodic task or priority-5 periodic task*3	
Communications Coupler Unit without an NX Unit assigned to an axis on the Slave Terminal	Primary periodic task or priority-5 and priority-16 periodic tasks*2*3	
EtherCAT slaves that are assigned to **axes**	Primary periodic task or priority-5 periodic task*2*3	Slave
Other EtherCAT slaves	Primary periodic task or priority-5 and priority-16 periodic tasks*2*3	
NX Units on the CPU Unit*4 and I/O built in the CPU Unit*6	Primary periodic task	---
X Bus Units*5	Primary periodic task or priority-16 periodic task	---
CJ-series Basic I/O Units*7	Primary periodic task or priority-16 periodic task Primary periodic task	Unit
CJ-series Special I/O Units*7		
CJ-series CPU Bus Units*7		

*1. If there are multiple Slave Terminals on which an NX-series Safety Control Unit is mounted, you cannot assign them to a combination of the periodic tasks other than primary-5 periodic task and primary-5 periodic task.
*2. You can use the priority-5 periodic task only with the NX701 CPU Unit.
*3. You can only use the primary periodic task with the NX502 CPU Units, NX102 CPU Units, or NX1P2 CPU Units.
*4. You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*5. You can use X Bus Units only on the NX502 CPU Units.
*6. You can use I/Os built in the CPU Unit only with the NX1P2 CPU Units.

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

Precautions for Safe Use

If two different function modules are used together, such as when you use CJ-series Basic Units and EtherCAT slaves, take suitable measures in the user program and external controls to ensure that safety is maintained in the controlled system if one of the function modules stops. The relevant outputs will behave according to the slave or Unit specifications if a partial fault level error occurs in one of the function modules.
Refer to *Non-fatal Errors* in the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for information on the partial fault level Controller error.

Additional Information

When using both EtherCAT slaves and CJ-series Units, you can reduce the task execution time of the primary periodic task by assigning the I/O refreshing to different tasks. Assign I/O refreshing for EtherCAT slaves to the primary periodic task and assign I/O refreshing for CJ-series Units to the priority-16 periodic task.

Sysmac Studio Setting Procedure

For the slaves and Units that are not assigned to axes, set the tasks in which to perform I/O refreshing in **I/O Control Task Settings** under **Configuration and Setup - Task Settings** on the Sysmac Studio.
Refer to *I/O Control Task Settings* on page 4-10 for details.
For the slaves and Units that are assigned to axes, specify the motion controls to use in **Motion Control Setup** on the Sysmac Studio. The tasks to perform I/O refreshing are set.
Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for details.

Timing of I/O Refreshing

The table below shows when I/O is refreshed for EtherCAT slaves, NX Units on the CPU Unit, X Bus Units, I/O built in the CPU Unit, Option Boards in the CPU Unit, and CJ-series Units.

I/O refreshing target	I/O refreshing task	Period of I/O refreshing
EtherCAT slaves	Primary periodic task	Task period of primary periodic task
	Priority-5 periodic task*1	Task period of priority-5 periodic task
	Priority-16 periodic task	Task period of primary periodic task*2
NX Units on CPU Unit*3	Primary periodic task	Task period of primary periodic task
X Bus Units*4	Primary periodic task	Task period of primary periodic task
	Priority-16 periodic task	Task period of priority-16 periodic task
I/O built in the CPU Unit*5	Primary periodic task	Task period of primary periodic task
CJ-series Units*6	Primary periodic task	Task period of primary periodic task
	Priority-16 periodic task	Task period of priority-16 periodic task

*1. You can use the priority-5 periodic task only with the NX701 CPU Unit.
*2. EtherCAT communications is executed during I/O refreshing in the primary periodic task. If the priority-16 periodic task is used to control EtherCAT slaves, the data refresh period during I/O refreshing will be the task period of the priority-16 periodic task.
*3. You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

*4. You can use X Bus Units only on the NX502 CPU Units.
*5. You can use built-in I/O only with the NX1P2 CPU Units.
*6. You can use CJ-series Units only on the NJ-series CPU Units.

● Priorities in Process Data Communications

EtherCAT process data communications is executed during I/O refreshing. With an NX701 CPU Unit, you can perform process data communications in the primary periodic task and the priority-5 periodic task.

When process data communications is executed in the primary periodic task or the priority-5 periodic task, process data communications in the primary periodic task is prioritized. Therefore, the I/O refresh time of the priority-5 periodic task is longer even if the process data communications cycle and process data size of two tasks are the same.

Accessing I/O from the User Program

You use device variables to access I/O ports from the user program.
Access the device variables from a program in the task that is set in the I/O Control Task Settings.

Input and Output Operations in Tasks to Which I/O Refreshing Is Not Assigned

If you attempt to output data directly from a task to which I/O refreshing is not assigned, an error will occur when you check the program on the Sysmac Studio. For example, if the processes for NX Units are assigned to different tasks on the Slave Terminal, data can be output only from the task to which I/O refreshing is assigned. In this case, you need to create the program to pass the data from the task to which I/O refreshing is not assigned to the task to which I/O refreshing is assigned and output data from the task to which I/O refreshing is assigned.
The following sample programming shows how to pass the data from the task to which I/O refreshing is not assigned to the task to which I/O refreshing is assigned and output data from the task to which I/O refreshing is assigned.
In this sample programming, the external data input processing is performed in a task to which I/O refreshing is not assigned. Usually, this kind of processing causes a warning to occur when you check the program on the Sysmac Studio. However, the execution of processing is possible.

● Unit Configuration

A Slave Terminal is used. The following table shows the Unit configuration of the Slave Terminal.

Model number	Product name
NX-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.

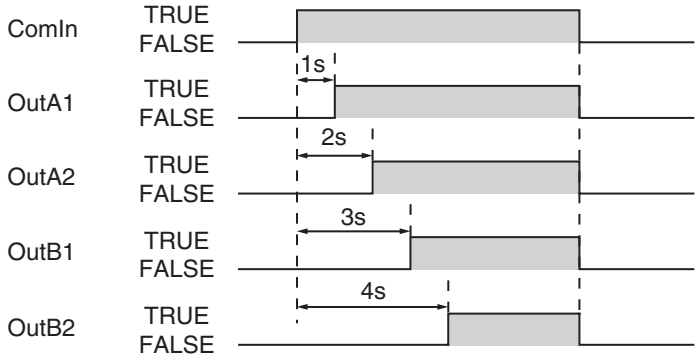
Position	Port	Description	R/W	Data type	Variable	Variable comment	Variable type
Unit1	NX-ID3317						
	Input Bit 00	Input bit 00	R	BOOL	ComIn	Common input value	Global variable
Unit2	NX-OD3256						
	Output Bit 00	Output bit 00	W	BOOL	OutA1	Output value A1	Global variable
	Output Bit 01	Output bit 01	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.
- When the value of *ComIn* changes from TRUE to FALSE, the values of *OutA1*, *OutA2*, *OutB1* and *OutB2* change from TRUE to FALSE.

The following figure shows the timing chart.



● Task Processing

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

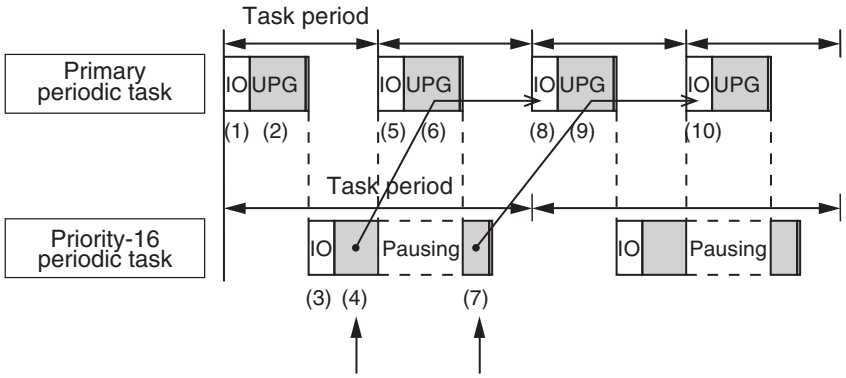
OutA1 and *OutA2* are controlled in the primary periodic task. A series of processing, i.e., input of *ComIn*, calculations and outputs of *OutA1* and *OutA2* , is performed in the primary periodic task.

For controls of *OutB1* and *OutB2*, the processing from input of *ComIn* to calculations is performed in the periodic task. The calculation results of the periodic task are assigned to the temporary global variables *tmp_OutB1* and *tmp_OutB2*. In the primary periodic task, the values of *tmp_OutB1* and *tmp_OutB2* are assigned to *OutB1* and *OutB2*. Then, *OutB1* and *OutB2* are output.

The Controller may pause the periodic task in order to execute the primary periodic task. The assignment of calculation results to *tmp_OutB1* and *tmp_OutB2* may be performed before or after the

pause of the periodic task. And the timing of assignment determines the timing of output of calculation results in the primary periodic task.

The following figure shows an example of processing flow. The task period of the periodic task is set to twice as long as that of the primary periodic task.



If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), they are output in step (8).

If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (7), they are output in step (10).

1. *ComIn* is input during I/O refreshing in the primary periodic task.
2. The calculation is performed during the user program execution in the primary periodic task and the values are assigned to *OutA1* and *OutA2*.
   Also, the values of *tmp_OutB1* and *tmp_OutB2* are assigned to *OutB1* and *OutB2*. However, the values of *tmp_OutB1* and *tmp_OutB2* are the initial values because the values are not assigned to *tmp_OutB1* and *tmp_OutB2* in the periodic task. Therefore, the values of *OutB1* and *OutB2* are also the initial values.
3. *ComIn* is input during I/O refreshing in the periodic task.
4. The calculation is performed during the user program execution in the periodic task.
   If the primary periodic task is executed while the periodic task execution is in progress, the periodic task is paused. Depending on the timing of processing in the periodic task, the assignment of the values to *tmp_OutB1* and *tmp_OutB2* may be performed before or after the pause of the periodic task.
5. *OutA1*, *OutA2*, *OutB1* and *OutB2* are output during I/O refreshing in the primary periodic task. Also, *ComIn* is input again.
6. The calculation is performed during the user program execution in the primary periodic task and the values are assigned to *OutA1* and *OutA2*.
   Also, the values of *tmp_OutB1* and *tmp_OutB2* are assigned to *OutB1* and *OutB2*. If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), the calculation results of the periodic task are reflected in *OutB1* and *OutB2*. If the values are not assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), the values of *tmp_OutB1* and *tmp_OutB2* are the initial values. Therefore, the values of *OutB1* and *OutB2* are also the initial values.
7. The calculation is performed during the user program execution in the periodic task that follows step (4).
   If the values are not assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), they are assigned here.
8. *OutA1*, *OutA2*, *OutB1* and *OutB2* are output during I/O refreshing in the primary periodic task.

If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (4), the calculation results of the periodic task are output as *OutB1* and *OutB2*.
If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (7), the initial values of *OutB1* and *OutB2* are output.

9. The calculation is performed during the user program execution in the primary periodic task and the values are assigned to *OutA1* and *OutA2*.
   Also, the values of *tmp_OutB1* and *tmp_OutB2* are assigned to *OutB1* and *OutB2*. If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (7), the calculation results of the periodic task are reflected in *OutB1* and *OutB2*.
10. *OutA1*, *OutA2*, *OutB1* and *OutB2* are output during I/O refreshing in the primary periodic task.
    If the values are assigned to *tmp_OutB1* and *tmp_OutB2* in step (7), the calculation results of the periodic task are output as *OutB1* and *OutB2*.

You can use the Lock and Unlock instructions to perform the task exclusive controls to prevent the values of *tmp_OutB1* and *tmp_OutB2* from being overwritten by the periodic task before they are accessed by the primary periodic task. Refer to *5-9-1 Ensuring Concurrency of Variable Values between Tasks* on page 5-114 for details on task exclusive control.

● 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

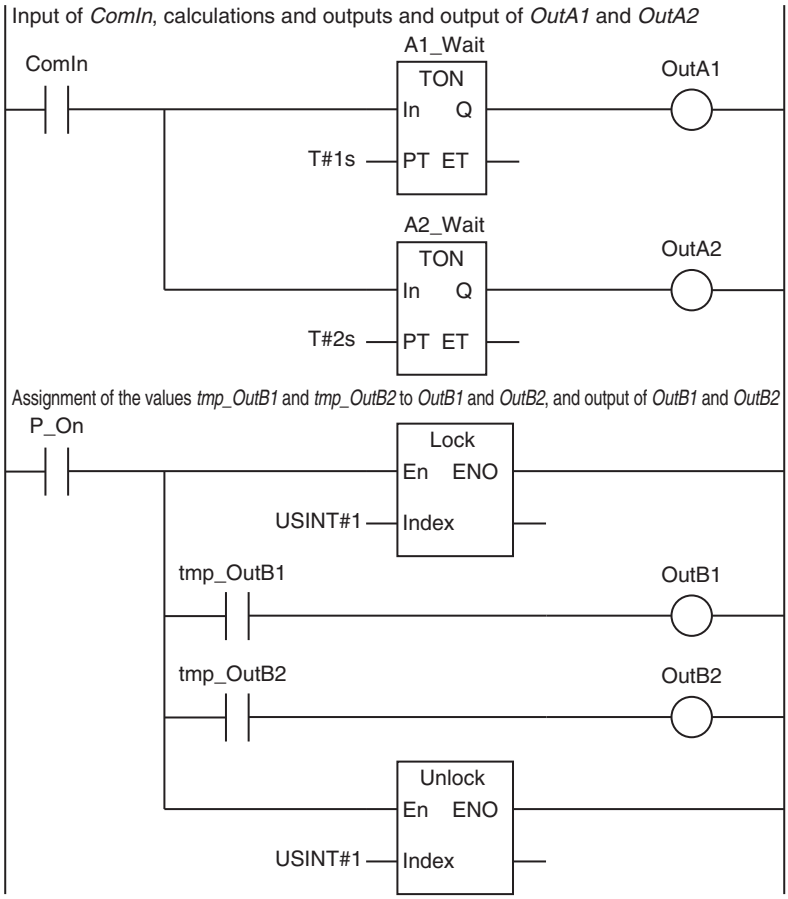
Name	Data type
A1_Wait	TON
A2_Wait	TON

External variable table

Name	Data type	Comment
ComIn	BOOL	Common input value
OutA1	BOOL	Output value A1

Name	Data type	Comment
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.

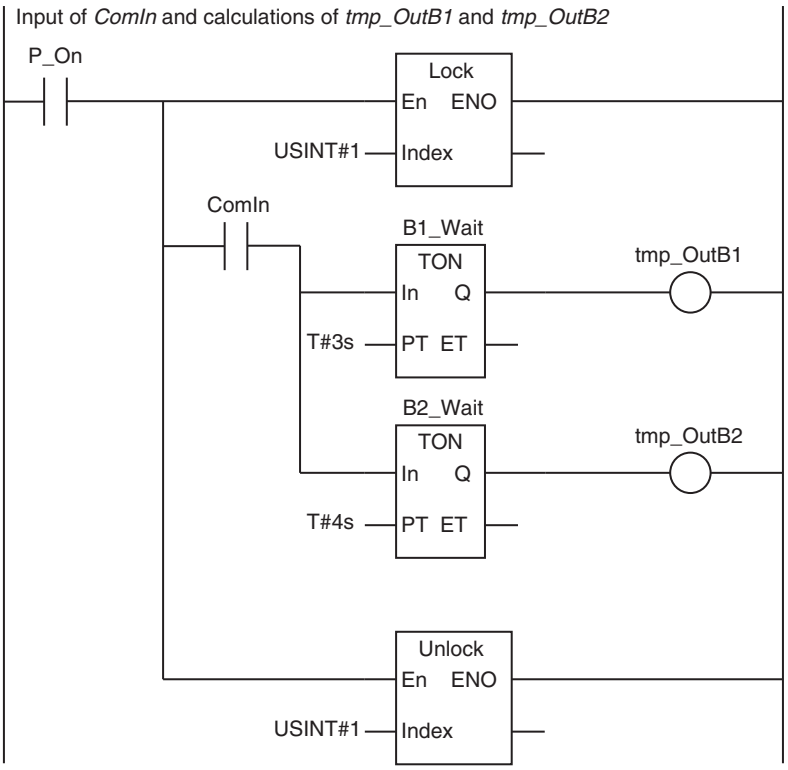
Internal variable table

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



5-8-2 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 4-11 for details.

Version Information

The CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the Initial Status setting for programs at the start of operation, the PrgStart instruction and the PrgStop instruction.

POUs That You Can Assign to Tasks

From 0 to 128 programs can be assigned to one task.
You can assign only program POUs. You cannot assign function block instances or functions directly to tasks. You cannot assign the same program to more than one task.

5-8-3 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	Not allowed.
	Execution priority	Always *4*.	---		
Task Name		Text string	Primary Task		
Period/ Execution Conditions	Task period *1	• NX701 CPU Units 125 µs, 250 µs to 8 ms (in 250-µs increments) • NX502 CPU Units 250 µs to 8 ms (in 250-µs increments) • NX102 CPU Units 1 ms to 32 ms (in 250-µs increments) • NX1P2 CPU Units 2 ms to 8 ms (in 250-µs increments)*2 • For NJ-series CPU Units 500 µs*3, 1 ms, 2 ms, or 4 ms	1 ms*4		
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 5-125 for details.	Detect.		

Parameter	Setting range	Default	Update timing	Changes in RUN mode
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 5-126 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 5-122 for details.	3%		

*1. The process data communications cycle (process data communications cycle 1) in the EtherCAT settings will be the same as this period.
*2. However, for the NX1P2-9B□□□□ CPU Unit, the setting range of the task period is 4 ms to 8 ms (in 250-μs increments).
*3. You can use this setting with the NJ301-□□□□ unit version 1.03 or later.
You cannot use this setting with the NJ101-□□□□.
*4. For the NX102-□□□□ CPU Unit, NX1P2-□□□□□□ CPU Unit, and NJ101-□□□□ CPU Unit, the default of the primary periodic task is 2 ms.
However, for the NX1P2 9B□□□□ CPU Unit, the default of the primary periodic task is 4 ms.

● Parameters for Priority-5 Periodic Task

Parameter		Setting range	Default	Update timing	Changes in RUN mode
Task Type		Specify the priority-5 periodic task. *1	---	When downloaded	Not allowed.
	Execution priority	Automatically set to 5.	---		
Task Name		Text string	PeriodicTask0		
Period/ Execution Conditions	Task period *2	• NX701 CPU Units 125μs, 250μs to 100ms (in 250-μs increments)	2 ms		
Task Period Exceeded Detection		The same as for the primary periodic task.	The same as for the primary periodic task.		
Task Timeout Detection Time					
Variable Access Time [%]					

*1. You can use the priority-5 periodic task only with the NX701 CPU Unit.
*2. The process data communications cycle 2 in the EtherCAT settings will be the same as this period.

● Parameters for Priority-16, Priority-17, and Priority-18 Periodic Tasks

Parameter		Setting range	Default	Update timing	Changes in RUN mode
Task Type		You can set any of the following. *Priority-16 periodic task*[*1] *Priority-17 periodic task* *Priority-18 periodic task*	---	When downloaded	Not allowed.
	Execution priority	Automatically set to *16*, *17*, or *18*.	---		
Task Name		Text string	PeriodicTask0		
Period/ Execution Conditions	Task period	• NX701 CPU Units Refer to *5-3-1 Specifications of Tasks for NX701 CPU Units* on page 5-11. • NX502 CPU Units Refer to *5-4-1 Specifications of Tasks for NX502 CPU Units* on page 5-31. • NX102 CPU Units and NX1P2 CPU Units Refer to *5-5-1 Specifications of Tasks for NX102 CPU Units and NX1P2 CPU Units* on page 5-51. • NJ-series CPU Units Refer to *5-6-1 Specifications of Tasks for NJ-series Controllers* on page 5-68.	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 [%]					

*1. You cannot use the priority-16 periodic task with the NX102 CPU Units or NX1P2 CPU Units.

● 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 downloaded	Not allowed.
Task Name	Text string	EvnetTask0		
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 *4-2-3 Task Settings* on page 4-7 for details.

5-9 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.

5-9-1 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.

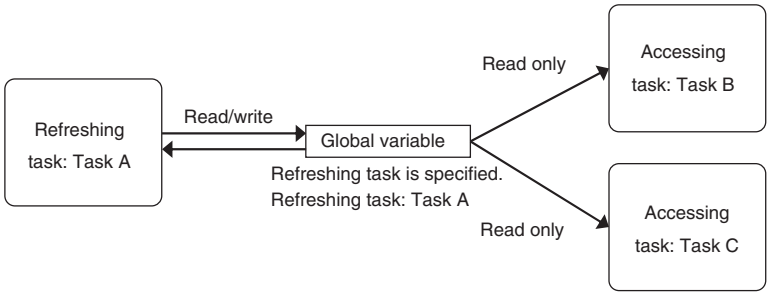
Method 1: Write the global variable from only one task and read the variable from the other tasks.
Use the settings for exclusive control of variables in tasks.
Method 2: Write the global variable from more than one task.
Use the task exclusive control instructions.

Method 1: Settings for Exclusive Control of Variables in Tasks

● Introduction

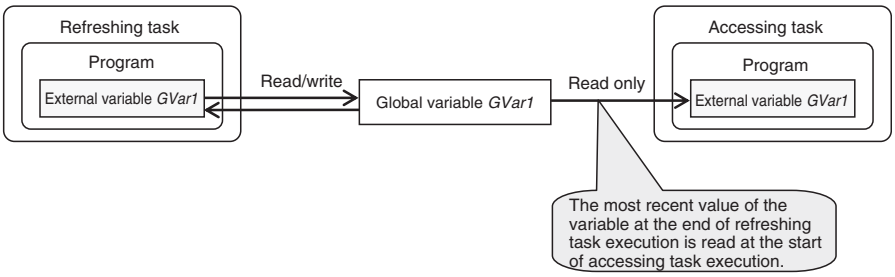
You can specify the task that refreshes a global variable and the tasks that access the global variable. This ensures the concurrency of the value of the global variable from the point of view of the tasks that access the variable.

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.



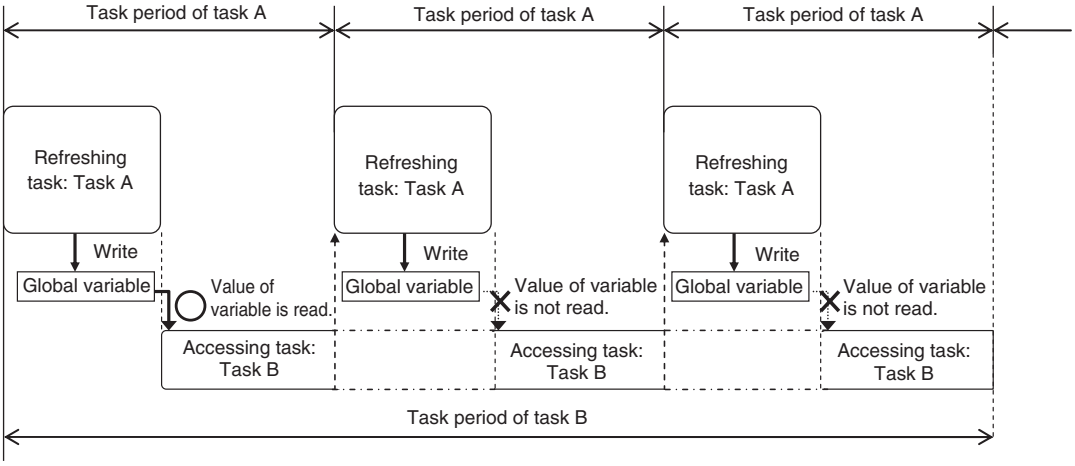
● Application Example

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.

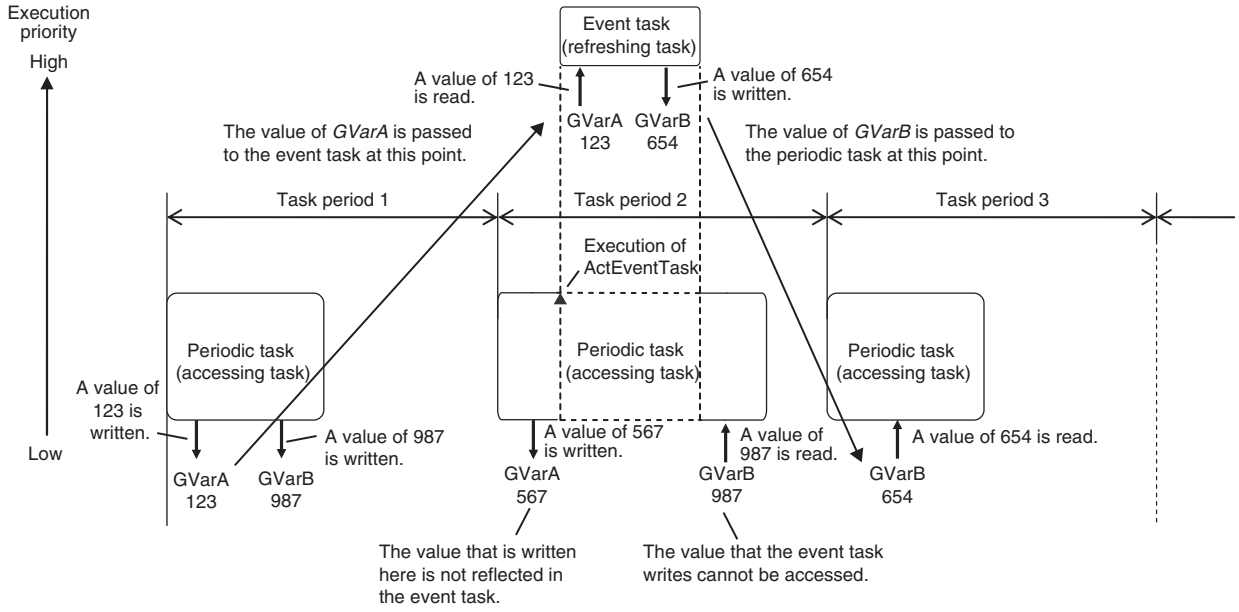


This will allow you to maintain the concurrency of the values of global variables within the tasks without performing any special programming.

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 *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for details on the Task_IsActive instruction.

● Restrictions

- Only one refreshing task can be set for each global variable. If it is necessary to write a global variable from more than one task, use the task exclusive control instructions described below to ensure concurrency.
- If you specify a refreshing task for a structure or union variable, you must specify only one refreshing task for the entire structure or union variable. You cannot specify a different refreshing task for different structure or union members.
- If you specify a refreshing task for an array variable, you must specify only one refreshing task for the entire array variable. You cannot specify a different refreshing task for different array elements.

Precautions for Correct Use

Do not write the value of a variable for which concurrency is required from any task that is not the refreshing task, e.g., do not write the value from the accessing task. If you read or write the value of a variable for which a refreshing task is set from any task that is not a refreshing or accessing task, the concurrency of the global variable may be lost. If you write such a program, a warning is given when the program is checked.

Additional Information

- You can use a data trace to sample an external variable for a global variable for which settings for exclusive control of variables in tasks are used. This allows you to sample the values of the global variable in the refreshing and accessing tasks in a data trace. Refer to *8-7-4 Data Tracing* on page 8-68 for information on data tracing.

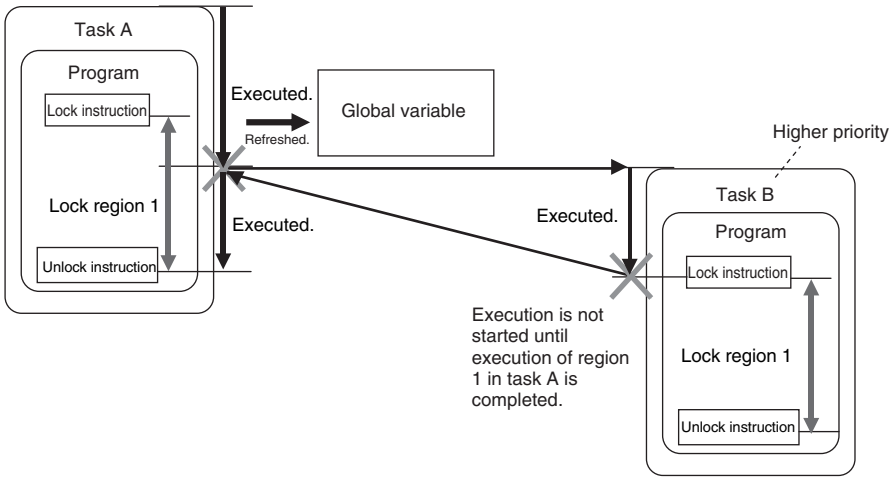
● Sysmac Studio Setting Procedure

Set the global variables for which to specify refreshing tasks, and set the accessing tasks in **Task Settings** - **Settings for Exclusive Control of Variables in Tasks** on the Sysmac Studio.
For details, refer to *Settings for Exclusive Control of Variables in Tasks* on page 4-12.

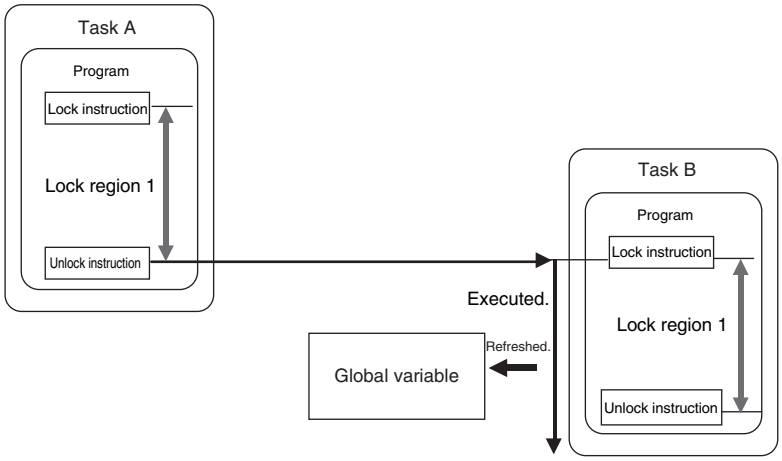
Method 2: Task Exclusive Control Instructions

Use the task exclusive control instructions (i.e., the Lock and Unlock instructions) when it is necessary to write the value of a global variable from more than one task while maintaining concurrency in the value of the variable.
The task exclusive control instructions create a lock region from one Lock instruction to the next Unlock instruction. If a lock region in one task is being executed, the lock regions with the same lock number in other tasks are not executed. If you place the instructions that write to the global variable in lock regions, the concurrency of the value is maintained even if you write the value of the variable from more than one task.
Refer to information on the Lock and Unlock instructions in the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for details.

Example:
In this example, task A and task B both have lock region 1. The priority of task B is higher than the priority of task A.
If the execution condition for task B is met during execution of lock region 1 in task A, execution of task A is paused during lock region 1 and task B is executed. However, in this case, lock region 1 in task A is not completed, so task B is paused before it processes lock region 1. When task B is paused, execution of lock region 1 in task A is started again.



When execution of lock region 1 in task A is completed, task A is paused again and the remainder of lock region 1 in task B is executed. The concurrency of the value of the global variable is maintained by implementing exclusive control of the write processing of the global variable between the tasks.



Precautions for Correct Use

- Do not make the locked regions any longer than necessary. If the lock regions are too long, the task execution period may be exceeded.
- Always use the Lock and Unlock instructions 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 gReq=TRUE AND ReqBusy=FALSE THEN (*Access the exclusive flag.*)
  ReqBusy:=TRUE;
  UserDefFB_ins.PutData:=gPar1;     (*Read the parameter from MainTask.*)
  UserDefFB_ins.PutDate:=gPar2;     (*Read the parameter from MainTask.*)
  gReq:=FALSE;                      (*Reset the exclusive flag.*)

  UserDefFB_ins.Execute:=TRUE;
END_IF;
Unlock(1);                        (*Stop an exclusive lock between tasks.*)
```

```
UserDefFB_ins();

IF UserDefFB_ins.Done:=TRUE THEN
  UserDefFB_ins.Execute:=FALSE;
  ReqBusy:=FALSE;
END_IF;
```

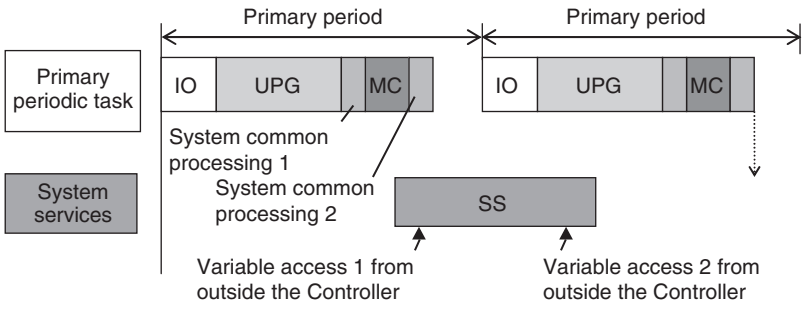
5-9-2 Variable Access from Outside the Controller

A variable access from outside the Controller is executed during the system service. The system service has a lower execution priority than tasks. This means, if multiple variables are accessed from outside the Controller, refreshing all variable values may not be completed in a task period. If refreshed variables and not-refreshed variables are mixed in the user program, the Controller may perform unintended operation.
To avoid this, make the variable access from outside the Controller be executed during the system common processing 2 of the task. By making this, multiple variable values can be securely refreshed in the same task period.

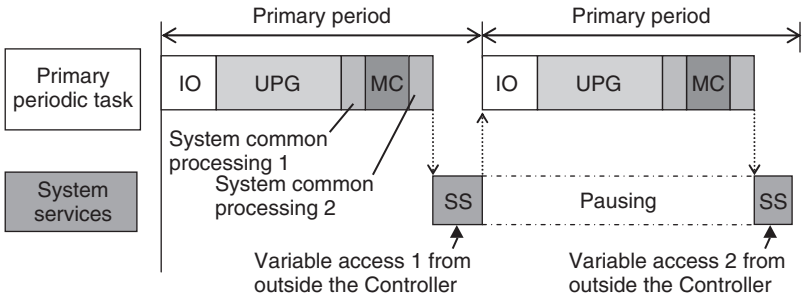
● Accessing Variables from Outside the Controller during the System Service

Whether you use an NX-series CPU Unit or an NJ-series CPU Unit, access to multiple variables may not be completed in the same task period.

● NX-series CPU Unit

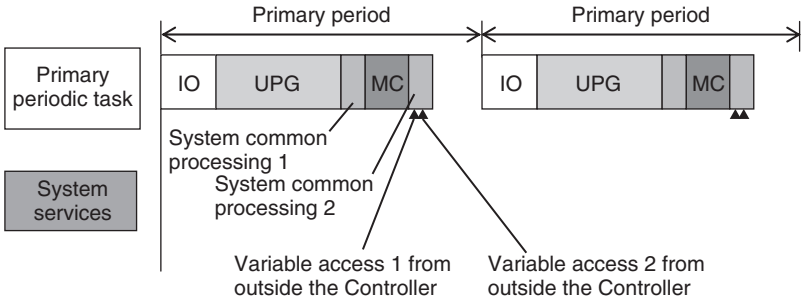


● NJ-series CPU Unit



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

Whether you use an NX-series CPU Unit or an NJ-series CPU Unit, access to multiple variables is 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*1
- CIP communications instruction from the host computer

*1. Tag data link communications used with the built-in EtherNet/IP port on the CPU Unit, NX-series EtherNet/IP Units, and CJ-series EtherNet/IP Units.

If the Sysmac Studio is used to access variables, it can only refresh the variable values during the system common processing 2 of the task. Values are accessed during the system services.

Tasks that Execute Variable Access during the System Common Processing 2

The tasks that execute variable access from outside the Controller during the *system common processing 2* are predetermined as follows according to the variable types.

Variable	Tasks that refresh values
Global variables specified in the settings for exclusive control of variables in tasks	The refreshing task specified in **Settings for Exclusive Control of Variable in Tasks** under **Configurations and Setup** − **Task Settings** on the Sysmac Studio.
Device variables for EtherCAT slaves	Tasks specified in **I/O Control Task Settings** under **Configurations and Setup** − **Task Settings** on the Sysmac Studio.
Device variables for CJ-series Basic I/O Units	
Device variables for CJ-series Special Units	Primary periodic task
Variables with AT specifications in memory used for CJ-series Units	

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

Note You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

Settings for Executing Variable Access during the System Common Processing 2

To access variables from outside the Controller during the system common processing 2, it is necessary to make the following two settings on the Sysmac Studio.

- Settings for exclusive control of variables in tasks (when the target variables are the global variables)
- Settings for variable access time

Settings for Exclusive Control of Variables in Tasks

If global variables are accessed from outside the Controller during the system common processing 2 of the task, it is necessary to make setting for exclusive control of variables in tasks. The exclusive control of variables in tasks refers to the function that specifies the task that can refresh the target global variable. This function prevents the target variable from being updated by other tasks or by other methods to access variables from outside the Controller.

For the details on the exclusive control of variables in tasks, refer to *Settings for Exclusive Control of Variables in Tasks* on page 4-12.

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

Precautions for Correct Use

- The number of variables is counted separately when the EtherNet/IP tag data links or NS/NA-series PT are accessed for the same variable.
- Add the variable access time that is set on the Sysmac Studio to the common processing time described in guidelines for the real processing times of tasks in the appendices. However, when tag data links are performed with the NX-series EtherNet/IP Unit, add the variable access time for tag data link communications.
- For the variable access time set on the Sysmac Studio, exclude the following variables from the calculation.
  a) Variables for EtherNet/IP tag data links for the NX-series EtherNet/IP Units

The values of the constants a to d in above equation vary depending on the model of the CPU Unit.

CPU Unit model	Constant value [μs]			
	a	b	c	d
NX701-□□□□	0.0005	0.033	2.67	7.22
NX502-□□□□	0.0012	0.0736	1.34	17.33
NX102-□□□□	0.0040	0.240	3.27	25.21
NX1P2-□□□□	0.0040	0.240	3.27	25.21
NJ501-□□□□	0.0010	0.490*1	1.41	6.68
NJ301-□□□□	0.0015*2	0.560*3	2.15	7.52
NJ101-□□□□	0.0015	1.070	3.83	10.29

*1. The constant value is *0.58* for a CPU Unit with unit version 1.02 or earlier.
*2. The constant value is *0.0009* for a CPU Unit with unit version 1.02 or earlier.
*3. The constant value is *1.03* for a CPU Unit with unit version 1.02 or earlier.

● Setting Variable Access Time

Set the variable access time in **Configurations and Setup** – **Task Settings** on the Sysmac Studio. The setting must be made for each task by entering the ratio to the task period. The default value is 3%. For the details on the settings, refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*.

● Example of Variable Access Time Setting

The following is an example of variable access time setting.
In this example, it is assumed that there are the following three variable accesses from outside the Controller to the task that operates in the NJ501-□□□□ CPU Units.

Access No.	Source of variable access	Total size of variables to access [bytes]	Number of variables to access	Number of accesses
1	EtherNet/IP tag data link	600	8	1
2	EtherNet/IP tag data link	200	4	1
3	CIP communications instruction	1,000	1	1

Using the equation, the variable access time for Access No.1 is calculated as follows.
Variable access time for Access No.1 = 600 * 0.001 + 8 * 0.49 + 1 * 1.41 + 6.68
= 12.61 [μs]

In the same way, you can calculate the access time for the other accesses and get the following values.

Access No.	Variable access time [μs]
1	12.61
2	10.25
3	9.58

If only one of these accesses occurs in one task period, you set the variable access time to the one for Access No.1, which requires the longest access time.
The variable access time for Access No.1 is 12.61 μs. Therefore, when the task period is 500 μs, the variable access time is set to 12.61/500 ≈ 3%.

If every access occurs once in one task period, the variable access time is calculated with the equation as follows.
Variable access time = (600 + 200 + 1000) * 0.001 + (8 + 4 + 1) * 0.49 + (1 + 1 + 1) * 1.41 + 6.68 = 19.08 [μs]

When the task period is 500 μs, the variable access time is set to 19.08/500 ≈ 4%.

● Processing in the Case That Actual Variable Access Time Became Longer Than Set Value

If actual variable access time became longer than the set value, the following processing is performed depending on the number of variable accesses in one task period.
Set a sufficiently long access time so that multiple variable accesses can be completed within a task period.

Number of variable accesses in one task period	Processing
Multiple times	Variable accesses are executed for the number of times that can be completed within the set variable access time. The rest of accesses that could not be done will be executed in the next task period. This means, the multiple variable accesses cannot be completed within the same task period.
Once	Variable accesses continue even after the set variable access time is exceeded. This means, the task execution time gets longer.

5-10 Errors Related to Tasks

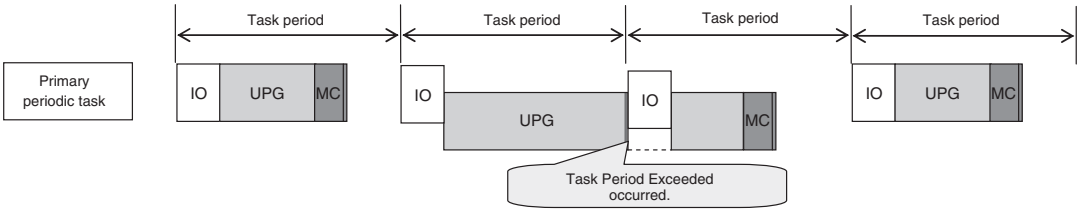
This section describes the following errors.

- Task Period Exceeded
- Motion Control Period Exceeded
- Task Execution Timeout
- I/O Refreshing Timeout Error
- Insufficient System Service Time Error

Task Period Exceeded

A *Task Period Exceeded* error occurs if the task execution time exceeds the specified task period. This is a minor fault level Controller error. Operation continues even when this error occurs.
It can occur for the primary periodic task and periodic tasks.

You can also disable the Task Period Exceeded errors with a setting. Select **Task Settings** − **Task Settings** on the Sysmac Studio and set it in **Task Period Exceeded Detection** setting. 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 event log and the following system-defined variables if task processing is not completed within the period: *_<TaskName\>_Exceeded* (Task Period Exceeded Flag), *_<TaskName>_ExceededCount*(Task Period Exceeded Count), *_ErrSta* (Controller Error Status).
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 slaves*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.
NX Units on the CPU Unit*2	
X Bus Units*3	
Built-in I/O*4	
CJ-series Units	I/O is not refreshed until execution of the task is completed.

*1. This includes NX Units on EtherCAT Slave Terminals.

*2. You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*3. You can use X Bus Units only on the NX502 CPU Units.
*4. You can use built-in I/O only with the NX1P2 CPU Units.

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.

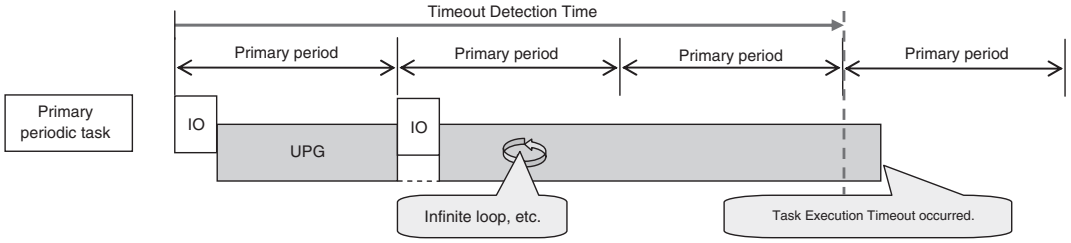
Motion Control Period Exceeded

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

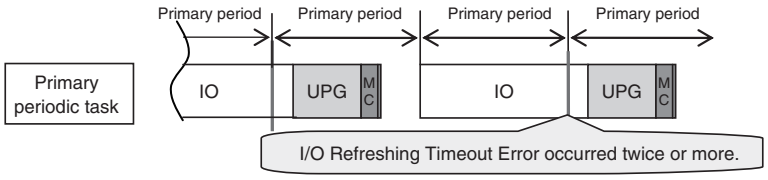
A Task Execution Timeout error occurs if task processing is not completed within the specified timeout detection time.
This is a major fault level Controller error. Execution of the user program stops when the error occurs.
This error also occurs when normal task operation is not possible due to errors 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 CPU Unit reset.

I/O Refreshing Timeout Error

An I/O Refreshing Timeout Error occurs when I/O refreshing for the primary periodic task or priority-5 and priority-16 periodic tasks is not completed within the period twice or more in a row.
This is a major fault level Controller error. Execution of the user program stops when the error occurs.



Error name	Error level	Correction
I/O Refreshing Timeout Error	Major fault	Review the task settings and download the project again. The power supply must be cycled or the CPU Unit reset.

Insufficient System Service Time Error

With an NJ-series CPU Unit, an Insufficient System Service Time Error occurs if the system service execution time that is specified in the System Service Monitoring Settings cannot be obtained.
This is a major fault level Controller error. Execution of the user program stops when the error occurs.

Error name	Error level	Correction
Insufficient System Service Time Error	Major fault	Review the task settings and the system service monitoring settings and download the project again. The power supply must be cycled or the CPU Unit reset.

Additional Information

NX-series CPU Units are designed to always secure sufficient time for system service execution, so the System Service Monitoring Settings are not provided. Also an Insufficient System Service Time Error will not occur.

5-11 Monitoring Task Execution Status and Task Execution Times

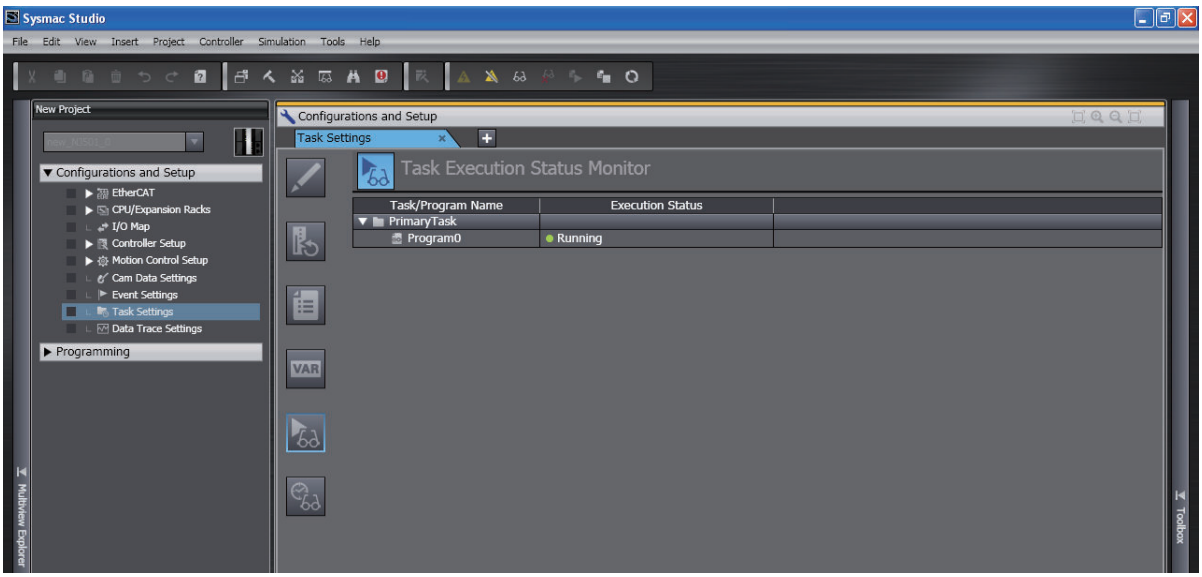
You can use online operations from the Sysmac Studio to monitor the task execution status and task execution times.

Monitoring Task Execution Status

You can monitor the execution status of the programs in all of the tasks (started/stopped) from the Sysmac Studio.

● Sysmac Studio Operation

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



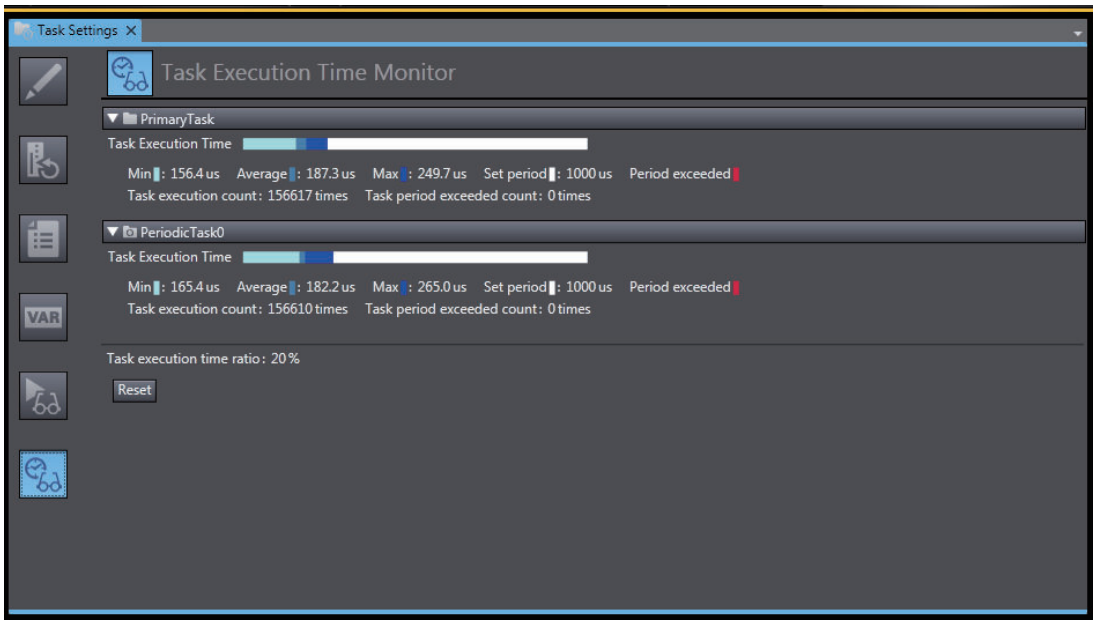
Task Execution Time Monitor

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

● Values You Can Monitor from the Sysmac Studio

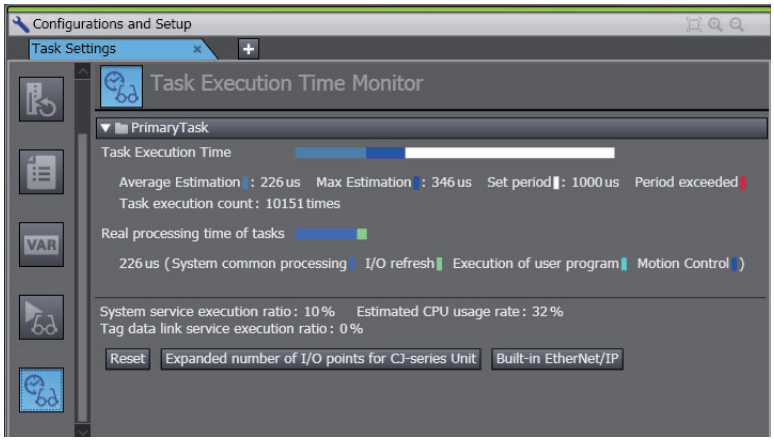
The display depends on whether you connect to the physical Controller or to the Simulator.

Connected to the Controller

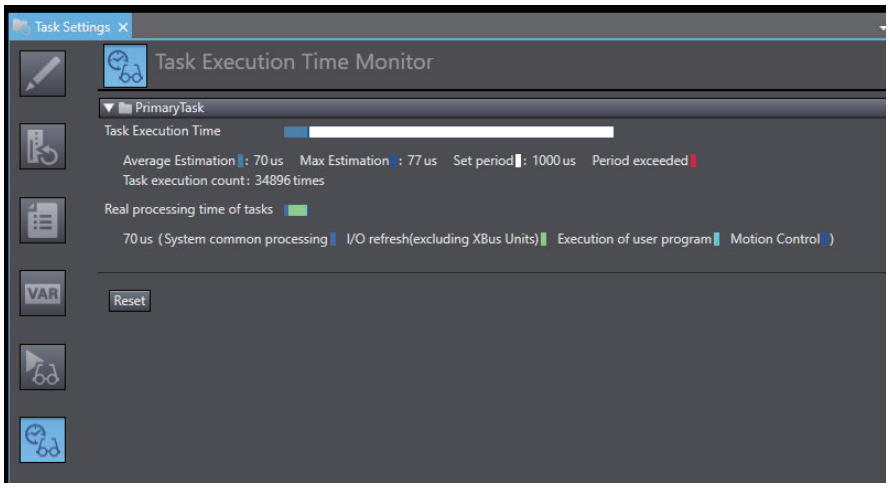


Connected to the Simulator

- For CPU Units other than NX502 CPU Units



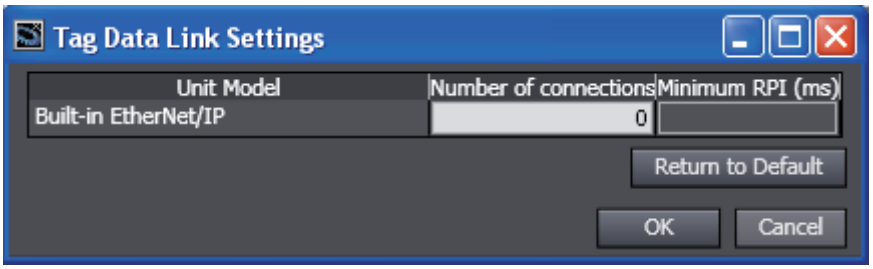
- For NX502 CPU Units



Additional Information

- To check or estimate the system service execution time, the task execution time ratio, system service execution ratio, estimated CPU usage rate, tag data service execution ratio, expanded number of I/O points for CJ-series Unit, and built-in EtherNet/IP are displayed for NJ-series CPU Units.
  The above information are not displayed for NX-series CPU Units because the system services are executed at the required time without being affected by the task.
- If an X Bus Unit is used, the X Bus Unit processing time is not included in the I/O refresh time when connected to the Simulator of the Sysmac Studio.

Built-in EtherNet/IP Dialog Box for Simulator Connection



The parameters are listed in the following table.

Port or Unit that is used	Parameter	Description	Set value	Default
Built-in EtherNet/IP port on the CPU Unit	Number of connections	This is the number of tag data link connections.	0 to 32	0
	Minimum RPI	This is the lowest packet interval (RPI) that is set for all of the tag data link connections.	1 ms to 10 s in 1-ms increments	---
CJ-series EtherNet/IP Unit (CJ1W-EIP21)*1	Number of tags	This is the number of tags in the tag data links.	0 to 256	0

*1. Entries are made on the Expanded number of I/O points for CJ-series Unit Dialog Box.

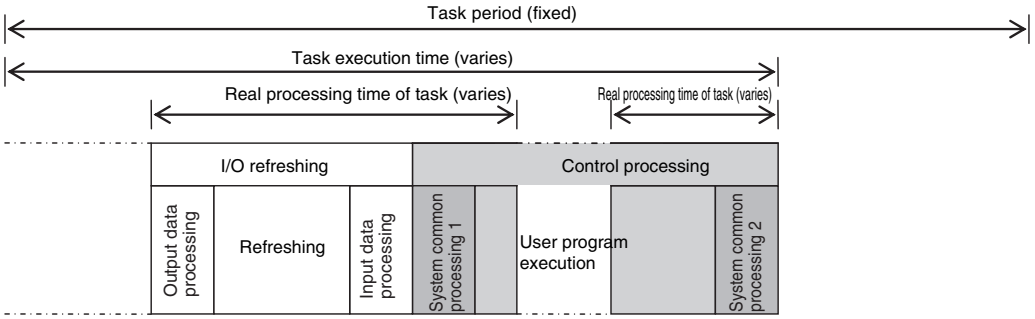
You can monitor the following items.

Monitor item		Description	Connected to the Controller	Connected to the Simulator
Task execution time*1	Min.	The minimum value of the task execution time.	Displayed.	Not displayed.
	Average	The average value of the task execution time.		Displayed.
	Max.	The maximum value of the task execution time.		
Set period		The specified task period. *2		
Period exceeded		If the task execution time exceeds the task period (i.e., if the *Task Period Exceeded Flag* system-defined variable is TRUE), the amount by which the time was exceeded is displayed in the bar. *2		
Task execution count		Displays the number of executions of the task. The value of the *Task Execution Count* system-defined variable is displayed.		
Real processing time of tasks*3		The time ratios are displayed with bars for the system common processing, I/O refreshing, user program execution, and motion control processing. (Specific time values are not displayed.)	Not displayed.	Displayed.
	Average estimation	The estimated average value of the real processing time of task is displayed.		
	Max estimation	The estimated maximum value of the real processing time of task is displayed.		
Task execution time ratio*4		The percentage of the total task execution time per unit time.	Displayed.	Not displayed.

*1. This is the actual time required from the point that task execution was started until it was completed. This interval includes both the time to execute other tasks and the time for system services that were executed from when task execution was started until it was completed.
Only the primary periodic task is displayed when a Simulator for an NX701 CPU Unit is connected.

*2. This item is not displayed for event tasks.

*3. This interval is the time required to execute only the task itself.
It is the same as the task execution time for the primary periodic task.
For periodic tasks, this is the task execution time minus the time to execute other tasks and the time for system services that were executed between the point that the execution condition is met until execution is completed.



*4. This item was added for version 1.12 of the Sysmac Studio. It is not displayed for an NX-series CPU Unit.

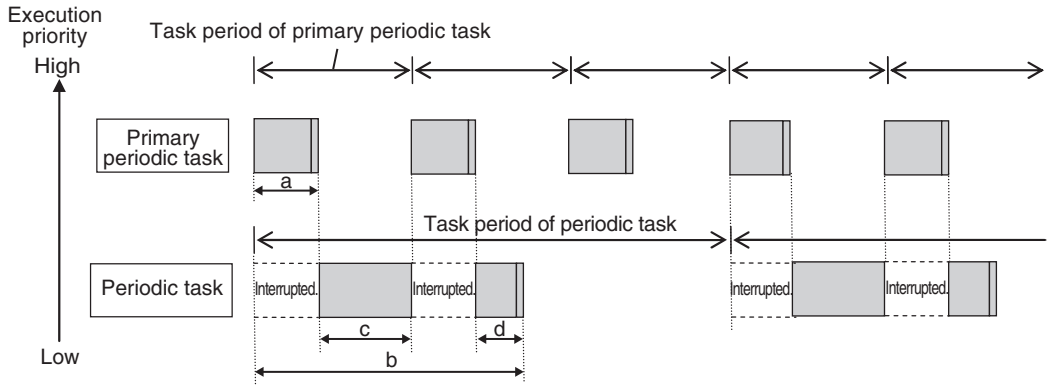
Precautions for Correct Use

The above values when connected to the Simulator of the Sysmac Studio may contain more error in comparison to the times when connected to the physical Controller. Use them as guidelines. Always confirm operation while connected to the physical Controller to study the designs and before starting actual system operation.

Meaning of the Task Execution Time and the Real Processing Time of the Task

The Task Execution Time and Real Processing Time of Tasks that are displayed in the Monitor View for the execution time of tasks are described in the following table. The Real Processing Time of Tasks shows only the time the Simulator was connected.

Displayed time	Meaning
Task execution time	This is the time from when the task period starts until task processing ends. However, this includes the time when the periodic task is interrupted for execution of tasks with higher execution priorities.
Real processing time of task	This is the time from when the task period starts until task processing ends. However, this does not include the time when the periodic task is interrupted for execution of tasks with higher execution priorities.



a: Task execution time of primary periodic task
b: Task execution time of periodic task
c+d: Real processing time of periodic tasks (only when the Simulator is connected).

5-12 Task Design Methods and I/O Response Times

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

The primary periodic task and periodic tasks of an NJ/NX-series Controller operate according to the specified task periods.
If the actual execution time exceeds the task period, an error occurs.

This section uses an example that consists of one primary periodic task to describe estimation and appraisal methods.

Precautions for Safe Use

The task execution times in the physical Controller depend 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.

5-12-1 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.

Precautions for Correct Use

The task execution time calculated with the Simulator on the Sysmac Studio does not include the X Bus Unit processing time. If the X Bus Units are used, calculate the task execution time by desktop calculation.

● Desktop Calculations

First, refer to *A-2 Calculating Guidelines for the Real Processing Times of Tasks for the NX701 System* on page A-30, *A-3 Calculating Guidelines for the Real Processing Times of Tasks for the NX502 System* on page A-42, *A-4 Calculating Guidelines for the Real Processing Times of Tasks for the NX102 System* on page A-53, *A-5 Calculating Guidelines for the Real Processing Times of Tasks for the NX1P2 System* on page A-64 and *A-6 Calculating Guidelines for the Real Processing Times of Tasks for the NJ-series System* on page A-75 to make a rough estimate of the average task execution time on paper.
You cannot estimate the maximum value on paper.

● Estimating with the Simulator on the Sysmac Studio

Use the Task Execution Time Monitor of the Simulator on the Sysmac Studio to estimate the average and maximum task execution times.
Use the following procedure to check operation on the Simulator.

1 Create the Unit and slave configurations, create the global variables and device variables, and create the axes (to create the Axis Variables).

2 Create the programs to check.

3 Set up the tasks and build the project.

4 Start the Simulator in Execution Time Estimation Mode.

5 Select the relevant hardware revision in the Unit that the hardware revision is displayed.

6 With an NJ-series CPU Unit, set the **Expanded number of I/O points** in the CJ-series Unit parameters in the **Task Execution Time Monitor** to create user-defined variables for specified CJ-series Special Units. Also set the sizes of the expansion areas (e.g., fixed I/O allocation areas for the DeviceNet Unit) for AT specifications (i.e., the number of output words and the number of input words).
These sizes are used to calculate the I/O refresh time for the specific Special Units.

7 Estimate the task execution times in the Task Execution Time Monitor.

You can check the following values in the Task Execution Time Monitor when you start the Simulator in Execution Time Estimation Mode.

- The average and maximum values of the task execution time
- The average and maximum values of the real processing times of the tasks
- Bar graph that shows the system common processing time, I/O refresh time, user program execution time, and motion control time
- CPU usage

Note 1. Only the primary periodic task is displayed for an NX-series CPU Unit.
Note 2. If an X Bus Unit is used, the X Bus Unit processing time is not included in the I/O refresh time when connected to the Simulator of the Sysmac Studio.

Additional Information

You can check the following values when connected to the Simulator of the Sysmac Studio. You cannot check these values when connected to the physical Controller.

- CPU usage:
  Displays how much of the task period is used by the total of the maximum estimated task processing time, the tag data link service execution time ratio, and the system service processing time of an NJ-series CPU Unit (as specified in the system service monitoring settings). If CPU usage exceeds 100%, it means that there is not sufficient time for task processing and the system service monitoring settings.
  This is not displayed for an NX-series CPU Unit.
- Real processing time of tasks:
  This is the time that was required for the task from when task execution is started until it is completed. The time to execute other tasks that were executed from when task execution was started until it was completed is not included.

● Calculating Times on the Physical Controller

You can check the following values in the Task Execution Time Monitor when you are connected to the physical Controller.

- The minimum, average, and maximum values of the task execution time

- Set period
- Number of times a task is executed (Task Execution Count)
- Number of times the task period was exceeded (Task Period Exceeded Count)
- Task execution time ratio

The maximum values that are displayed on the Sysmac Studio are the results of operation on the physical Controller.
As described previously, the maximum value of the task execution time varies depending on the internal status of the physical Controller.
As a result, the maximum values obtained here may be exceeded in actual operation. Use the obtained values or the larger values in the following calculating results as guidelines of maximum values.

NX-series CPU Units

(Average value of task execution time + (Average value of task execution time − Minimum value of task execution time)) x 1.2 + 25 μs

NJ-series CPU Units

- Task period of 500 μs:
  Average value of task execution time + (Average value of task execution time − Minimum value of task execution time) + 100 μs
- Task period of 1, 2, or 4 ms:
  Average value of task execution time + (Average value of task execution time − Minimum value of task execution time) + 120 μs

For NJ-series CPU Units, you can also check whether sufficient system service execution time is obtained.
Insufficient system service execution time may decrease the online operations of the Sysmac Studio or the communications response performance with external devices such as an HMI. For NJ-series CPU Units, system services are executed during the unused time between execution of the tasks.
Use the following guideline for the task execution time ratio.

CPU Unit model	Guideline for task execution time ratio
NJ501-□□□□	90% or less
NJ301-□□□□	80% or less
NJ101-□□□□	70% or less

Version Information

- The task execution time ratio is displayed when an NJ-series CPU Unit is used with Sysmac Studio version 1.12 or higher.
- Use an NJ101-□□□□ CPU Unit with Sysmac Studio version 1.13 or higher. You cannot use an NJ101-□□□□ CPU Unit with Sysmac Studio version 1.12 or lower.
- On an NJ101-□□□□, NJ301-□□□□, NJ501-□□□□, NX102-□□□□, and NX1P2-□□□□□□□□ of unit version 1.47 or higher and an NX701-□□□□ of unit version 1.28 or higher, performance (task time) will be improved if the number of registered device variable is large.

Precautions for Correct Use

NX-series CPU Units are designed to always secure sufficient time for system service execution.
Therefore, the task execution time ratio is not displayed.

Additional Information

The average values of the task execution times that are displayed for task execution time monitoring are the averages for 10 task execution times.

5-12-2 Examples of Task Design

This section provides the design procedure for a project that consists of only the primary periodic task. In any actual application or for specific conditions, you need to consider any elements for which the design procedure must be changed. This example is therefore for reference only.

1. Find the I/O response times that are required for the system from the equipment specifications.
2. From the system I/O response times, determine the task period for the primary periodic task.
3. See if the task execution time fits into the task period that you determined.
   Then, work on paper or use the Task Execution Time Monitor of the Sysmac Studio to estimate the average and maximum values of the task execution time.
4. For NJ-series CPU Units, see if the system service times are within the monitor settings.
   If you use the Sysmac Studio, check the CPU usage.
5. Use the physical Controller to see if the task execution time fits into the task period.
   Place the Sysmac Studio online with the physical Controller and use Task Execution Time Monitor to check the task execution times.

- **If only the primary periodic task is too large to fit within the specified task period, consider separating it into periodic tasks as follows.**
  - For the NX701 CPU Units, among processes required for device control, assign those which require the high-speed control to the primary periodic task and other processes to the priority-5 periodic task.
  - To reduce the task execution time, use the Enable Program (PrgStart) and Disable Program (PrgStop) instructions to execute the program assigned to the primary periodic task only when necessary.
  - If the primary periodic task will still not fit within the specified task period after these measures, among the processes of primary periodic task, assign those which do not require the high-speed or high-accuracy control to the priority-16 periodic task or the priority-17 periodic task.
    However, if variables such as the Axis Variables are accessed between the primary periodic task and the priority-16 periodic task, the task execution time for the primary periodic task may be longer. For details, refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)*.
  - If the primary periodic task will still not fit within the specified task period even after you take all of these measures, change the task period for the primary periodic task.

● **If a task is separated, the periodic task will vary greatly with the unused time for primary periodic task execution.**

For a periodic task, use twice the average and maximum values calculated for the task execution time to set the task period and then fine-tune the setting from there.

5-12-3 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

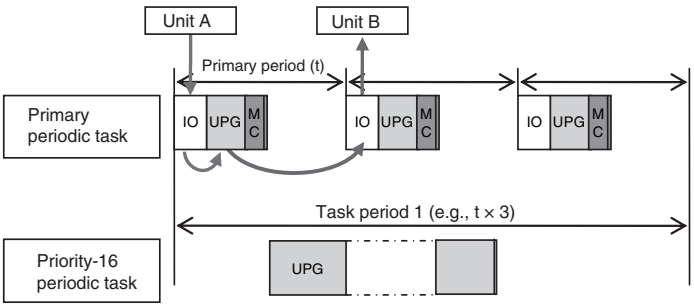
I/O refreshing between Basic I/O Units and external devices is performed in the task to which I/O refreshing is assigned.
The I/O response times that include EtherCAT communications times are given below.

● **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.

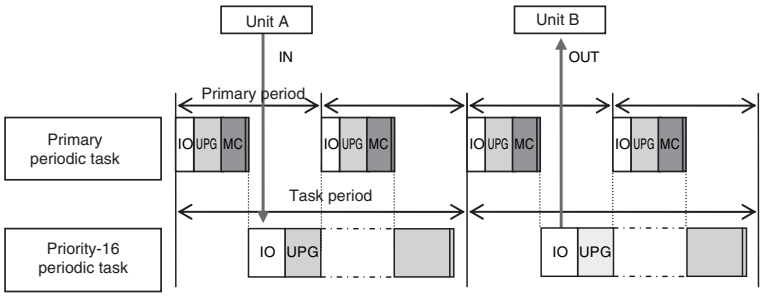
Maximum I/O response time = Primary period × 2

● **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

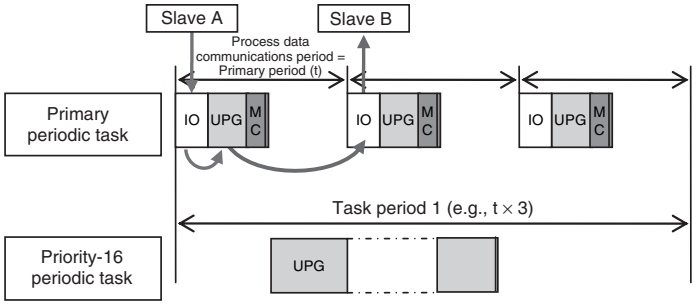
For EtherCAT slaves, EtherCAT communications with external devices is performed for I/O refreshing in the primary periodic task.
The I/O response times that include EtherCAT communications times are given below.

● 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.

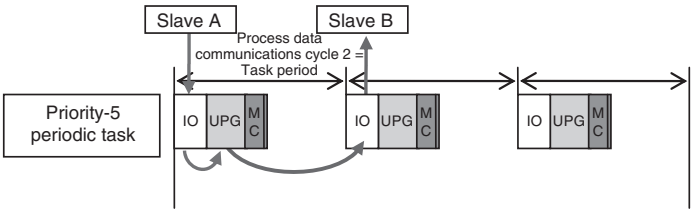
Maximum I/O response time = Primary period (= process data communications cycle) × 2

● Performing Control with the Programs in the Priority-5 Periodic Task

When you perform control with the user program in the priority-5 periodic task, the NX701 CPU Unit makes a response in the following I/O response time.

Minimum I/O response time = Priority-5 periodic task period (= process data communications cycle 2)

Example: Controlling EtherCAT Input Slave A and EtherCAT Output Slave B with the Priority-5 Periodic Task



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

However, the I/O response time may be as follows depending on the timing of the input from the slave.

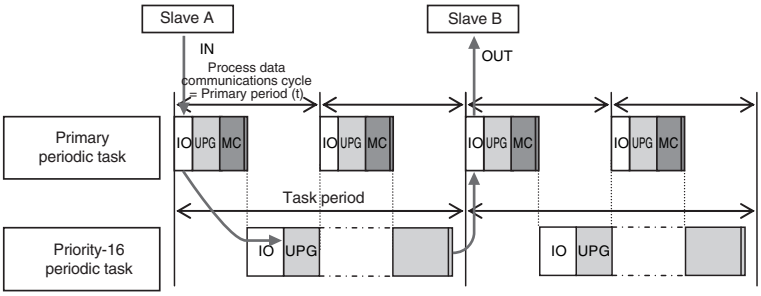
Maximum I/O response time = Priority-5 periodic task period (= process data communications cycle 2) x 2

● 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

Additional Information

Refer to the *NX-series EtherCAT Coupler Units User's Manual* (Cat. No. W519) for the I/O response times for EtherCAT Slave Terminals that the NX Units are mounted on the EtherCAT Coupler Units.

Sequence Control with NX Units on the CPU Unit

For NX Units on the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, data exchange with external devices is performed for I/O refreshing in the primary periodic task.

For NX Units on the CPU Unit, the following describes both of the I/O response times for the CPU Unit inside and I/O response times that include NX Unit processing times.

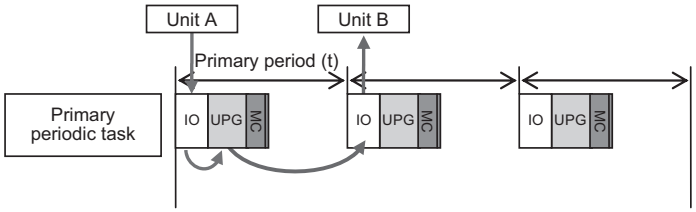
The I/O response times for the CPU Unit inside are given below.

● 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 Input Unit A and Output 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.

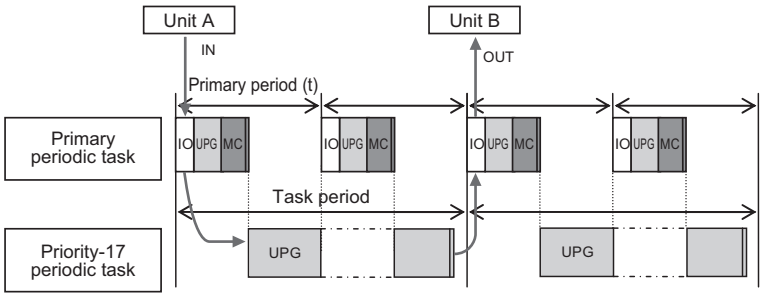
Maximum I/O response time = Primary period × 2

● Performing Control with the Programs in the Priority-17 Periodic Task

The Controller makes a response in the following I/O response time.

I/O response time = Priority-17 periodic task period

Example: Controlling Input Unit A and Output Unit B with the Priority-17 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-17 periodic task period × 2

Next, the I/O response times that include NX Unit processing times are given below.
The I/O response time is the time required for the following processing: The CPU Unit processes an external signal input to one NX Unit, and another NX Unit outputs the processed result as an external signal.

The I/O response times that include NX Unit processing times depend on the I/O refreshing method of the NX Unit. The following provides a description to perform control with the programs in the primary periodic task for each of the I/O refreshing method of the NX Unit, including synchronous I/O refreshing, time stamp refreshing, and Free-Run refreshing.

The elements in the formulas are as follows:

Tnx-InProc	:Input data processing time of the NX Unit
Tnx-OutProc	: Output data processing time of the NX Unit
Tnx-Indelay	: Input delay time of the NX Unit
Tnx-Outdelay	: Output delay time of the NX Unit

For each element, there is a unique value for each type of NX Unit. Refer to the appendix of the *NX-series Data Reference Manual* (Cat. No. W525-E1-07 or later) for the value of each element.

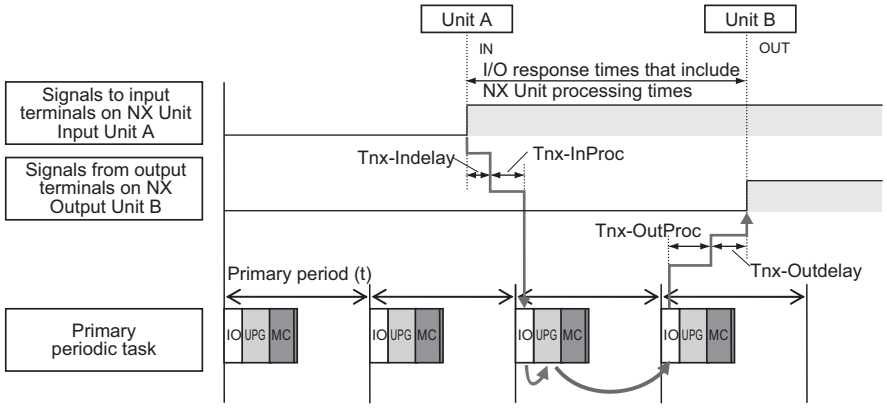
● Synchronous I/O Refreshing

With synchronous I/O refreshing, when there is more than one NX Unit that is connected to the CPU Unit, the inputs are read or the outputs are updated simultaneously for all NX Units that support synchronous I/O refreshing. Therefore, Tnx-InProc (input data processing time of the NX Unit) and Tnx-OutProc (output data processing time of the NX Unit) are the longest time of the NX Units that support synchronous I/O refreshing.

The Controller makes a response in the following I/O response time.

Minimum I/O response time = Tnx-InDelay + Tnx-InProc + Primary period + Tnx-OutDelay + Tnx-OutProc

Example: Controlling Input Unit A and Output 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 = Tnx-InDelay + Tnx-InProc + Primary period x 2
+ Tnx-OutDelay + Tnx-OutProc + NX Unit processing time*1

*1. Refer to *NX Unit Processing Time* on page A-66 for information on the NX Unit processing time.

Time Stamp Refreshing

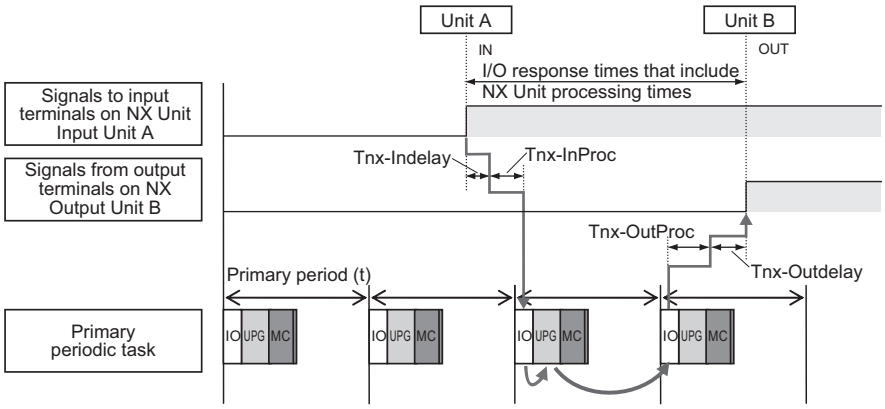
The I/O response time for time stamp refreshing is the specific time required to produce the output after the input changed time. You specify the time in the user program. Specify a time that has a sufficient leeway to ensure that the output is produced at the expected time.
The minimum specified time for which the expected output is produced is defined as the maximum I/O response time for time stamp refreshing. It is described in the following.

With input refreshing with input changed times for time stamp refreshing, when there is more than one NX Unit that is connected to the CPU Unit, the inputs for all NX Units that support synchronous input refreshing and the changed times for all NX Units that support input refreshing with input changed times are loaded at the same time. Therefore, Tnx-InProc (input data processing time of the NX Unit) and Tnx-OutProc (output data processing time of the NX Unit) are the longest time of the NX Units that support synchronous I/O refreshing.

The Controller makes a response in the following I/O response time.

Minimum I/O response time = Tnx-InDelay + Tnx-InProc + Primary period + Tnx-OutDelay + Tnx-OutProc

Example: Controlling Input Unit A and Output 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 = Tnx-InDelay + Tnx-InProc + Primary period x 2 + Tnx-OutDelay + Tnx-OutProc + NX Unit processing time*1

*1. Refer to *NX Unit Processing Time* on page A-66 for information on the NX Unit processing time.

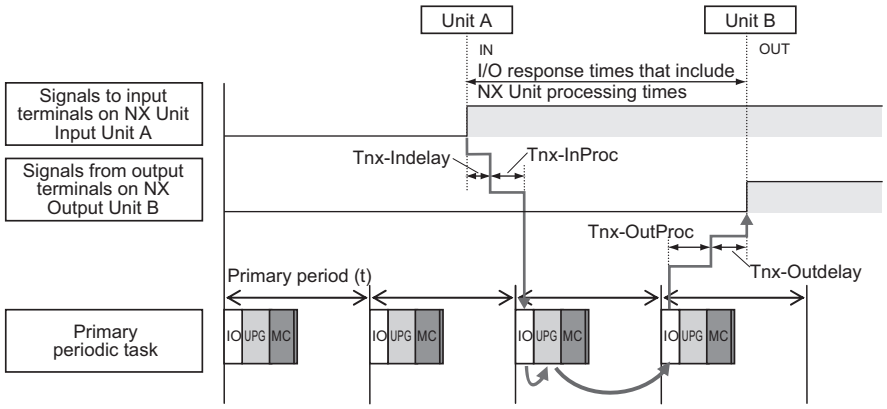
Free-Run Refreshing

With Free-Run refreshing, the refresh cycle of the NX bus and the I/O refresh cycle of the NX Units operate asynchronously.

The Controller makes a response in the following I/O response time.

Minimum I/O response time = Tnx-InDelay + Tnx-InProc + Primary period + Tnx-OutDelay + Tnx-OutProc

Example: Controlling Input Unit A and Output 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 and the timing of the output to the Unit.

Maximum I/O response time = Tnx-InDelay + Tnx-InProc × 2 + Primary period x 2
+ Tnx-OutDelay + Tnx-OutProc × 2+ NX Unit processing time*1

*1. Refer to *NX Unit Processing Time* on page A-66 for information on the NX Unit processing time.

Additional Information

With Free-Run refreshing, the Input Unit, CPU Unit and Output Unit operate asynchronously. Therefore, an offset of timing that the Input Unit reads signals from input terminals, an offset of timing that the CPU Unit reads the input data from the Input Unit, and an offset of timing that the Output Unit outputs signals to output terminals will occur.
Tnx-InProc, primary period, and Tnx-OutProc will be the longest time of an offset of timing for each Unit.

Performing Motion Control with Motion Control Instructions

Motion control instructions access the Servo Drives and encoder input slaves that are assigned to axes.
For the NX102 CPU Units and NX1P2 CPU Units, motion control instructions can be used in the primary periodic task.
The motion control instructions are processed in the "motion control processing (MC)" section of the primary periodic task.

For the NX502 CPU Units, and NJ-series CPU Units, motion control instructions can be used in the primary periodic task and in a priority-16 periodic task.
In either case, the motion control instructions are processed in the "motion control processing (MC)" section of the primary periodic task.

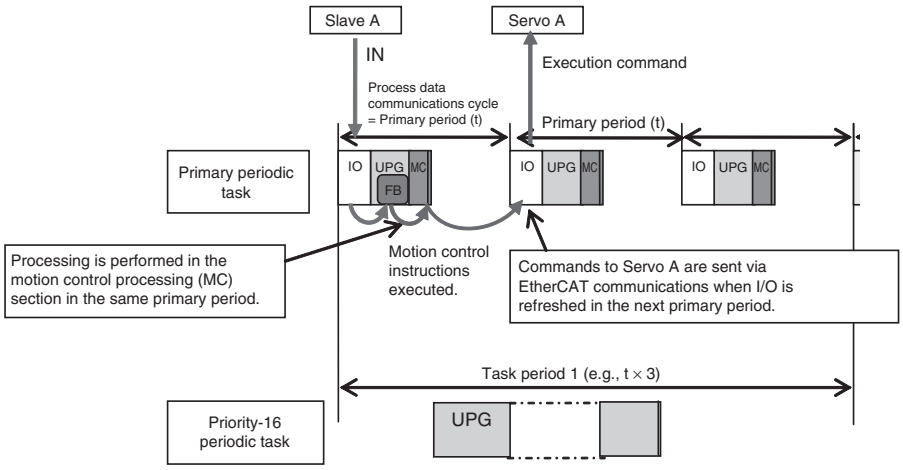
For the NX701 CPU Units, motion control instructions can be used in the primary periodic task, in the priority-16 periodic task, and in the priority-5 periodic task.
The motion control instructions included in the primary periodic task and in a priority-16 periodic task are executed in the "motion control processing (MC)" section of the primary periodic task.
The motion control instructions included in a priority-5 periodic task are executed in the "motion control processing (MC)" section of the priority-5 periodic task.

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

● Programming Motion Control Instructions in the Primary Periodic Task

The motion control instructions are processed in the next motion control processing (MC) section of the primary periodic task. The results of processing are output to the EtherCAT slave or NX Unit to which the axis is assigned during the I/O refresh period in the next primary periodic task.
The Controller makes a response in the following I/O response time.

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



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

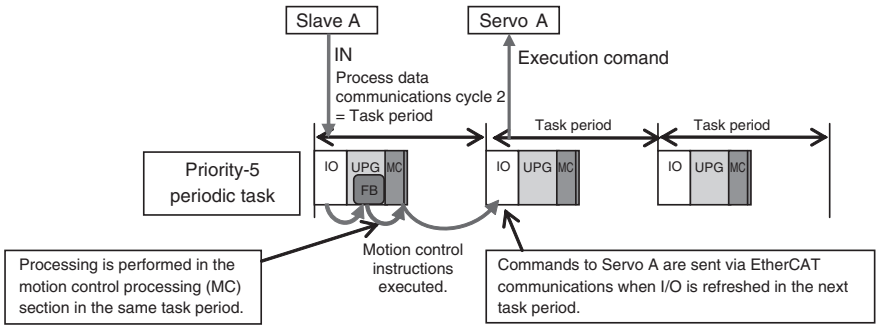
However, the I/O response time may be as follows depending on the timing of the input from the slave.

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

● Programming Motion Control Instructions in the Priority-5 Periodic Task

The motion control instructions are processed in the immediate motion control processing (MC) section in the priority-5 periodic task. The results of processing are output to the EtherCAT slave to which the axis is assigned during the I/O refresh period in the next priority-5 periodic task.
The Controller makes a response in the following I/O response time.

I/O response time = Priority-5 periodic task period (= process data communications cycle)



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

However, the I/O response time may be as follows depending on the timing of the input from the slave.

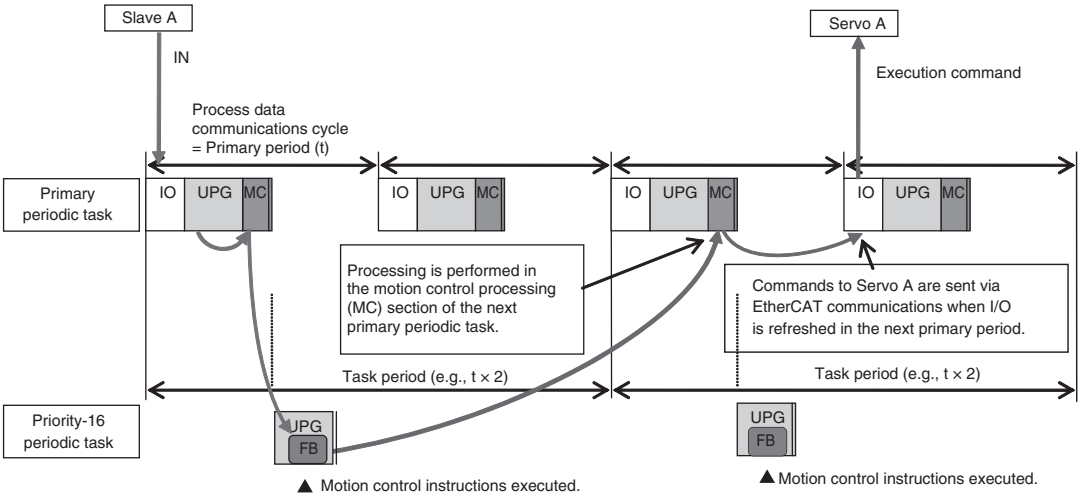
Maximum I/O response time = Priority-5 periodic task period (= process data communications cycle 2) x 2

● 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 EtherCAT slave to which the axis is assigned during the I/O refresh period in the next primary periodic task.

The Controller responds in the following I/O response time regardless of the execution timing of the motion control instructions.

Minimum I/O response time = Priority-16 periodic task period+
Primary period (= process data communications cycle)



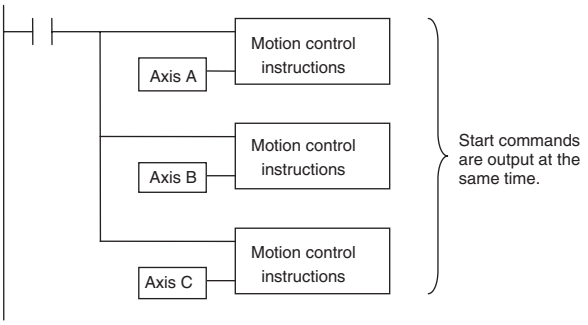
Note: The above diagram shows only one input and one output.

However, the response time may be as follows depending on the timing of the input from the slave.

Maximum I/O response time = Priority-16 periodic task period +
Primary period (= process data communications cycle) × 2

● Simultaneous Execution of More Than One Axis

If more than one axis is controlled in the same task period by the programs in the same task, they can be started at the same time.



Additional Information

You can access the values of Axis Variables in the tasks other than those for axis control.
For detailed usage and precautions, refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)*.

Sequence Control with X Bus Units

For X Bus Units, data exchange with external devices is performed for I/O refreshing in the primary periodic task and priority-16 periodic task.

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

Additional Information

Refer to the *NX-series EtherNet/IP Unit User's Manual (Cat. No. W627)* for information on the I/O response time for the NX-series EtherNet/IP Unit.

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 6-3

6-2 POUs (Program Organization Units) 6-5

6-2-1 What Are POUs? 6-5
6-2-2 Overview of the Three Types of POUs 6-5
6-2-3 Differences between Programs, Functions, and Function Blocks 6-6
6-2-4 Details on Programs 6-7
6-2-5 Details on Function Blocks 6-8
6-2-6 Details on Functions 6-16
6-2-7 Operation That Applies to Both Functions and Function Blocks 6-22
6-2-8 POU Restrictions 6-25

6-3 Variables 6-28

6-3-1 Variables 6-28
6-3-2 Types of Variables 6-28
6-3-3 Types of User-defined Variables in Respect to POUs 6-28
6-3-4 Attributes of Variables 6-30
6-3-5 Data Types 6-32
6-3-6 Derivative Data Types 6-41
6-3-7 Array Specifications and Range Specifications for Data Types 6-51
6-3-8 Variable Attributes 6-60
6-3-9 Changes to Variables for Status Changes 6-69
6-3-10 Function Block Instances 6-83
6-3-11 Monitoring Variable Values 6-83
6-3-12 Restrictions on Variable Names and Other Program-related Names 6-84

6-4 Constants (Literals) 6-86

6-4-1 Constants 6-86
6-4-2 Notation for Different Data Types 6-86

6-5 Programming Languages 6-91

6-5-1 Programming Languages 6-91
6-5-2 Ladder Diagram Language 6-91
6-5-3 Structured Text Language 6-98

6-6 Instructions 6-134

6-6-1 Instructions 6-134
6-6-2 Basic Understanding of Instructions 6-134
6-6-3 Instruction Errors 6-136

6-7 Namespaces 6-142

6-7-1 Namespaces 6-142
6-7-2 Namespace Specifications 6-143
6-7-3 Procedure for Using Namespaces 6-146

6-8 Libraries 6-147
6-8-1 Introduction to Libraries 6-147
6-8-2 Specifications of Libraries 6-147
6-8-3 Library Object Specifications 6-148
6-8-4 Procedure to Use Libraries 6-149

6-9 Programming Precautions 6-151
6-9-1 Array Specifications for Input Variables, Output Variables, In-Out Variables 6-151
6-9-2 Structure Variables for Input Variables, Output Variables, In-Out Variables 6-151
6-9-3 Master Control 6-152

6-1 Overview of Programming Procedures

This section provides an overview of programming procedures.
The shaded steps in the overall procedure that is shown below are related to programming.

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

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

Step 3. Mounting and Setting Hardware

Step 4. Wiring

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

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series* on page 1-23 for detail.

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)* on page 6-5
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* on page 6-91

Variable Design	Reference
Design the user-defined variables that you need to create.	*6-3-1 Variables* on page 6-28 *6-3-2 Types of Variables* on page 6-28
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* on page 6-28
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* on page 3-8
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* on page 6-30 *6-3-5 Data Types* on page 6-32 *6-3-6 Derivative Data Types* on page 6-41
Keep the following precautions in mind when you design variables. • Retention: Set the Retain attributes to determine the values that are used for variables when the power supply is turned ON or when the operating mode changes. • Structures: When a structure is used for a variable in an instruction, design the program to use the same structure data type for the input parameter, output parameter, or in-out parameter. Example: Communications Instructions • Array Specifications: When an array variable is used for the variable for an instruction, design the program to use an array variable for the input parameter, output parameter, or in-out parameter. Examples: Shift Instructions, Stack Instructions, and Table Instructions • AT Specifications: Use AT specifications for the variables used for input parameters to certain instructions. Example: Fixed or user I/O allocations for DeviceNet Units • Network Publishing: Design the variables for EtherNet/IP tag data links.	*6-3-4 Attributes of Variables* on page 6-30 *6-3-5 Data Types* on page 6-32 *6-3-6 Derivative Data Types* on page 6-41

6-2 POUs (Program Organization Units)

The user program that runs on an NJ/NX-seriesCPU Unit is made from a combination of POUs (program organization units).
This section describes the configuration and specifications of POUs.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on creating POUs in the Sysmac Studio.

6-2-1 What Are POUs?

A POU (program organization unit) is a unit that is defined in the IEC 61131-3 user program execution model. A POU includes a local variable table and an algorithm (i.e., a series of code or logic). It is the basic unit used to build the user program.
You combine POUs to build a complete user program.
There are three types of POUs, as described below.

- Programs
  A program corresponds to a main routine. It is the main type of POU that is used for algorithms. You can place any instruction, function, or function block in the algorithm of a program.
- Function Blocks (FBs)
  A function block can output different values even with the same inputs. Function blocks are executed when they are called from a program or another function block.
- Functions (FUNs)
  A function always outputs the same values for the same inputs. Functions are executed when they are called from a program, another function, or a function block.

The POUs consists of a combination of these three types of POUs. You can create many POUs.
You assign the programs to tasks to execute them.

6-2-2 Overview of the Three Types of POUs

Programs

● Executing Programs and Execution Conditions

- You execute a task to execute the programs that are assigned to that task.
- Programs are always executed.

● Notation

- The POUs must include at least one program. You can assign up to 128 programs to a single task.

Function Blocks (FBs)

● Executing Function Blocks and Execution Conditions

- You can call function blocks from programs or other function blocks to execute them.

- Function blocks are always executed.
- If you want a function block to execute only when a condition is met, you must define an input variable that sets the execution condition.

● Notation

- You can use any instruction, user-defined function, or user-defined function block in the algorithm of a function block.
- You can retain the values of internal variables. Therefore, you can retain status, such as for timers and counters.
- There are both user-defined and system-defined functions.
  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* on page 6-8.

Functions

● Executing Function 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 functions.
  User-defined functions are called user-defined functions. System-defined functions are sometimes called FUN instructions.

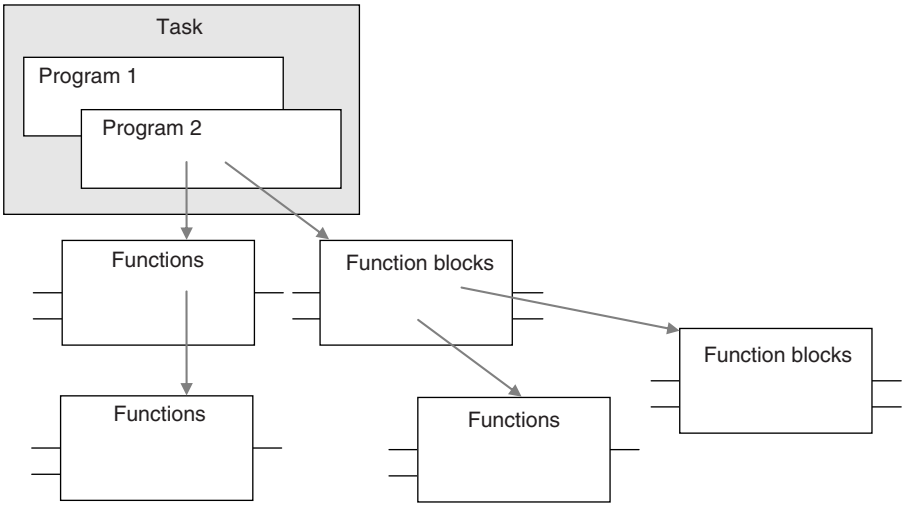
For details on functions, refer to *6-2-6 Details on Functions* on page 6-16.

6-2-3 Differences between Programs, Functions, and Function Blocks

POU type / Item		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.

POU type / Item	Programs	Function blocks	Functions
Execution condition	Executed each period.	Executed each period. Specify the execution condition with an input variable.	Specify the execution condition with the EN input.

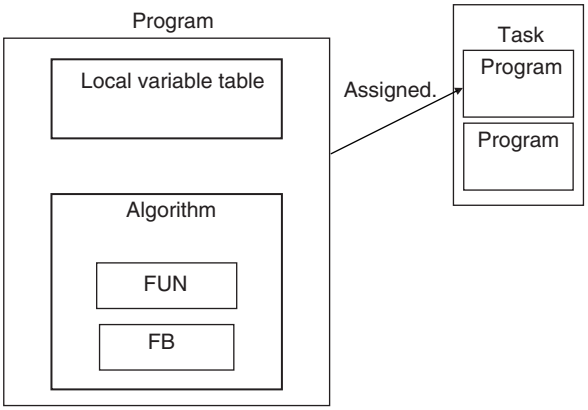
The hierarchical relationships between programs, functions, and function blocks are shown in the following figure.



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 CPU Unit begins operation.	BOOL	Read
P_First_Run*1	First Program Period Flag	This flag is TRUE for one task period after execution of the program starts. *2 Use this flag to perform initial processing when execution of a program starts.	BOOL	Read
P_PRGER	Instruction Error Flag	This flag changes to and remains TRUE when an instruction error occurs in the program or in a function/function block called from the program. After this flag changes to TRUE, it stays TRUE until the user program changes it back to FALSE.	BOOL	Read/ write
P_CY	Carry Flag	This flag is updated by some instructions.	BOOL	Read

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

*2. To enable or disable the program, use the PrgStart or PrgStop instruction. You can make setting for the PrgStart instruction so that it executes the program without changing *P_First_Run* to TRUE.

6-2-5 Details on Function Blocks

Procedure to Create Function Blocks

A function block consists of a function block definition that is made in advance and instances that are used in the actual programs.
Create function blocks in the following order.

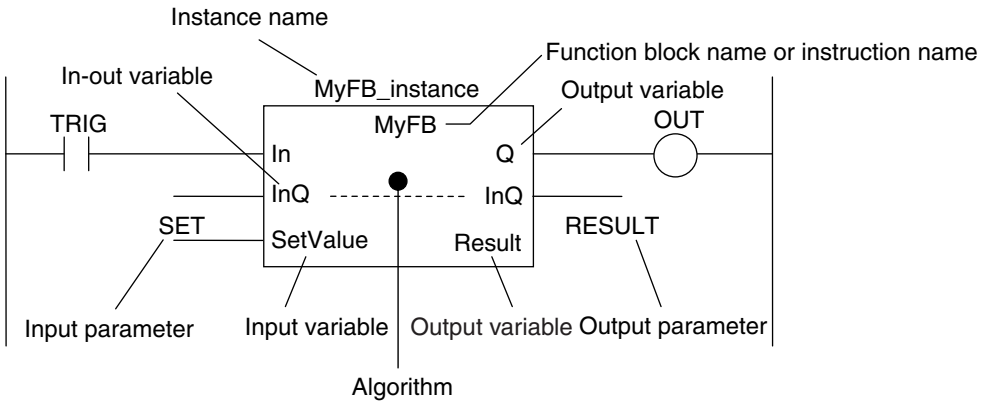
1 Create the function block definition.
Create the algorithm.

2 Place 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.

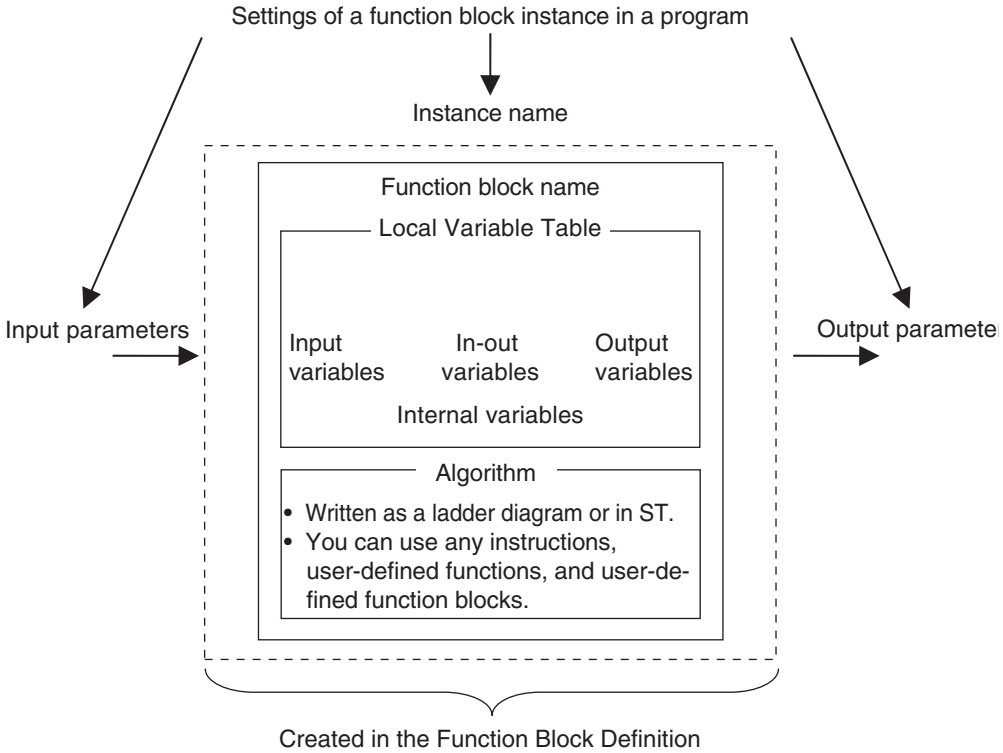
Structure of Function Blocks

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

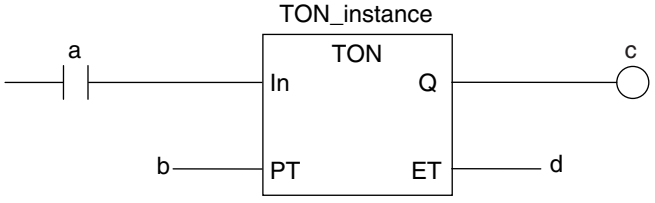
You can omit input and output parameters. Refer to information on operation when parameters are omitted in *Operation When Parameters Are Omitted* on page 6-23 for details.

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.



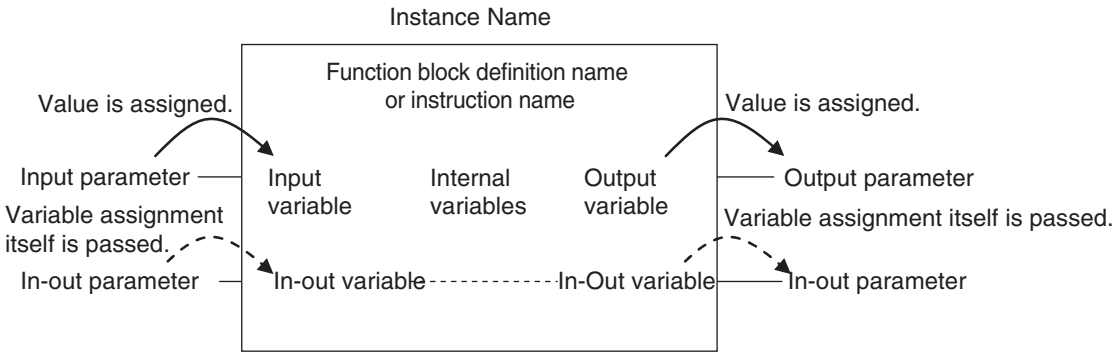
Function Blocks Expressed in ST:

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* on page 6-121 for details.

Variable Designations for Function Blocks



The specifications for variables in function blocks are given below.

Variables	Number	Specification
Input variables[*1]	1 to 64	Input variables are used as input arguments within the function block. They cannot be changed inside the function block.
		• When the function block is executed, the input variables are set to the values of the input parameters. • You can specify either constants or variables for input parameters. • Omitting input parameters Refer to information on operation when parameters are omitted in *Operation When Parameters Are Omitted* on page 6-23. • You can specify to detect when the variable changes to TRUE or changes to FALSE. • You can access and change the values from outside the function block. Access these values using the following format: *InstanceName.InputVariableName*. [*2]
Output variables[*3]	1 to 64	Output variables are used as output arguments from the function block.
		• The output parameters are set to the values of the output variables at the end of function block execution. • You cannot specify a constant or a variable with constant attribute for an output parameter. • You can omit output parameter connections. If you omit an output parameter, the value of the output variable is not assigned to any parameter. • You can access the values of output variables from outside of the function block. Access these values with the following format: *InstanceName.OutputVariableName*. However, you cannot write values directly to an output variable.

Variables	Number	Specification
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.
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 user-defined function blocks. *4 • Refer to *ENO* on page 6-20 for details.

*1. In the Sysmac Studio version 1.01 or lower, at least one BOOL input variable is required when you use function blocks in a ladder diagram or in ST.

*2. In the Sysmac Studio version 1.07 or lower, it is impossible to change the value of an input variable from outside the function block. However, accessing it from outside the function block is possible.

*3. At least one BOOL output variable (including *ENO*) is required when you use function blocks in a ladder diagram.
In the Sysmac Studio version 1.01 or lower, at least one BOOL output variable (including *ENO*) is required when you use function blocks in ST.

*4. When an error occurs in an input parameter, the power flow output changes to FALSE. If you omit ENO for user-defined function blocks, the power flow output remains FALSE from the next time you call the function blocks. The power flow output cannot change to TRUE. When an error may occur in an input parameter, define the ENO and write the algorithm so that the power flow output changes to TRUE when you call function blocks.
Refer to *Operation for Parameter Errors* on page 6-24 for information on the operation for the input parameter error.

Refer to *6-3-4 Attributes of Variables* on page 6-30 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.

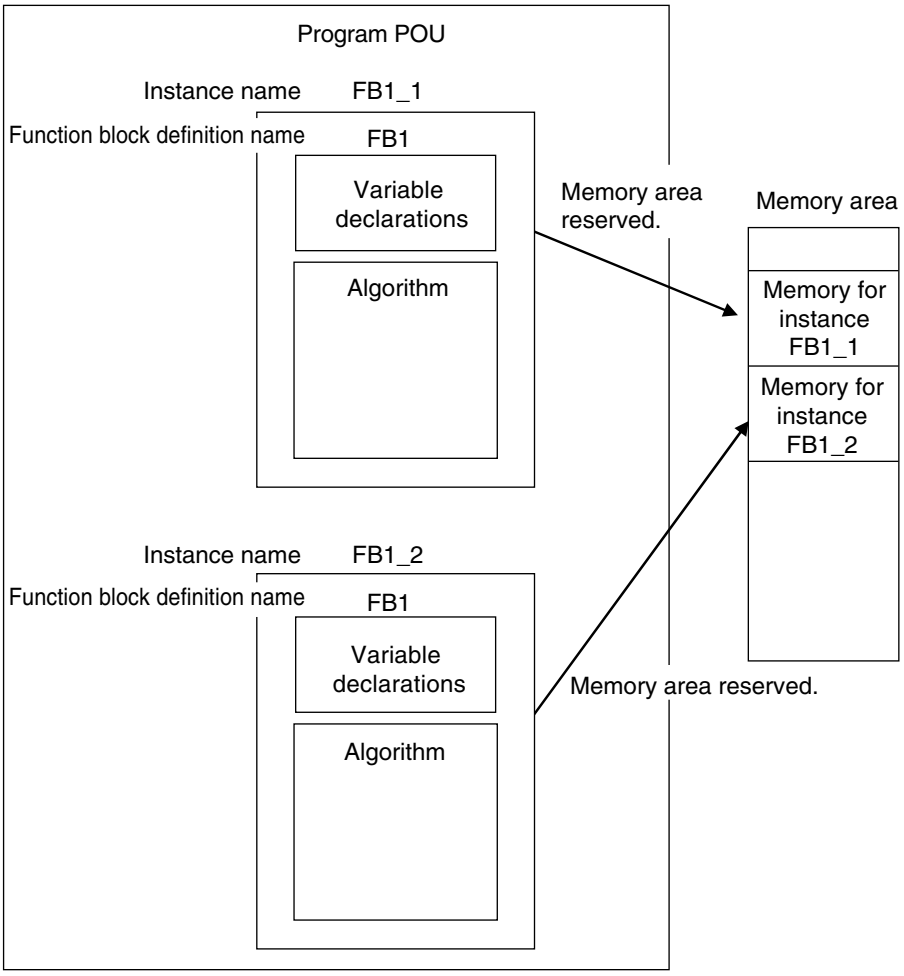
Function Block Definitions and Instances

A function block consists of a function block definition that is made in advance and instances that are 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.



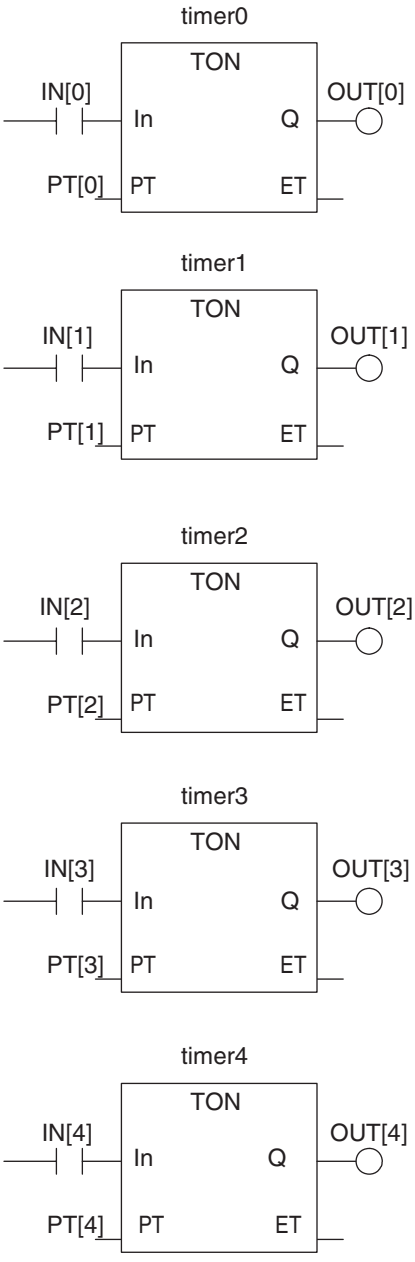
Instances cannot be read from other programs or function blocks. If an instance with the same name as another instance is placed in a different program or another function block, that instance will operate as a completely separate instance.

Array Specifications for Instances

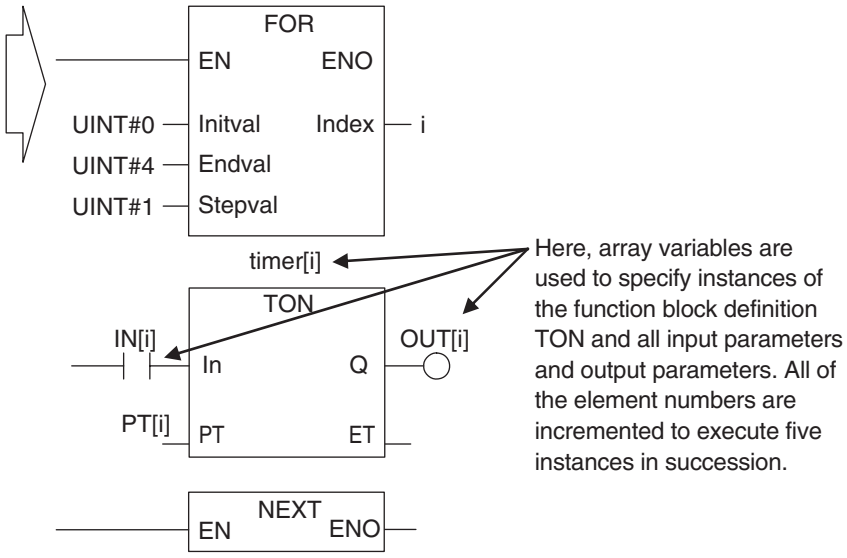
Array specifications can be made for instances.
You can indirectly specify an array element number with a variable to execute multiple instances with one instance name.
Furthermore, you can switch input sources and output destinations and effectively execute multiple instances with a single instance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



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

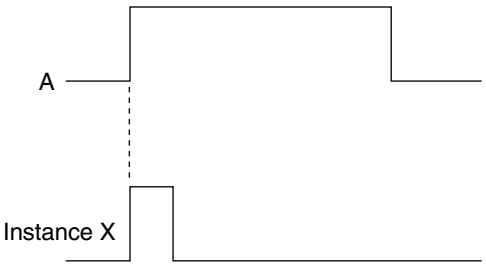
Function blocks do not have an *EN* input like functions. They are executed each period.

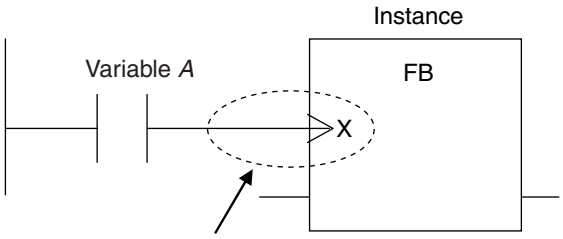
Case	Algorithm in FB		ENO*1	Operations other than ENO
Normal operation	Executed.	Normal end	TRUE	Output parameters: Values are updated according to the internal algorithm. In-out parameters: Values are updated according to the internal algorithm.
		Error end	FALSE	Output parameters: Retained In-out parameters: 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.

*1. Refer to *Variable Designations for Function Blocks* on page 6-11 for details on the ENO specifications of function blocks.

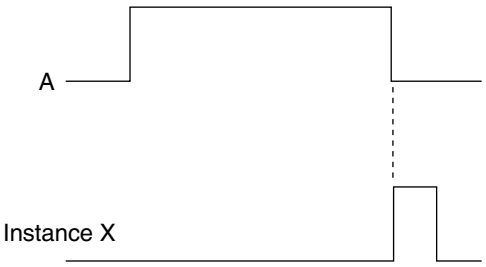
Refer to *6-5-2 Ladder Diagram Language* on page 6-91 for details on power flow output and parameter output.

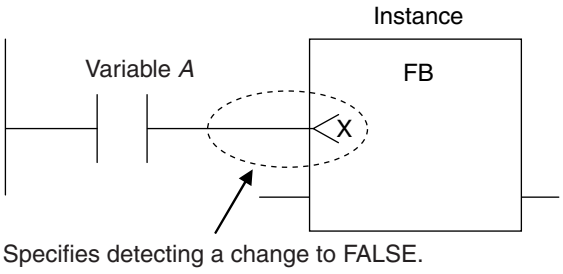
You can specify the edge for an input variable to make the variable TRUE only when the input parameter changes to TRUE.





You can specify falling edges too.





Accessing Variables in a Function Block from Outside the Function Block

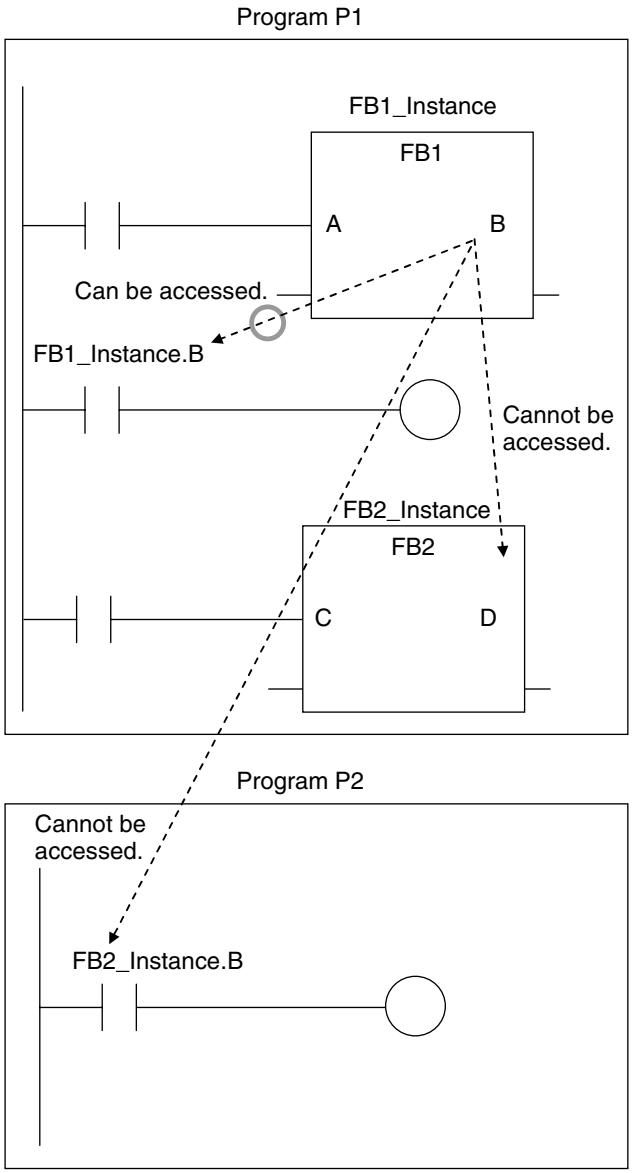
You can access the input and output variables of a function block from outside the function block. Variables are written as follows:

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.

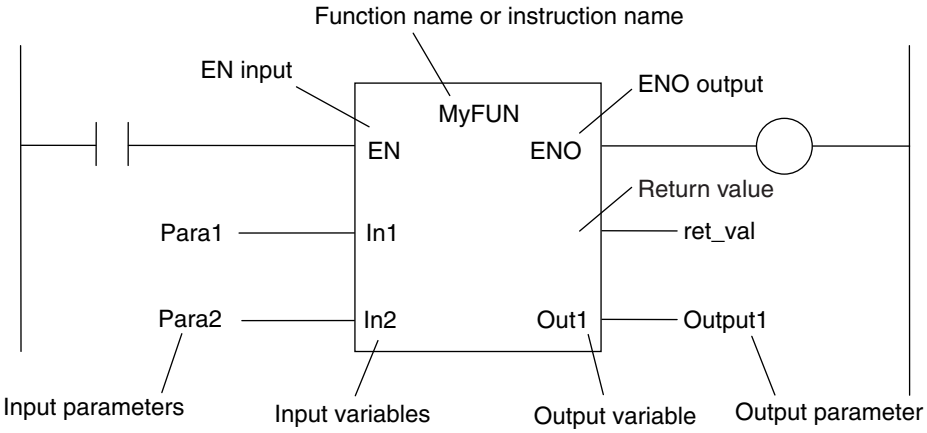
The following variables cannot be accessed from external devices. If these variables are accessed, a building error will occur.
- In-out variables for function blocks
- Input variables for FB instructions for which an initial value is not applied when the input parameter is omitted

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

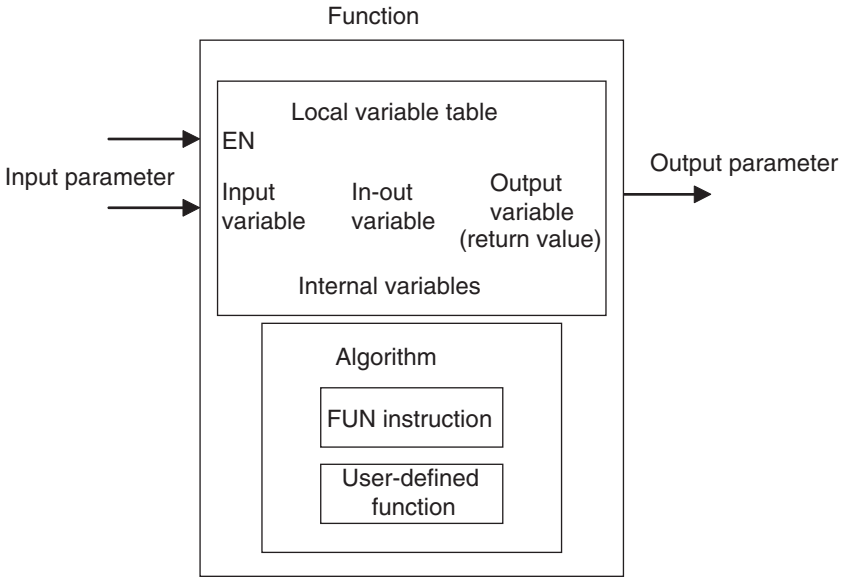
This is the function name or instruction name assigned in the function definition when the function is defined.

● 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.

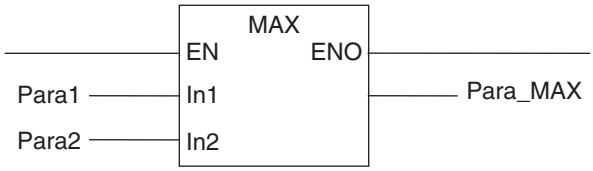
Expressing Functions in ST

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:

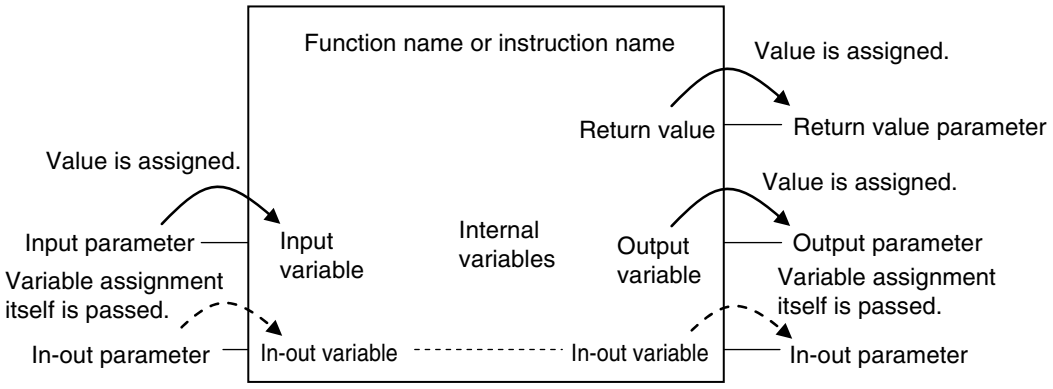


Function name

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

Refer to *Function Calls* on page 6-123 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 undefined.

Variables	Number	Specification
In-out variables	0 to 64	In-out variables are used as inputs to and outputs from the function. They can be changed inside the function.
		• In-out parameters (variable designations) are directly passed to or received from the in-out variables. • You cannot specify a constant or a variable with constant attribute for an in-out parameter. • If you change the value of an in-out variable within a function, the value of the in-out parameter changes at that time. • You cannot omit in-out parameters. • You cannot access the values of in-out variables from outside of the function.
Internal variables	No limit	Internal variables are used for temporary storage within a function.
		• The value is not retained after execution is completed. • You cannot access the values of internal variables from outside of the function.
External variables	No limit	External variables are used to access global variables.
EN	1	This is a BOOL input variable used to execute the function.
		• The function is executed when *EN* is TRUE. The value of *EN* in the function is always 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 undefined. • Refer to *Return Values* on page 6-21 for details.

Refer to *6-3-4 Attributes of Variables* on page 6-30 for details on setting variable attributes.

Additional Information

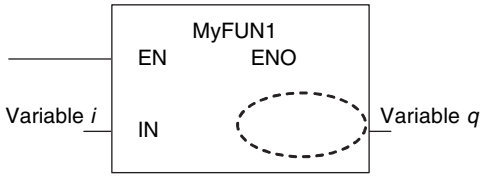
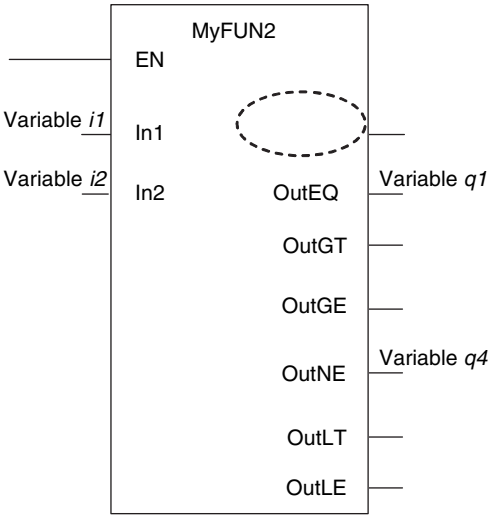
You can register global variables as external variables in a function variable table to access global variables.
We recommend that you create your functions so that they produce output values uniquely based on their input parameter values. Algorithms that access global variables and use them to affect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

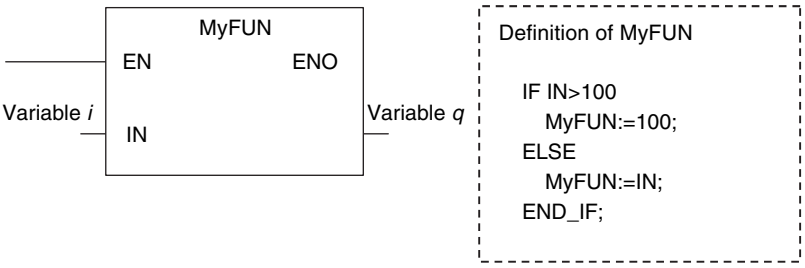
- When *ENO* is FALSE, the previous values of all other output variables are retained.

● 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*.

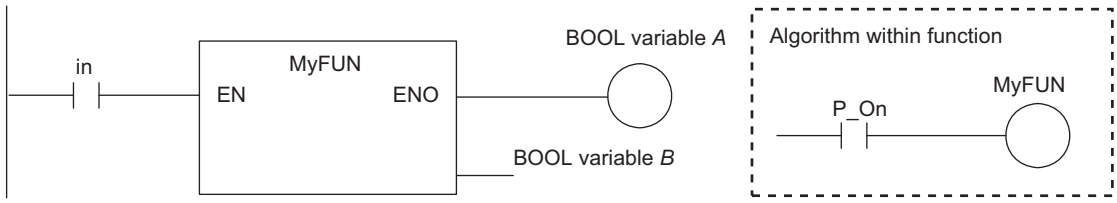


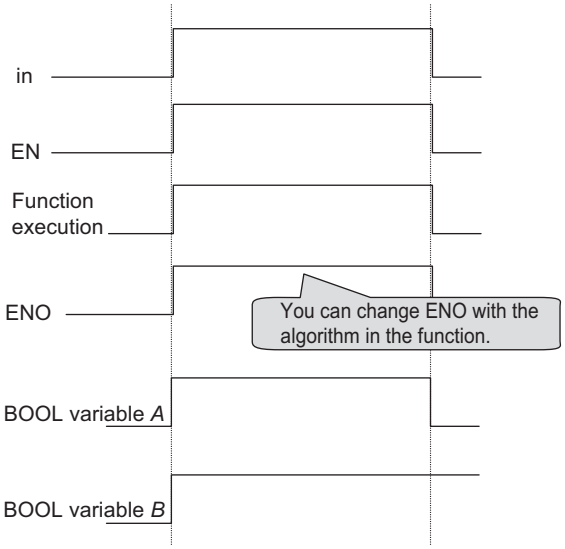
Execution Conditions for Functions

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: 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.

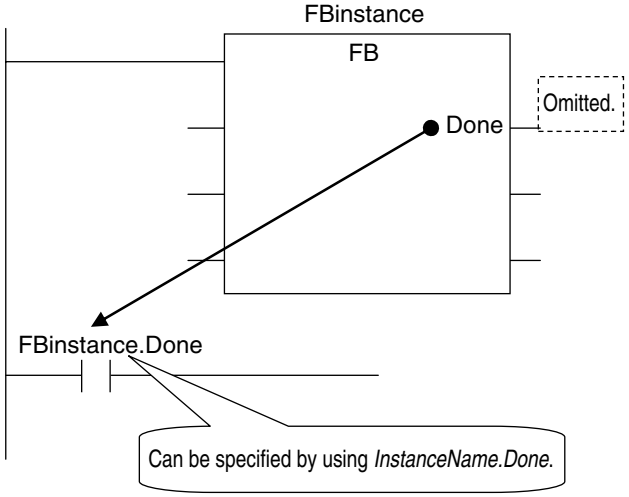
POU		EN	ENO
FB	User-defined function blocks	Cannot be used. A building error occurs if you try to define *EN* in the variable table from the Sysmac Studio.	Can be used or omitted. You define *ENO* as an output variable in the Sysmac Studio.
	Instruction	All FB instructions do not use *EN*.	Some instructions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

*1. 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.



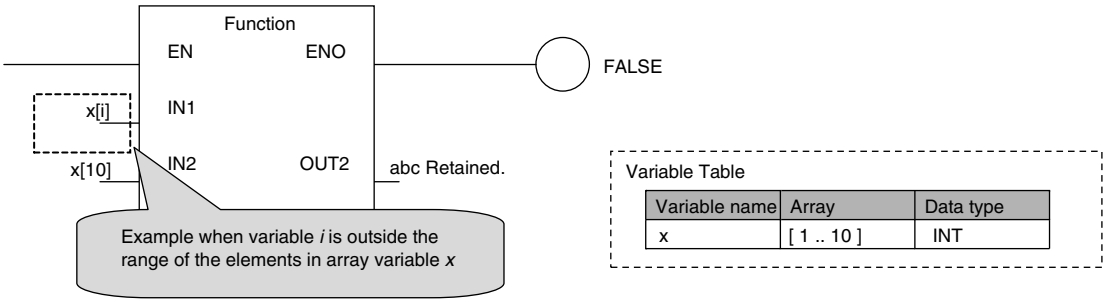
Operation for Parameter Errors

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:



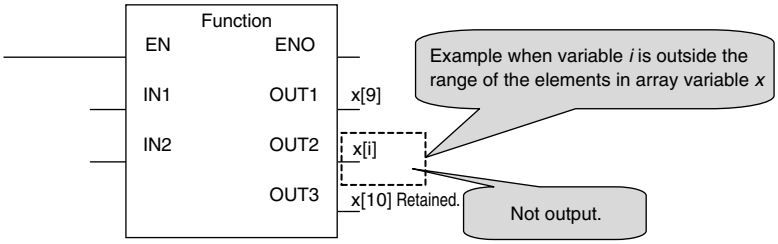
Precautions for Correct Use

When an error occurs in an input parameter, the power flow output changes to FALSE. If you omit ENO for user-defined function blocks, the power flow output remains FALSE from the next time you call the function blocks. The power flow output cannot change to TRUE. When an error may occur in an input parameter, define the ENO and write the algorithm so that the power flow output changes to TRUE when you call function blocks.

● Errors in Output Parameters

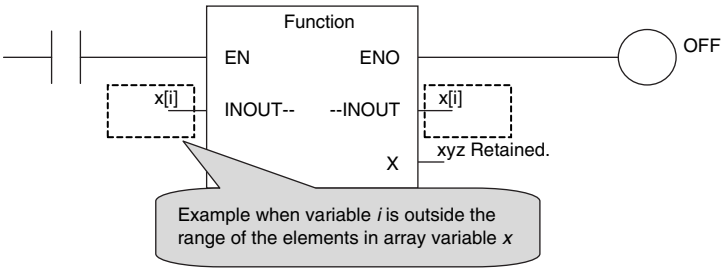
If an error is detected in an output parameter, all values after that parameter are not output but their values are retained.

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* on page 6-84 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.
In this case, it is better to use an in-out variable than an input variable to reduce the processing time. However, be aware that if you use an in-out variable, the data passed to the function block or function as a parameter is written and the original data is not retained.

Additional Information

Specifying an Array Variable or Structure Variable as a Parameter

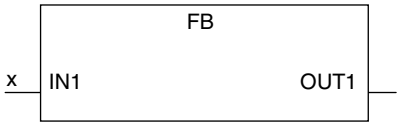
You can also specify an array variable or a structure variable as an input or output parameter. However, it will take longer to pass and receive data for these data types in comparison to a variable with a basic data type (depending on the size).

Therefore, when handling array variables or structure variables in a function block, we recommend that you design them in such a way that these variables are passed to and received from in-out variables.

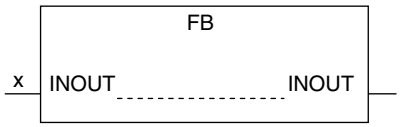
Example 1: Specifying an Array

Program Variable Table

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



Do not assign INT variable *x [0..199]* to the function block input variable *IN1* (Data type of the *IN1* input variable in the function block: ARRAY[0..199] OF INT).

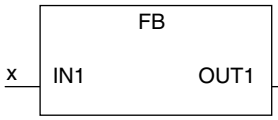


Instead, assign INT variable *x [0..199]* to the *INOUT* in-out variable of the function block. (Data type of INOUT in-out variable in the function block: ARRAY[0..199] OF INT)

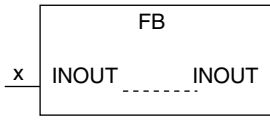
Example 2: Specifying a Structure Variable

Program Variable Table

Variable name	Data type
x	MyStructure



Do not assign MyStructure variable x to the *IN1* input variable of the function block (Data type of *IN1* input variable of the function block: MyStructure).



Instead, assign MyStructure variable x to the *INOUT* in-out variable of the function block. (Data type of *INOUT* in-out variable of the function block: MyStructure)

Nesting Levels

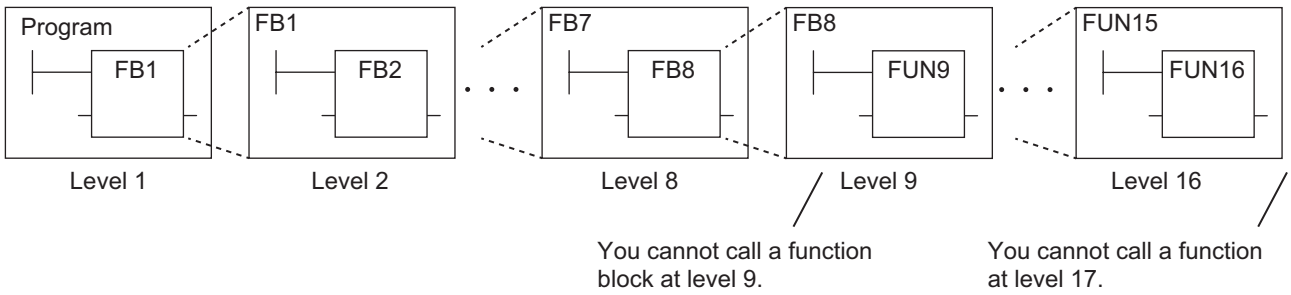
Calling another function or function block from a function or function block that was called from a program is called nesting. The limits that are given in the following table apply to the POUs that you can call from a user-defined function or function block and the number of nesting levels. A building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

POU	Called POUs	Nesting depth
Function blocks	Functions and function blocks	8 levels max.

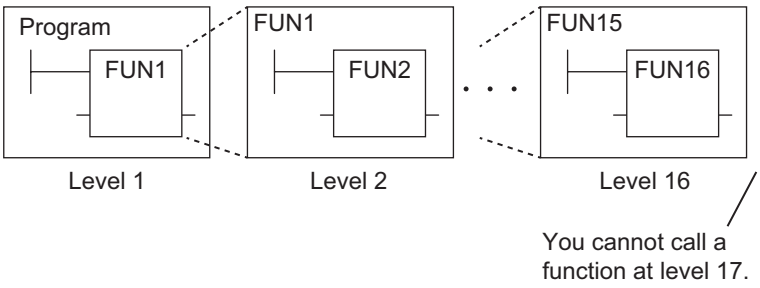
POU	Called POUs	Nesting depth
Functions	Functions	16 levels max.*1

*1. A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required. For other versions, the limit is 8 levels.

Example 1: From a program, you can call function blocks to a depth of 8 levels. You can then call functions to a depth of 16 levels.



Example 2: From a program, you can call functions to a depth of 16 levels.



6-3 Variables

In the NJ/NX-series System, variables are used to exchange I/O information with external devices, to perform data calculations, and to perform other processes.
This section describes variable designations in detail.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on setting variables with the Sysmac Studio.

6-3-1 Variables

Variables store I/O data for exchange with external devices or temporary data that is used for internal POU processing. In other words, a variable is a container for data with a name, data type, and other attributes.

You do not need to assign a memory address to a variable. However, you can assign a specific memory address if necessary (see notes). The NJ/NX-series CPU Unit automatically allocates memory addresses in the memory area for variables.

*1. This is done to use specific functions for some CJ-series Special Units or perform the serial communications using the host link (FINS) protocol. You must specify the CJ-series Unit memory address in the AT Specification attribute of the variable. Refer to *AT Specification* on page 6-61 for details.
*2. You can use CJ-series Units only with NJ-series CPU Units.

6-3-2 Types of Variables

Variables are broadly classified into the following three types.

● User-defined Variables

The user defines all of the attributes of a user-defined variable.
The rest of this section describes user-defined variables.

● Semi-user-defined Variables

These variables are used to access specific devices and data.
There are two types of semi-user-defined variables: device variables and cam data variables.
Refer to *2-3-1 Type of Variables* on page 2-13 and *3-3-1 I/O Ports* on page 3-8 for details on device variables.

● System-defined Variables

System-defined variables are provided in advance in an NJ/NX-series CPU Unit. The names and all attributes are defined by the system. They have specific functions. System-defined variables are supplied for each function module. Refer to *A-7 System-defined Variables* on page A-91 for details.

Refer to *2-3-1 Type of Variables* on page 2-13 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.

○: Supported.

Type of user-defined variable		POU type		
		Programs	FB	FUN
Local variables	Internal variables	○	○	○
	Input variables	Not supported.	○	○
	Output variables	Not supported.	○	○
	In-out variables	Not supported.	○	○
Global variables		○ (see note)	○ (see note)	○ (see note)
External variables		○	○	○

*1. 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

An internal variable can be used only within one POU.
An internal variable is declared in the local variable table of 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 values of the input parameters are assigned to the input variables from the calling POU. An input variable is declared in the local variable table of the POU.

● Output Variables

Before processing a POU is completed, the output parameters returned to the calling POU are assigned to the output variables. An output variable is declared in the local variable table of the POU.

● In-Out Variables

When a POU is called, the in-out variables are assigned to the in-out parameters themselves (variable designations) from the calling POU. If you change the value of an in-out variable within a POU, the value of the in-out parameter changes at that time.
An in-out variable is declared in the local variable table of the POU.

● External Variables

External variables are used to access data outside of a POU.
You can access global variables from POUs.

Global Variables

A global variable is declared in the global variable table.

Device variables that are automatically generated from the Unit configuration and slave configuration and axis/axes group variables that are generated from the Axis Setting Table are automatically registered as global variables.

6-3-4 Attributes of Variables

You can set the following attributes for variables.

Variable Attributes According to Variable Type

● Attributes of Variables

Attribute	Description	Specification	Default
Variable name	The variable name is used to identify the variable.	---	---
Data Type	The data type defines the format of the data that is stored in the variable.	---	BOOL
AT Specification	If you want to handle a specific address for a CJ-series Unit as a variable, specify the address to assign to that variable. *1	• Not specified. • Specify.	Not specified.
Retain	Specify whether to retain the value of the variable in the following cases. • When power is turned ON after a power interruption • When the CPU Unit changes to RUN mode • When a major fault level Controller error occurs	• Retain: Value specified on the left is retained if there is a Battery. • Non-retain: Changes to initial value.	Non-retain: Reset to initial value.
Initial Value	You can select to set or not set an initial value. If the initial value is set, specify the value of the variable in the following cases and do not specify the Retain attribute. • When power is 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 Value* on page 6-64.)
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.	---

Attribute	Description	Specification	Default
Network Publish	This attribute allows you to use CIP communications and data links to read/write variables from outside of the Controller.	• Do not publish • Publish Only • Input • Output	Do not publish
Edge	An Edge attribute allows you to detect when the input parameter of a function block changes to TRUE or changes to FALSE. This can be used only on BOOL input variables.	• None • Change to TRUE • Change to FALSE	None

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

Additional Information

Exclusive Control between Tasks
You can restrict writing to global variables to a single task to prevent changes to the values of global variables during processing. Specify this as a task setting, not as a variable attribute.

● Attributes Supported by Each Type of Variable

Type of variable		Variable Name	Data Type	AT Specification	Retain	Initial Value	Constant	Network Publish	Edge
Global variables		Supported.	Supported.	Supported.	Supported.	Supported.	Supported.	Supported.	Not supported.
Programs	**Internal variables**	Supported.	Supported.	Supported.	Supported.	Supported.	Supported.	Not supported.	Not supported.
	External variables	Not supported.	Not supported.	Not supported.	Not supported.	Not supported.	Supported.	Not supported.	Not supported.
Function blocks	**Internal variables**	Supported.	Supported.	Supported.	Supported.	Supported.	Supported.	Not supported.	Not supported.
	Input variables	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.

Type of variable		Variable Name	Data Type	AT Specification	Retain	Initial Value	Constant	Network Publish	Edge
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 variables	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).
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	TRUE or FALSE	BOOL#0 or BOOL#1 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[*1]	**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 bytes	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 +∞/-∞	

Classification	Data type	Data size	Alignment	Range of values	Notation
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
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 *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for detailed instruction specifications.

*2. Variables are compared with nanosecond precision for comparison instructions. To change the precision for comparison, use the TruncTime, TruncDt, or TruncTod instruction. Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for detailed instruction specifications.

*3. A NULL character (1 byte) is added to the end of text strings. Therefore, reserve memory for one more character than the number of handled characters. For example, if a maximum of 10 single-byte characters are handled, define a STRING variable for 11 characters (11 bytes). STRING[11]

*4. If you want to insert tabs, line break codes, or other special characters, you can use a single-byte dollar sign ($) as an escape character before them. Refer to *Text Strings* on page 6-89 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.
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-11 Variable Memory Allocation Methods* on page A-252 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* on page 6-41 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* on page 6-51 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	

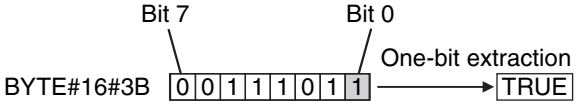
Bit String, Real Number, and Text String Data Formats

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

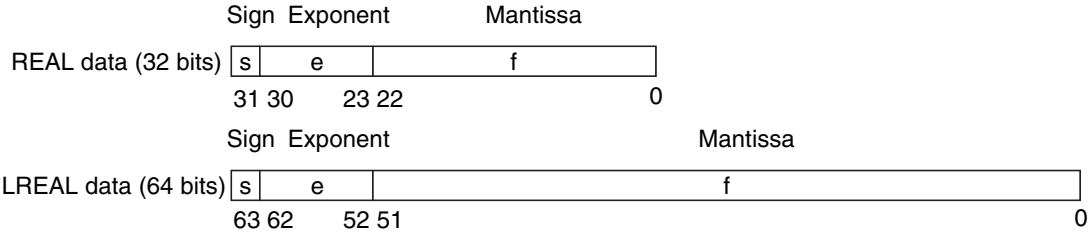
The floating-point decimal format is a way to express a real number as a combination of a sign, an exponent, and a mantissa.
To express a real number as shown below, the value of s is the sign, the value of e is the exponent, and the value of f is the mantissa.

- REAL Data

  Number = $(-1)^s 2^{e-127}(1 + f \times 2^{-23})$
- LREAL Data

  Number = $(-1)^s 2^{e-1023}(1 + f \times 2^{-52})$

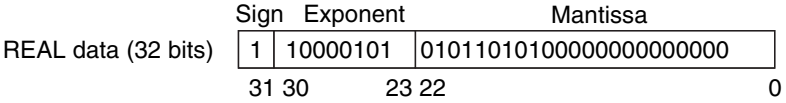
This floating-point decimal format follows the IEEE 754 standard. The formats are given below.



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.

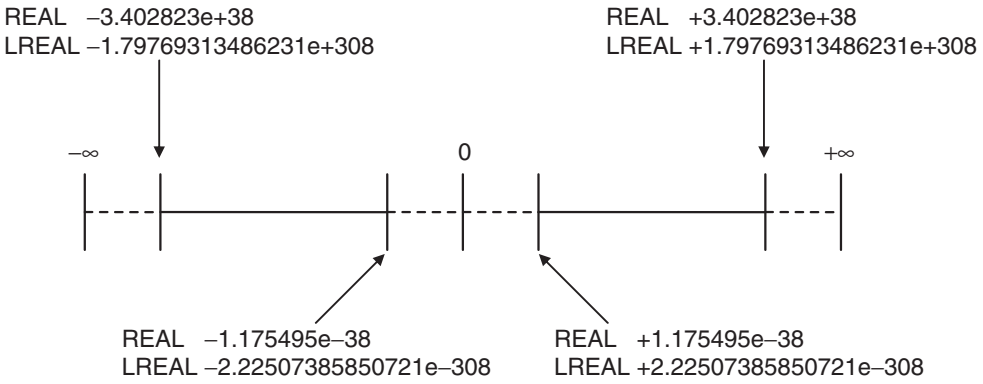


Valid Ranges

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	+∞

Data type	-∞	Negative numbers	0	Positive numbers	+∞
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. Details are provided below.

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 decimal 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 = $(-1)^{s}2^{-126}(f \times 2^{-23})$

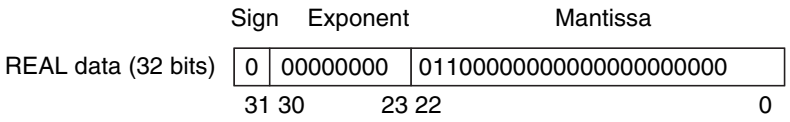
- LREAL Data

  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 decimal 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 decimal 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 CPU Unit.

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 erro
r
                                                    // of 0.000001 is used.
```

Overflows and Underflows

When the true absolute value exceeds the values that can be expressed in the floating-point decimal format, it is called an overflow.
On the other hand, if the value is smaller than the values that can be expressed in the floating-point decimal format, it is called an underflow.

- 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).

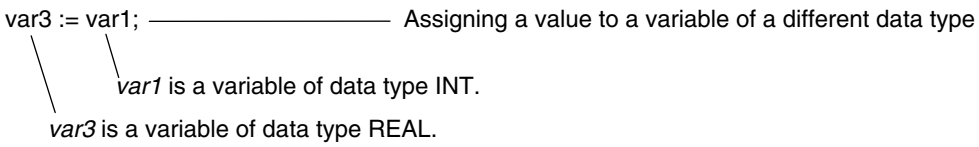
Converting Data Types

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 *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for details.

6-3-6 Derivative Data Types

A derivative data type has a configuration that is based on one of the basic data types.
The following is a list of the derivative data types.

- Structures
- Unions
- Enumerations

Refer to *6-3-12 Restrictions on Variable Names and Other Program-related Names* on page 6-84 for restrictions on the number of characters in data type names and other restrictions when you create a derivative data type.

Additional Information

NJ/NX-series Controllers come with three different types of system-defined derivative data types.

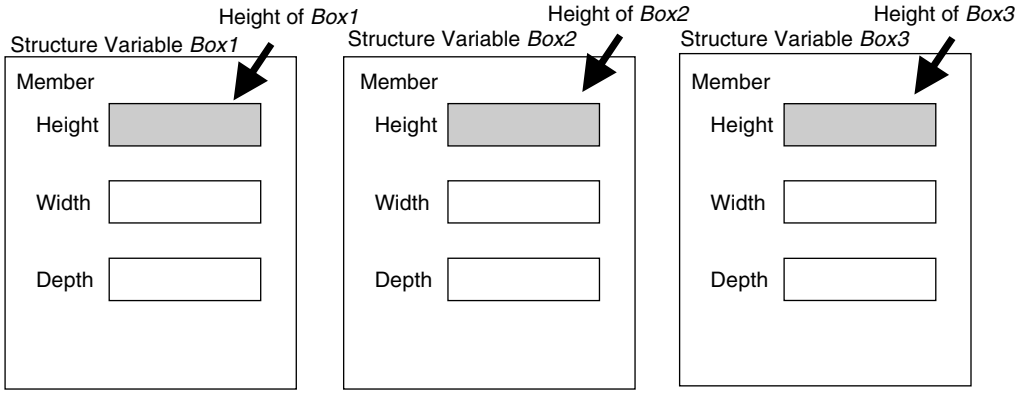
- System-defined variables that are structures
- Structures used for input, output, and in-out variables for instructions
- Structures for Special Unit expansion memory (You must register these in the Unit Editor to use them.)

Structures

A structure is a derivative data type that groups together data with the same or different variable types. You can easily change data and add new data if you place your data into a structure.

For example, you can define a “Box” structure that has three members (Width, Height, and Depth) in order to organize and group your data.
You can then use this structure data type to add a variable called *Box1*. You can then use it to access the different levels of the data by placing a period after the variable name followed by the name of the data you want to access. For example, *Box1.Width* or *Box1.Height*.

If you need to create a new variable to store more box data, you can perform the same steps to add a new variable called *Box2* to the variable table.



When a structure is used for a variable in an instruction, it is necessary to select a structure for the input parameter, output parameter, or in-out parameter, and register the variable.
Example: Communications Instructions

● Expressing Structure Variables and Structure Variable Members

Specifying Members

The individual pieces of data that make up a structure are called “members.”
You can express individual members of a structure by putting a period after the variable name that represents the entire structure followed by the member name that you want to access.
You can even have a structure that is the member of another structure.

Example:
abc.x: Member *x* of structure variable *abc*
abc.Order.z: Memober *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.

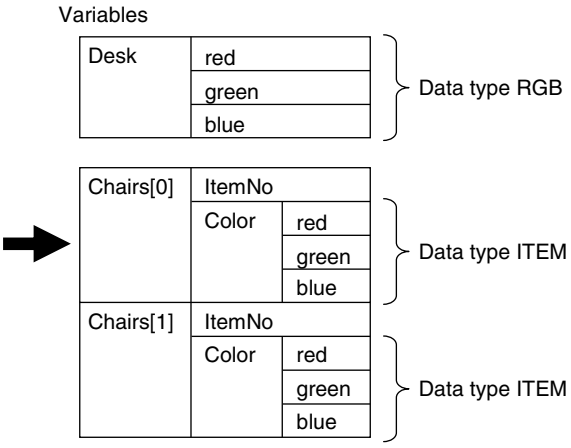
Data Type Table

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	8 levels max. (however, a member name must be 511 bytes or less, including the variable name)
Maximum size of one structure variable	NX701-□□□□: 8 MB NX502-□□□□: 8 MB NX102-□□□□: 4 MB NX1P2-□□□□: 1 MB NJ501-□□□□: 4 MB NJ301-□□□□: 2 MB NJ101-□□□□: 2 MB

Data Type 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 *1. Recursions and loops are not allowed. (An error will occur when the program is checked.)	Supported
	Array specifications for structures, unions, and enumerations	Supported
POU instances		Not supported.

● Arrays and Structures

You can set an array in which the elements are structures. You can also set a structure in which the members are arrays.

● Specifying Structure Member Offsets

When you specify an offset for a member, you can set the memory configuration of the members as required for each structure data type. This allows you to align the memory configuration of the members of the structure data type when you use tag data links with CJ-series CPU Units or with other external devices.

You can select *NJ*, *CJ*, or *User* as the offset type for structure members. If you select *NJ*, the memory configuration that is optimum for the NJ/NX-series Controllers is automatically used. Refer to *A-11 Variable Memory Allocation Methods* on page A-252 for details on the memory configuration of the NJ/NX-series Controllers. Refer to *A-11-2 Important Case Examples* on page A-261 for examples of tag data links with CJ-series CPU Units.

The meanings of the offset type are as follows:

Offset type	Meaning
NJ	The memory configuration that is optimum for the NJ/NX-series Controllers is automatically used and operating speed is maximized.
CJ	The memory configuration for CJ-series PLCs is automatically used. This allows you to use the same memory configuration as a CJ-series CPU Unit.
User	You can set the memory offsets for each member. This allows you to use the same memory configuration as external devices other than CJ-series CPU Units.

Version Information

The following table gives the unit version of the CPU Units and the Sysmac Studio version that are required to specify member offsets.

Unit version of CPU Unit	Sysmac Studio version		
	1.01 or lower	1.02	1.03 or higher
1.01 or later	Not possible.	Possible.*1	Possible.
1.00	Not possible.	Not possible.	Not possible.

*1. You cannot select the memory offset type. You can set member offsets.

Setting Offsets

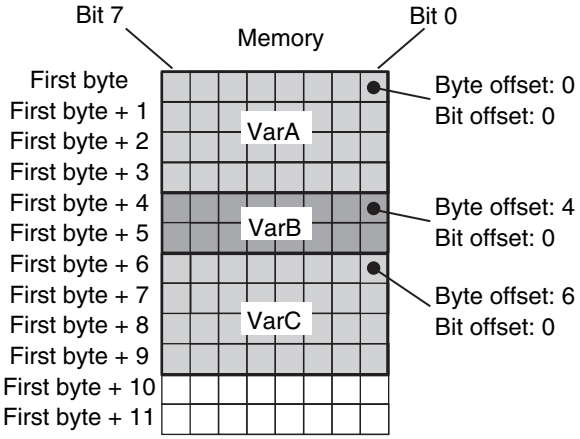
If you set the memory offset type to *User*, you can set memory offsets for each member of the structure. There are byte offsets and bit offsets. If you set the memory offset type to *NJ* or *CJ*, the memory configuration is determined automatically. You do not need to set offsets.
The meanings of the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

Offset	Meaning	Unit	Range of values
Byte offset	The byte offset is the offset of a member from the start of the structure. Bytes offsets are used for all basic data types and derivative data types.	Byte	*1
Bit offset	The bit offset is the offset of a member from the start of the byte position that is specified with the byte offset.	Bit	0 to 63

*1. For NX-series CPU Units, the range of values is 0 to 8191. For NJ-series CPU Units, the range of values is 0 to 1023.

Example:
This example shows the memory configuration when the following settings are made with the Structure Editor.

Name	Data type	Offset type	Byte offset	Bit offset
StrA	STRUCT	User		
VarA	DINT		0	0
VarB	INT		4	0
VarC	DINT		6	0



Offsets That You Can Set

Even if you set the memory offset type to *User*, the offsets cannot be changed for some data types. The following table shows when offsets can be set.

Classification	Data Type	Byte offsets	Bit offsets
Boolean	**BOOL**	Can be set.	Can be set.
Bit strings	**BYTE, WORD, DWORD, LWORD**	Can be set.	Fixed.
Integers	**SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT**	Can be set.	Fixed.

Classification	Data Type	Byte offsets	Bit offsets
Real numbers	**REAL, LREAL**	Can be set.	Fixed.
Durations	**TIME**	Can be set.	Fixed.
Date	**DATE**	Can be set.	Fixed.
Time of day	**TIME_OF_DAY**	Can be set.	Fixed.
Date and time	**DATE_AND_TIME**	Can be set.	Fixed.
Text strings	**STRING**	Can be set.	Fixed.
Array specifications		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 when you specify 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 6-66 for details.
- 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 values of the offsets to suitable values.
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

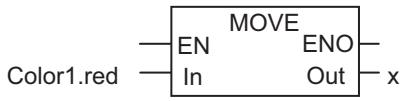
Data Type Table

Name	Member	Data type
RGB	red	UINT
	green	UINT
	blue	UINT

Variable Table

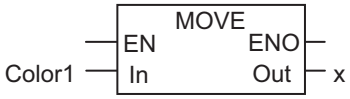
Variable name	Data type
Color1	RGB

■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.

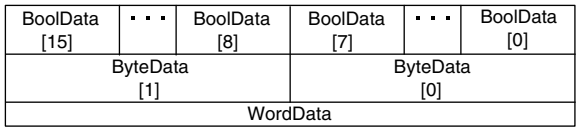
2 Specify the union data type from above as the data type and register the variable in the variable table.

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




```
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 specifications 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

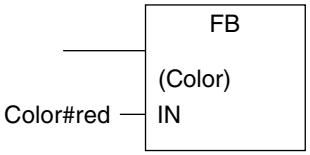
1 Create an enumeration data type in the Enumeration Table.
Set the enumerators and their values for the enumeration.

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* is registered with a data type of Color. 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. Prohibited characters are the same as for variable names. 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 building 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

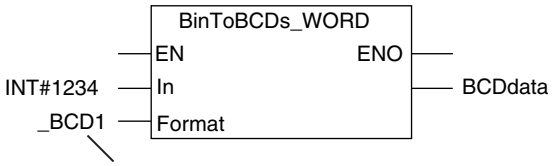
Notation to Use an Enumerator as a Function Block or Function Parameter

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

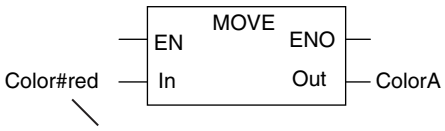


The *Format* input variable to the BinToBCDs_WORD is specified as an enumerator.
Therefore, it is necessary to specify only the enumerator.

***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
  - Fixed-length array specifications
  - Variable-length 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).

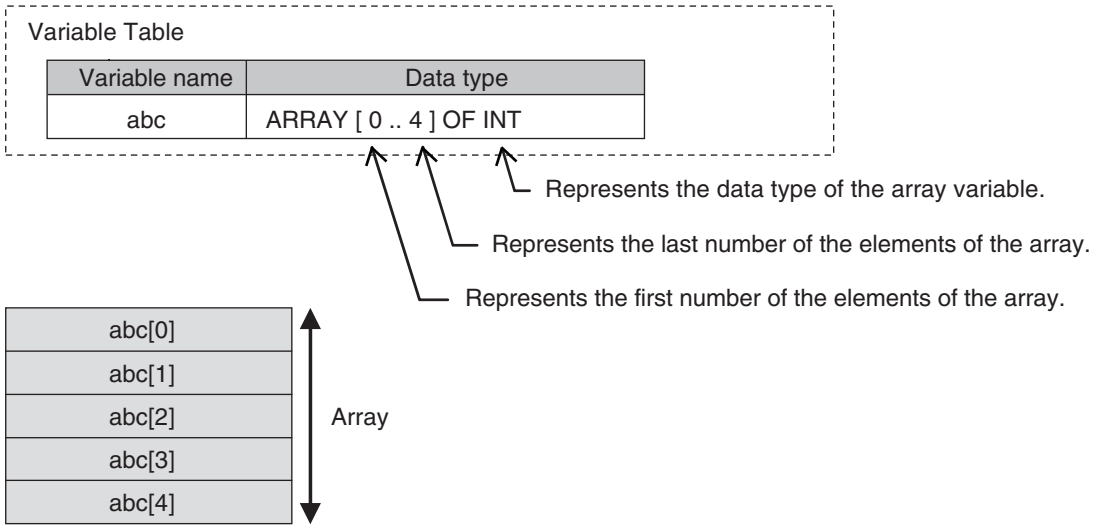
Specifying An Array (i.e., the Entire Array)

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.



Additional Information

You can use a fixed-length array specification for a variable when you specify the first number and last number for the subscripts. You can use a variable-length array specification for a variable when you specify a single-byte asterisk (*) for the subscript.
There are the restrictions on fixed-length array specifications when you use a variable-length array specification. Refer to *Variable-length Array Specifications* on page 6-56 for details on variable-length array specifications.

Version Information

A CPU Unit with unit version 1.18 or later and Sysmac Studio version 1.22 or higher are required to use variable-length array specification.

● 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:*1 Arithmetic expressions: Arithmetic expressions can be specified only in ST. Example: y:= x[a+b];
Maximum size of one array variable	NX701-□□□□: 8 MB NX502-□□□□: 8 MB NX102-□□□□: 4 MB NX1P2-□□□□: 1 MB NJ501-□□□□: 4 MB NJ301-□□□□: 2 MB NJ101-□□□□: 2 MB

*1.

Classification	Data type		Usage
Basic data types	Integer	SINT, INT, DINT, USINT, UINT, or UDINT	Possoble.
		LINT or ULINT	Not possible.
	Boolean, bit string, real, duration, date, time of day, date and time, or text string data		Not possible.
Derivative data types	Structure data types, union data types, and enumeration data types		Not possible.
POU instances			Not possible.

● Dimensions of Array Variables

You can regard the elements of a one-dimensional array as one-dimensional data lined up in a single row.

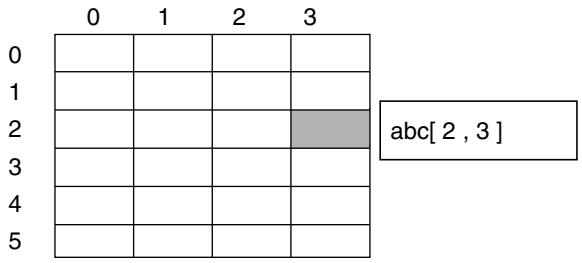
You can set two-dimensional and three-dimensional arrays in the same way.

The array elements are expressed by adding the same number of subscripts to the array variable name as the number of dimensions. Arrays can have a maximum of three dimensions.

Two-dimensional Array Specifications

Variable Table

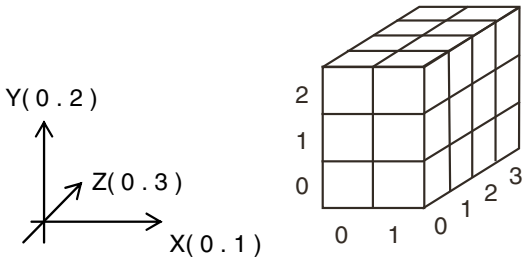
Variable name	Data type
abc	ARRAY [0..5 , 0..3] OF INT



Three-dimensional Array Specifications

Variable Table

Variable name	Data type
ITEM	ARRAY [0..1, 0..2, 0..3] OF INT



● Arrays and Structures

You can set an array in which the elements are structures.

You can also set a structure in which the members are arrays.

Arrays with Structure Elements

Date Type Table

Data type	Member	Member
Str	x y	INT DINT

Variable Table

Variable name	Data type
abc	ARRAY [1..4] OF Str

Variables

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

Structure with Array Members

Data Type Table

Data type	Member	Data type
Str	x	ARRAY [0..1] OF INT
	y	DINT

Variable Table

Variable name	Data type
abc	ARRAY [1..4] OF Str

Variables

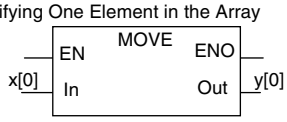
abc[1].x[0]
abc[1].x[1]
abc[1].y
:
abc[4].x[0]
abc[4].x[1]
abc[4].y

● Instructions with an Array Parameter

Some instructions pass array variables as parameters. To do so, specify only the name of the array variable as the input parameter.

Example: Passing a Single Array Element to the MOVE Instruction and Passing an Array to the MOVE instruction

Specifying One Element in the Array

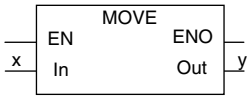


Moves INT variable *x[0]* to *y[0]*

Variable Table

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

Specifying the Array

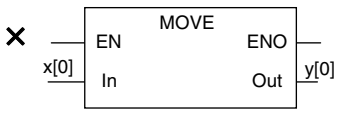


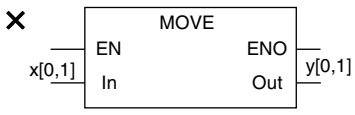
Moves INT variable *x[0..199]* to *y[0..199]*

Restrictions:
When you specify an array variable, it must be moved to a variable of the same data type with the same range of element numbers.

Additional Information

You cannot specify part of a multi-dimensional array as a parameter.





Variable Table

Variable name	Data type
x	ARRAY [0..9 , 0..9 , 0..9]OF INT
y	ARRAY [0..9 , 0..9 , 0..9]OF INT

● Array Protection

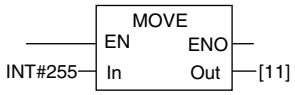
The following errors occur if you attempt to access an element that exceeds the number of elements in an array.

When the Subscript Is a Constant

An error is displayed when you input the variable or when you check the program on the Sysmac Studio.

Variable Table

Variable name	Data type
x	ARRAY [1..10] OF INT

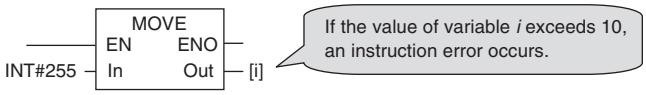


When the Subscript Is a Variable

When an output parameter is assigned to an output variable, the CPU Unit checks to see if the number of elements was exceeded after it executes the instruction. When a subscript variable exceeds the range of the elements of the array variable, an instruction error occurs. Even if this error occurs, the value of ENO will be TRUE because internal processing of the instruction ends normally.

Variable Table

Variable name	Data type
x	ARRAY [1..10] OF INT



Variable-length Array Specifications

You can use a variable-length array variable if you specify a single-byte asterisk (*) for the subscript when you declare the array variable. The maximum number of elements, dimensions, and data types that can be declared are the same as the specification of fixed-length array variables.
However, there are the restrictions on variable declaration and processing.

● Restrictions on Variable Declaration

Only in-out variables for functions (FUN) and in-out variables for function blocks (FB) can be declared as variable-length array variables. Other variables cannot be declared as variable-length array variables.

Array variable to declare	Allowed or not allowed
In-out variables for functions	Allowed.
In-out variables for function blocks	Allowed.
Input variables or output variables for functions	Not allowed.
Input variables or output variables for function blocks	Not allowed.
Internal variables, external variables, and global variables	Not allowed.
Members in structure variables or members in union variables	Not allowed.

When you declare an array variable with two or three dimensions as a variable-length array variable, you need to specify a single-byte asterisk (*) for all dimensions. You cannot declare a variable which uses both variable-length dimension and fixed-length dimension.

Array variable to declare	Specification example	Allowed or not allowed
Variable-length array specification for all dimensions	• ARRAY[*] • ARRAY[*, *] • ARRAY[*, *, *]	Allowed.
Both variable-length dimension and fixedlength dimension are used	• ARRAY[*, 3..5] ARRAY[*, *, 3..5]	Not allowed.
Individual specification for first number or last number	• ARRAY[1..*] • ARRAY[*..*]	Not allowed.

● Restrictions on Processing

If an in-out variable for a function (FUN) and function block (FB) is a variable-length array variable, an array variable that you can specify for the in-out parameter is only the one whose data type and number of dimensions are the same as the in-out variable. You can use either a variable-length array variable or fixed-length array variable as long as its data type and number of dimensions are the same as the in-out variable. You cannot specify an array variable whose data type or number of dimensions is different.

If an in-out variable for a function (FUN) and function block (FB) is a fixed-length array variable, you cannot specify a variable-length array variable for the in-out parameter.

Example of in-out variable for function and function block	Example of in-out parameter to specify	Allowed or not allowed
Variable-length array whose data type is INT and dimension is one	Variable-length array whose data type is INT and dimension is one	Allowed.
Variable-length array whose data type is INT and dimension is one	Fixed-length array whose data type is INT and dimension is one	Allowed.
Variable-length array whose data type is INT and dimension is one	Variable-length array whose data type is UINT and dimension is one	Not allowed.
Variable-length array whose data type is INT and dimension is one	Variable-length array whose data type is INT and dimension is two	Not allowed.
Fixed-length array whose data type is INT and dimension is one	Variable-length array	Not allowed.

When you write an assignment expression for the entire array in an ST, if you include a variable-length array variable in either the left side or right side of the expression, a building error will occur. Similarly, when you transfer the entire array with the MOVE instruction in a ladder diagram, if you include a variable-length array variable in either the output parameters or input parameters, a building error will occur.

Left side/ Output parameter	Right side/ Input parameter	Example of expression in ST *1	Allowed or not allowed
Variable-length array variable	Variable-length array variable	varlen_ary := varlen_ary;	Not allowed.
Variable-length array variable	Fixed-length array variable	varlen_ary := fixed_ary;	Not allowed.
Fixed-length array variable	Variable-length array variable	fixed_ary := varlen_ary;	Not allowed.
Fixed-length array variable	Fixed-length array variable	fixed_ary := fixed_ary;	Allowed.

*1. *varlen_ary* shows a variable-length array variable, and *fixed_ary* shows a fixed-length array variable.

● First and Last Element Numbers of Variable-length Array Variable

When the user program is executed, the first and last element numbers are determined. You can obtain the first and last numbers of each dimension of a variable-length array variable with the LOWER_BOUND and UPPER_BOUND instructions.

Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for the details on the LOWER_BOUND and UPPER_BOUND instructions.

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

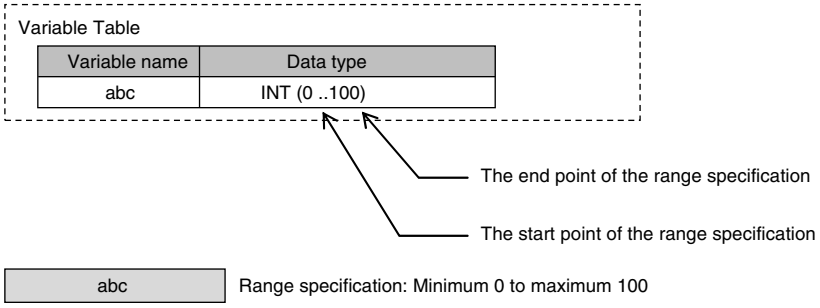
● Making a Range Specification

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

You can specify the range for integer variables.

Operation for attempts to write out-of-range value is given in the following table.

Case	Operation
User program	An error does not occur and the value is written. The CPU Unit does not perform a range check when the value of a variable changes due to the execution of an instruction.

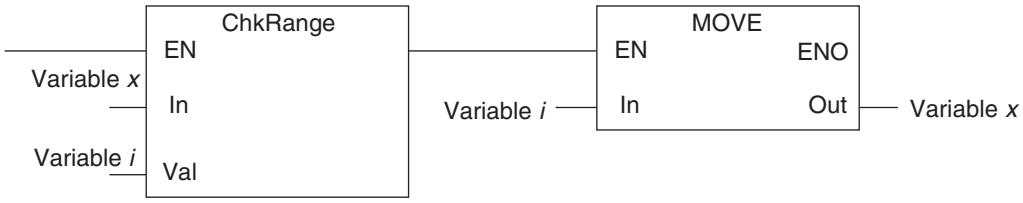
Case		Operation	
Communications	**Write from the Sysmac Studio or a CIP message**	When the value is an integer	A command error occurs.
		For an element of an integer array variable	
		For a member of an integer structure	
		For entire an integer structure	A command error does not occur and the value is written.
		For an entire integer array	
	Tag data links (both via built-in EtherNet/IP ports and EtherNet/IP Units)	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.	
Input refreshing from slaves and Units		An error does not occur and the value is written.	
Forced refreshing values		An error does not occur and the value is written.	

Precautions for Correct Use

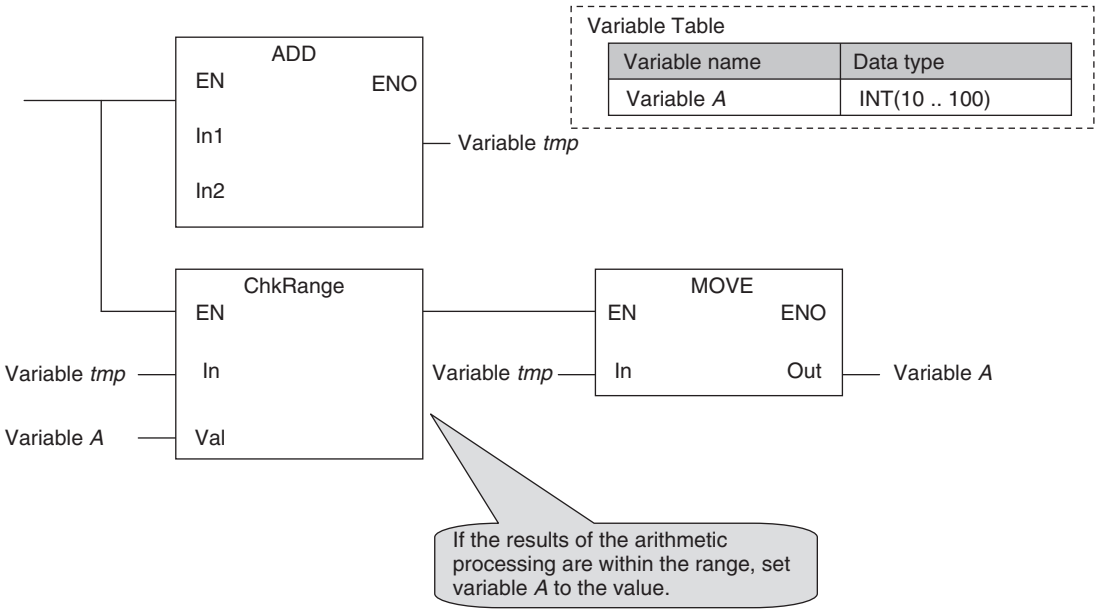
Variables with range specifications are not checked for changes in variable values that result from the execution of instructions in the user program. To check the range of values for a variable that are set from execution of the user program, use instructions that perform range checks.

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* on page 6-84 for restrictions on variable names.

AT Specification

Use the AT Specification attribute to specify the internal I/O memory address of a variable in memory used for CJ-series Units.
AT specifications are used mainly to specify specific memory addresses for the following Special Units.

- Addresses in fixed allocations for DeviceNet Units
- Addresses in user-specified allocations for DeviceNet Units or CompoNet Master Units from the CX-Integrator (A CompoNet Master Unit must be set to communications mode 8 to use the CX-Integrator.)
- Addresses in expansion memory for High-speed Counter Units
- Addresses in expansion memory for Process I/O Units
- Addresses in target node information and user-defined status areas that are used with EtherNet/IP Units

If this attribute is not set, the variable is automatically assigned to an address in variable memory.

Precautions for Correct Use

You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

Additional Information

When you assign a device variable to an I/O port, they are automatically given an AT specification internally.

● Allocation Areas

You can specify addresses in the following areas.

Area	Expression
CIO	0 to 6143
WR	W0 to W511
HR	H0 to H1535
DM	D0 to D32767*1
EM*2	NX502-□□□□: E0_0 to E18_32767 NX102-□□□□: E0_0 to E18_32767 NJ501-□□□□: E0_0 to E18_32767 NJ301-□□□□: E0_0 to E3_32767 NJ101-□□□□: E0_0 to E3_32767

*1. For the NX1P2-□□□□, the address is D0 to D15999.
*2. For the NX1P2-□□□□, the EM Area is not supported.

The following table gives the data assignments by variable data type.

Variable data type	Assignment position
BOOL	You can specify an assignment for each bit.

Variable data type	Assignment position
BYTE/SINT/USINT	You can specify bit 0 or bit 8 of the specified CJ-series address as the start position of the data assignment. Example 1: AT Specification at Bit 0 of D100 (%D100) D100: 16#**12....One-byte data (12) is stored from bit 0. Example 2: AT Specification at Bit 8 of D100 (%D100.8) D100: 16#12**....One-byte data (12) is stored from bit 8.
WORD/INT/UINT	Stored in increments of the data size from bit 0 of the specified CJ-series address.
DWORD/DINT/UDINT REAL	
LWORD/LINT/ULINT LREAL	
STRING	You can specify bit 0 or bit 8 of the specified CJ-series address as the start position of the data assignment.
TIME DATE TIME_OF_DAY DATE_AND_TIME	Stored in increments of the data size from bit 0 of the specified CJ-series address.

● Variables for Which You Can Set AT Specifications

AT specifications are specified separately for each variable. Set them for all elements and members of array, structure, and union variables.

● Attributes of Variables with AT Specifications

	Specification	Remarks
Name	Supported.	
Data Type	Supported.	
Retain	Supported.	An error occurs if the setting of the Retain attribute does not agree with the attribute of the CJ-series Unit memory where the address is assigned.
Initial Value	Supported.	Set the initial value setting to *None* if you want to use the memory value as it is.
Constant	Supported.	You cannot write to a constant with an instruction.
Network Publish	Supported.	
Edge	Not supported.	(You can specify the Edge attribute only for function block input variables.)

● Entering and Displaying AT Specifications

When you specify the AT Specification attribute, input the following in the *AT* field of the variable table in the Sysmac Studio. The following is displayed in the *AT* field of the variable table or the I/O Map.

Type of variable	Entries and displays in the *AT* field.	Example
User-defined variables with AT specifications to word addresses	%[word_address]	%D100
User-defined variables with AT specifications to bit addresses	%[word_address].[bit_position]	%W0.00

The following variables are also allocated an address internally. The following is displayed in the *AT* field.

Type of variable	Displays in the *AT* field.	Example
Device variables for CJ-series Units	IOBus://rack#[rack_number]/slot#[slot_number]/[I/O_port_name]	Basic I/O Units: IOBus://rack#0/slot#1/Ch1_In/Ch1_In00 Special Units IOBus://rack#0/slot#1/PeakHoldCmd/ch1_PeakHoldCmd
Device variables for EtherCAT slaves	NX Units on EtherCAT Slave Terminals ECAT://node#[node_address.NX_Unit_number]/[I/O_port_name]	ECAT://node#[10.15]/Input1
	Other variables ECAT://node#[node_address]/[I/O_port_name]	ECAT://node#1/Input1
Device variables for NX Units[*1]	IOBus://unit#[NX_Unit_number]/[I/O_port_name]	IOBus://unit#8/Input Bit 00
Device variables for X Bus Units[*5]	XBus://unit#[unit_number]/io/[I/O_port_name]	XBus://unit#4/io/PowerOnHour
Device variables for built-in I/O[*2]	BuiltInIO://cpu/#0/[I/O_port_name]	BuiltInIO://cpu/#0/Input Bit 00
Device variables for Option Boards[*2]	BuiltInIO://opt/#[physical_port_number[*3]]/[I/O_port_name]	BuiltInIO://cpu/#1/Ch1 Analog Input Value
Axis Variables	MC://_MC_AX[] MC://_MC1_AX[][*4] MC://_MC2_AX[][*4]	MC://_MC_AX[1]
Axes Group Variables	MC://_MC_GRP[] MC://_MC1_GRP[][*4] MC://_MC2_GRP[][*4]	MC://_MC_GRP[1]

*1. These system-defined variables are available only for the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit.
*2. These system-defined variables are available only for the NX1P2 CPU Unit.
*3. The physical port number indicates a physical location for an I/O port. 0 is given for the option board slot 1 and 1 is given for the option board slot 2.
*4. These system-defined variables are available only for the NX701 CPU Unit.
*5. These system-defined variables are available only for the NX502 CPU Unit.

Precautions for Correct Use

You can assign the same address to more than one variable. However, this is not recommended as it reduces readability and makes the program more difficult to debug. If you do this, set an initial value for only one of the variables. If you set a different initial value for each individual variable, the initial value is undefined.

Retain

Use the Retain attribute to specify whether a variable should retain its value in the following cases.

- When power is turned ON after a 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 CPU Unit is turned OFF.

For a variable with an AT specification, the setting of the Retain attribute must agree with address in the memory area where the address is assigned.
(Retained areas: Holding, DM, and EM Areas
Non-retained areas: CIO and Work Areas)

● Conditions Required to Enable the Retain Attribute

For the CPU Unit that is retained variables by the Battery, the CPU Unit must mount the Battery.

Additional Information

For the NX502, NX102, and NX1P2 CPU Units, retained variables are retained in the non-volatile memory.

● 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		Value of variables	
		Retain attribute specified	Retain attribute not specified
When power is turned ON after a 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 not selected.		
	When the *Clear the present values of variables with Retain attribute* Check Box is selected.	Not retained.	

● Variables for Which You Can Specify the Retain Attribute

AT specifications are specified 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
- Variables with AT specifications (retained areas or DM, Holding, or EM Area specifications only)

Precautions for Correct Use

For the CPU Unit that is retained variables by the Battery, if the CPU Unit has no Battery, the above variables are also initialized.

Constant

If you specify the Constant attribute, the value of the variable cannot be written by any instructions, ST operators, or CIP message communications. Setting the Constant attribute will prevent any program from overwriting the variable.
The values of variables with a Constant attribute cannot be written from instructions after the initial value is set.
If there is an instruction in a POU that attempts to write a value to a variable with the Constant attribute, an error will occur when the user program is built.

● Operation

If there is an instruction or operator in a POU that attempts to write a value to a variable with the Constant attribute, the following operations will occur.

Source		Operation for attempts to write the value
User program		An error is detected during the program check. The Sysmac Studio checks the program when it is built. A building error occurs at that time.
Communications	**Writing from Sysmac Studio**	Not supported.
	CIP messages	A command error occurs.
	Tag data links	An error occurs when a tag data link starts. The tag data link will continue to operate. However, the values of variables with the Constant attribute are not written.
Input refreshing from slaves and Units		An error does not occur and the value is written.
Forced refreshing		

● Range for Constant Attribute Specification

The Constant attribute is specified separately for each variable. Set them for all elements and members of array, structure, and union variables.

Additional Information

You cannot write to variables with the Constant attribute from the user program.

Network Publish

The Network Publish attribute allows a variable to be read/written from external devices (other Controllers, host computers, etc.) through CIP message communications or tag data links.
If this attribute is not set, you can read/write the variable from the Controller that declared the variable and external devices (other Controllers, host computers, etc.) cannot read/write that variable.

Variables that have been published to the network are called network variables.

● Network Publish Specifications

There are three specifications for publishing variables to the network: Publish Only, Input, and Output. The specifications are given in the following table.

Network Publish		Specifications
Do not publish		You cannot access a variable with this attribute from external devices. However, Support Software can still access the variable regardless of this setting.
Publish	Publish Only	You can access a variable with this attribute from external devices through CIP communications. Tag data links are not possible for variables with this attribute setting.
	Input	You can access a variable with this attribute from external devices through CIP communications or a tag data link. For tag data links, this will be a variable for data input (from another CPU Unit to the local CPU Unit).
	Output	You can access a variable with this attribute from external devices through CIP communications or a tag data link. For tag data links, this will be a variable for data output (from the local CPU Unit to another CPU Unit).

● Ranges for Published to the Network

The Network publish attribute is specified separately for each variable. Set them for all elements and members of array, structure, and union variables.

Edge

The Edge attribute makes the variable pass TRUE to a function block when a BOOL variable changes from FALSE to TRUE or from TRUE to FALSE.
You can specify the Edge attribute only for BOOL input variables to function blocks.

● Application

Use the Edge attribute when you want the function block to accept the input only when the input parameter changes from FALSE to TRUE or from TRUE to FALSE.
For example, you can use this attribute when you want to execute the function block any time there is a change detected in an input parameter.

● Operation

- If you specify a change to TRUE, the input variable changes to TRUE only when the input parameter connected to that input variable changes from FALSE to TRUE.

- If you specify a change to FALSE, the input variable changes to TRUE only when the input parameter changes from TRUE to FALSE.

Specification	Value of input parameter	Value of variable
Change to TRUE	FALSE to TRUE	TRUE
	Other	FALSE
Change to FALSE	TRUE to FALSE	TRUE
	Other	FALSE
None	---	Changes according to the input parameter value.

6-3-9 Changes to Variables for Status Changes

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

Version Information

When you are using the or the CPU Unit version 1.13 or later and Sysmac Studio version.1.17 or higher in combination, 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 CPU Unit will change as shown in the following table when the power is turned ON, when the operating mode changes, or when downloading.

Retain attribute of variable	Type of variable	When power is turned ON	When the operating mode is changed	After downloading	
				When the *Clear the present values of variables with Retain attribute* Check Box is selected.	When the *Clear the present values of variables with Retain attribute* Check Box is not selected.
Non-retain	User-defined variables	• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to 16#00. *1			
	AT specifications for CIO and Work Area addresses in the memory used for CJ-series Units	• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to 16#00. *1		• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the previous value is retained.	
	Device variables for EtherCAT slaves*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			
	Device variables for NX Units*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			
	Device variables for built-in I/O*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			
	Device variables for Option Boards*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			
	Device variables for CJ-series Units*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		• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the previous value is retained.	
	IO and Work Area addresses in the memory used for CJ-series Units *2*3	16#00		The previous values are retained.	

Retain attribute of variable	Type of variable		When power is turned ON	When the operating mode is changed	After downloading	
					When the *Clear the present values of variables with Retain attribute* Check Box is selected.	When the *Clear the present values of variables with Retain attribute* Check Box is not selected.
Retain	**User-defined variables**	**Retain condition is met.** *4	No change (retains values before power interruption).	No change (retains values when the operating mode is changed).	• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to 16#00. *1	The values before the download are retained.
		Retain condition is not met. *3				• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to 16#00. *1
		At specifications for Holding, DM, and EM Area addresses in the memory used for CJ-series Units			• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to the values of the memory addresses in the AT specifications.	
	Device variables for CJ-series Units				• If initial values are set, the variables change to the initial values. • If initial values are not set (None), the variables change to the values of the memory addresses in the AT specifications.	The variables change to the values of the memory addresses in the AT specifications.
	Holding, DM, and EM Area addresses in the memory used for CJ-series Units *2				The previous values are retained. • If device variables for CJ-series Units are assigned to *setup data* in the DM Area, the initial values that are set for the variables are used. • If user-defined variables are assigned, the initial values of the variables are used.	The previous values are retained.

*1. Values other than 16#00 may be used depending on the data type. Refer to *When the Initial Value Specification Is Left Blank* on page 6-66 for details.

*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 6-72 for the device output hold configurations.

*3. This does not include user-defined variables and device variables for CJ-series Units if they have AT specifications.

*4. Refer to *Retain Condition* on page 6-74 for the retain conditions.

The values of variables in the CPU Unit 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	

Precautions for Correct Use

You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

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* on page 6-69.
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 the device output hold configuration.

- Device variables for EtherCAT slaves
- Device variables for NX Units
- Device variables for X Bus Units
- Device variables for built-in I/O
- Device variables for Option Boards
- Device variables for CJ-series Units (operating data)
- CIO and Work Area in the memory used for CJ-series Units

● 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
CPU/Expansion Racks	Special Unit Settings of CJ-series Units
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	
DB 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 or Unit operation settings of the NX bus.

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.

5 Right-click a Unit in the CPU and Expansion Racks Tab Page and select **Edit Unit Operation Settings**.

6 Click the **Transfer to Unit** Button for the Unit operation settings.

If you execute **Transfer to Slave** or **Transfer to Unit**, the slave or Unit is restarted after the data is transferred. In this case, device outputs of the slave or Unit are not retained. However, device outputs are retained for the slave or Unit 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-8 Specifications for Individual System-defined Variables* on page A-163 for details on system-defined variables.

Variable name	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 6-75 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 6-77 for the value of a variable when its data type is changed.

Version Information

For the CPU Unit with unit version 1.10 or earlier, the retain condition indicates that all of the following conditions are met both before and after the download.

- The variable name is the same.
- The data type name and data type size are the same.
- The Retain attribute is set to retain the value of the variable.

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 paste 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 • Automatic transfer from SD Memory Cards • Program transfer from SD Memory Card	Pattern 1
• Restoring with variable and memory backup functions on the Sysmac Studio	Pattern 2
• Downloading from the Sysmac Studio	Pattern 2*1

*1. Pattern 1 applies for a CPU Unit with unit version 1.10 or earlier. The pattern difference depends only on the unit version of the CPU Unit. It does not depend on the unit version settings for project devices.

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	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.

● 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

Member	Present value
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
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

Change description	Pattern 1	Pattern 2
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 6-126 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	

Change description	Pattern 1	Pattern 2
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 6-126 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	

Change description	Pattern 1	Pattern 2
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
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 6-126 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 (without NULL)</th><th>Character code</th></tr>
<tr><td>Variable name (including POU instance names)</td><td rowspan="16">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="16">Refer to Reserved Words below.</td><td rowspan="16">Supported.</td><td rowspan="13">Not case sensitive.</td><td rowspan="4">127 bytes</td><td rowspan="16">UTF-8*1</td></tr>
<tr><td>POU definition names</td></tr>
<tr><td>Data type names</td></tr>
<tr><td>Structure member names
and union member names</td></tr>
<tr><td>Enumerator names</td></tr>
<tr><td>Task names</td><td>63 bytes</td></tr>
<tr><td>Name of namespace</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="4">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>

*1. 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 the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)*

- A name that is the same as any of the instructions that are described in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*
- Words that are reserved by the system

Names That Must Be Unique

The following names must be unique. An error is detected during the program check if they are not.

- Global variable names in the same CPU Unit
- Variable names in the same POU
- Section names in the same POU
- Member names in the same union or structure
- Enumerators in the same enumeration
- Local variable names and global variable names
- POU names and data type names
- Data type names and variable names
- Enumerators of an enumeration and enumerators of another enumeration
- Enumerators and variable names

6-4 Constants (Literals)

This section describes constants in detail.

6-4-1 Constants

The value of a variable changes depending on the data that is assigned to that variable. The value of a constant never changes.
Unlike variables, constants are not stored in memory. You can use constants in the algorithm of a POU without the need to declare them.
In the NJ/NX-series Controllers, constants have a data type in the same way as variables.

6-4-2 Notation for Different Data Types

This section gives the notation for constants with different data types. A building error will occur if you use any other notation for a constant.

Boolean Data

"BOOL" is used as the data type name for Boolean data. You can use the following values: 1, 0, TRUE, and FALSE. The meanings of the notations are given in the following table.

Notation	Meaning
TRUE or FALSE	All of the following are equivalent: TRUE, BOOL#1, BOOL#TRUE, and 1. All of the following are equivalent: FALSE, BOOL#0, BOOL#FALSE, and 0.
BOOL#1 or BOOL#0	
BOOL#TRUE or BOOL#FALSE	
1 or 0*1	

*1. Sysmac Studio version 1.03 or higher is required to use 1 and 0. A building error will occur if you use 1 or 0 on Sysmac Studio version 1.02 or lower.

Bit Strings

You can use any of the following data type names for bit string data: BYTE, WORD, DWORD, and LWORD. You can use any of the following bases: 2, 8, 10, and 16. The notations and notation examples are given in the following table.

Notation	Notation example
{data_type_name}#{base}#{numeric_value}	WORD#16#0064
*{base}#{numeric_value}**1	16#0064
*{numeric_value}**1*2	100

*1. Sysmac Studio version 1.03 or higher is required to omit the data type name. A building error will occur if you omit the data type name on Sysmac Studio version 1.02 or lower.
*2. A base of 10 (i.e., a decimal number) is assumed.

Version Information

Sysmac Studio version 1.03 or higher is required to use base 10 for bit string data. A building error will occur if you use 10 on Sysmac Studio version 1.02 or lower.

Integers

You can use any of the following data type names for integer data: SINT, USINT, INT, UINT, DINT, UDINT, LINT, and ULINT. You can use any of the following bases: 2, 8, 10, and 16.
The notations and notation examples are given in the following table.

Notation	Notation example
{data_type_name}#{base}#{numeric_value}	INT#10#-1
*{base}#{numeric_value}**1	10#-1
*{data_type_name}#{numeric_value}**2	INT#−1
*{numeric_value}**1*2	-1

*1. Sysmac Studio version 1.03 or higher is required to omit the data type name. A building error will occur if you omit the data type name on Sysmac Studio version 1.02 or lower.
*2. A base of 10 (i.e., a decimal number) is assumed.

Real Numbers

You can use any of the following data type names for real number data: REAL and LREAL. You can use only base 10 for real number data. The notations and notation examples are given in the following table.

Notation	Notation example
{data_type_name}#{base}#{numeric_value}	LREAL#10#-3.14
*{base}#{numeric_value}**1	10#-3.14
*{data_type_name}#{numeric_value}**2	LREAL#−3.14
*{numeric_value}**1*2	-3.14

*1. Sysmac Studio version 1.03 or higher is required to omit the data type name. A building error will occur if you omit the data type name on Sysmac Studio version 1.02 or lower.
*2. A base of 10 (i.e., a decimal number) is assumed.

Durations

You can use any of the following data type names for durations: TIME and T. The notations and notation examples are given in the following table.

Notation	Notation example
TIME#*{day}*d*{hour}*h*{minutes}*m*{seconds}*s*{milliseconds}*ms	TIME#61m5s
T#*{day}*d*{hour}*h*{minutes}*m*{seconds}*s*{milliseconds}*ms	T#61m5s

The following rules apply to duration data constants.

- It is not necessary to give all of the following: days, hours, minutes, seconds, and milliseconds. You must give at least one of them.
- You can use decimal points, such as in *TIME#12d3.5h*.
- You can give times that exceed the valid time ranges. For example, *T#-61m5s* expresses the same duration as *T#-1h1m5s*.
- All numeric values are interpreted as decimal values. A building error will occur if any number that is not a decimal number is used.
- You can change the order of the duration units. For example, *T#1h2d* expresses the same duration as *T#2d1h*.

Dates

You can use any of the following data type names for date data: DATE and D. The notations and notation examples are given in the following table.

Notation	Notation example
DATE#*{year}*-*{month}*-*{day}*	DATE#2010-1-10
D#*{year}*-*{month}*-*{day}*	D#2010-1-10

The following rules apply to date data constants.

- You can add one or more zeroes to the beginning of the year, month, or day. For example, *DATE#2010-01-10* expresses the same date as *D#2010-1-10*.
- A building error will occur if a valid date range is exceeded. For example, *D#2010-01-35* will cause an error.
- All numeric values are interpreted as decimal values. A building error will occur if any number that is not a decimal number is used.

Times of Day

You can use any of the following data type names for time of day data: TIME_OF_DAY and TOD. The notations and notation examples are given in the following table.

Notation	Notation example
TIME_OF_DAY#*{hour}*:*{minutes}*:*{seconds}*	TIME_OF_DAY#23:59:59.999999999
TOD#*{hour}*:*{minutes}*:*{seconds}*	TOD#23:59:59.999999999

The following rules apply to time of day data constants.

- You can add one or more zeroes to the beginning of the hour, minutes, or seconds. For example, *TOD#23:01:01* expresses the same time of day as *TOD#23:1:1*.
- A building error will occur if a valid time range is exceeded. For example, *TOD#24:00:00* causes an error.
- All numeric values are interpreted as decimal values. A building error will occur if any number that is not a decimal number is used.

Dates and Times

You can use any of the following data type names for date and time data: DATE_AND_TIME and DT. The notations and notation examples are given in the following table.

Notation	Notation example
DATE_AND_TIME#*{year}*-*{month}*-*{day}*-*{hour}*:*{minutes}*:*{seconds}*	DATE_AND_TIME#2010-10-10-23:59:59.123
DT#*{year}*-*{month}*-*{day}*:*{hour}*:*{minutes}*:*{seconds}*	DT#2010-10-10-23:59:59.123

The following rules apply to date and time data constants.

- You can add one or more zeroes to the beginning of the year, month, day, hour, minutes, or seconds. For example, *DT#2010-01-10-23:01:01* expresses the same date and time as *DT#2010-1-10-23:1:1*.
- A building error will occur if a valid date and time range is exceeded. For example, *DT#2010-01-35-00:00:00* or *DT#2010-01-30-24:00:00* causes an error.
- All numeric values are interpreted as decimal values. A building error will occur if any number that is not a decimal number is used.

Text Strings

To give text string data, enclose the text string in single-byte single quotation marks ('). You can also use “STRING” as the data type name. The notations and notation examples are given in the following table.

Notation	Notation example
'*{String}*'	'This is a string'
STRING#'*{String}*'*1	STRING#' ‘This is a string'

*1. Sysmac Studio version 1.08 or higher is required to use “STRING.”

The following rules apply to text string data constants.

- You can also specify a string with 0 characters. To do so, the notation is ’’.
- As in the following example, a building error will occur if you specify any strings that span across multiple lines.
  strVar := 'ABC
  DEF'
- If you want to insert tabs, line break codes, or other special characters, you can use a dollar sign ($) as an escape character before them. The escape character names and meanings are given in the following table.

Escape character	Name	Meaning
$$	Single-byte dollar sign	Single-byte dollar sign ($: character code 0x24)
$'	Single-byte single quotation mark	Single-byte single quotation mark (': character code 0x27)
$"	Single-byte double quotation mark	Single-byte double quotation mark (”: character code 0x22)
$L or $l	Line feed	Moves the cursor to the next line. LF control character (line feed: character code 0x0A)

Escape character	Name	Meaning
$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.

Enumerations

For enumeration data, the enumeration data type name and enumerator are given. The notations and notation examples are given in the following table.

Notation	Notation example
{enumeration_data_type_name}#{enumerator}	_eDAYOFWEEK#_WED

Additional Information

To pass an enumerator to a function or function block for which the parameter specifies an enumerator, you can omit the enumeration data type name and give only the enumerator.
For example, the _eBCD_FORMAT enumeration is specified for the *Format* input variable in the BinToBCDs instruction. Therefore, you can give either the enumeration data type name and enumerator as _eBCD_FORMAT#_BCD0 or omit the enumeration data type name and give only _BCD0.

6-5 Programming Languages

This section describes the programming languages in detail.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on entering programs with the Sysmac Studio.

6-5-1 Programming Languages

The languages used to express the algorithms in a POU (program, function, or function block) are called the programming languages.
There are two different programming languages that you can use for an NJ/NX-series Controller: ladder diagram language (LD) and ST (structured text) language.

6-5-2 Ladder Diagram Language

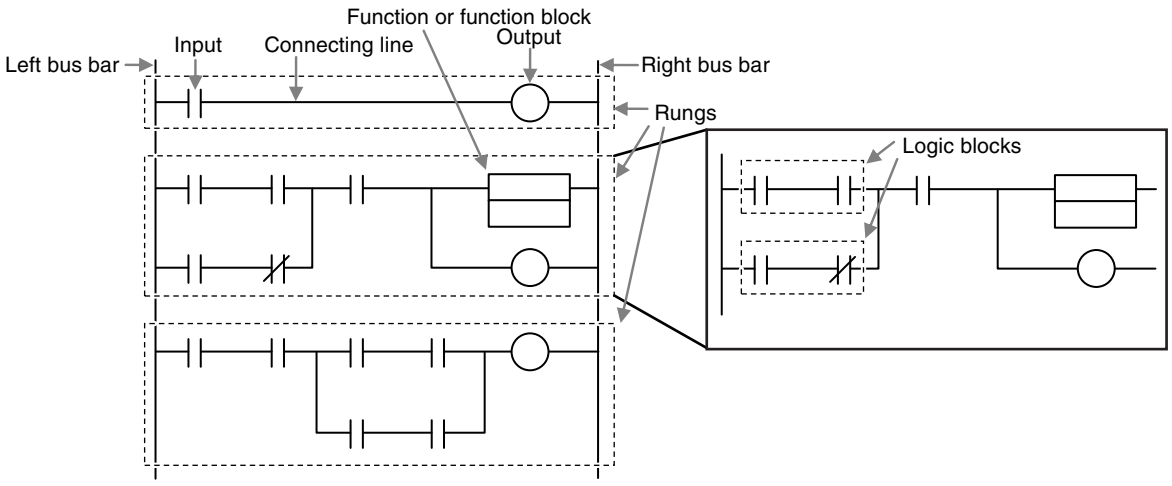
The ladder diagram language (LD) is a graphical programming language that is written in a form that appears similar to electrical circuits. Each object for processing, including functions and function blocks, is represented as a diagram. Those objects are connected together with lines to build the algorithm. Algorithms that are written in the ladder diagram language are called ladder diagrams.

General Structure of the Ladder Diagram Language

A ladder diagram consists of left and right bus bars, connecting lines, ladder diagram structure elements (e.g., inputs and outputs), functions, and function blocks.*

* Only Jump instructions and Label instructions are expressed with symbols that indicate the jumps and labels.

Algorithms are made of multiple rungs connected together.
A rung is a connection of all configuration elements between the left bus bar and the right bus bar. A program rung consists of logic blocks that begin with an LD/LD NOT instruction that indicates a logical start.



● Bus Bars

The vertical lines on the left and right sides of a ladder diagram are called the bus bars.
These bus bars always have a status of either TRUE or FALSE. If you think of the ladder diagram as an electrical circuit, these states represent the flow of current through the circuit.
When a POU that is written as a ladder diagram is executed, the value of the left bus bar changes to TRUE. As a result, all inputs and other configuration elements connected to the left bus bar also become TRUE. Execution progresses as elements to the right are also changed to TRUE based on the operation of these configuration elements.
This cascade of the TRUE state is called the “power flow.” The left bus bar is the source of this power flow.

● Connecting Lines

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 FALSE.
For detailed specifications on inputs, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)*.

● Outputs

Outputs are placed along the connecting line to receive the power flow and operate accordingly.
An output writes the TRUE or FALSE value to a variable.
There are different types of outputs. For detailed specifications on outputs, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)*.

● Functions and Function Blocks

Functions and function blocks are placed along the connecting line to receive the power flow and operate accordingly.
For detailed instruction specifications, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)*.

Order of Execution for Ladder Diagrams

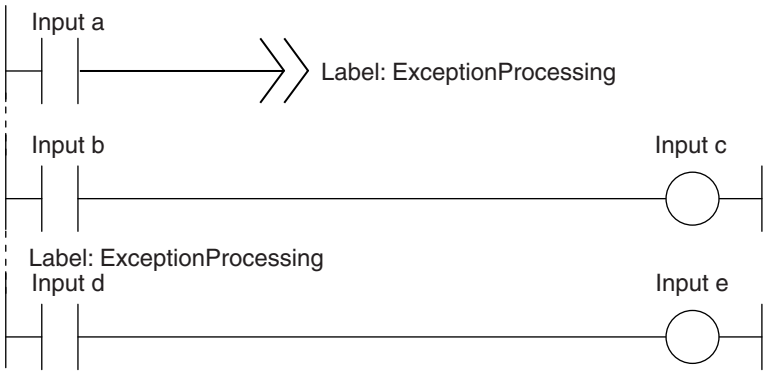
Inputs, outputs, functions, and function blocks are executed when they receive the power flow.
The order of execution for a ladder diagram is from top to bottom. Elements at the same level are executed from left to right.

Ladder Diagram Completion

A ladder diagram is executed in order from top to bottom. When the execution reaches the very bottom, the process is completed.
However, the process will also end if an END or RETURN instruction is encountered at any point during the process. No processes after those instructions are executed.

Controlling Execution of Ladder Diagrams

Ladder diagrams are generally executed from top to bottom, but you can use execution control instructions to change the execution order.
In the following example, when the value of program input a changes to TRUE, execution will move to the point labeled 'ExceptionProcessing'.



Connecting Functions and Function Blocks in a Ladder Diagram

● Connection Configurations

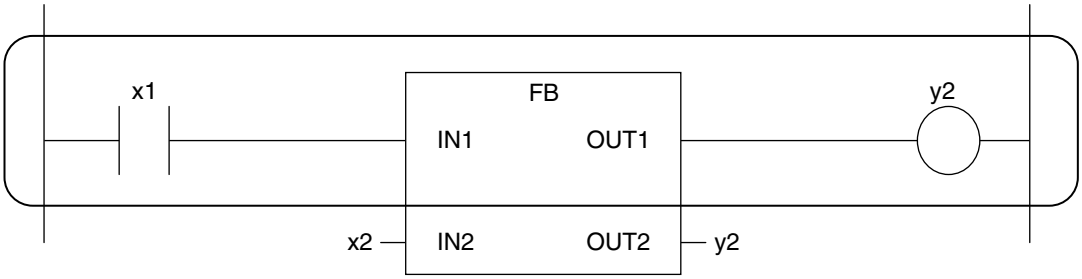
You use the following two types of connections for functions or function blocks.

1. **Power Flow Input and Output**

   In a ladder diagram, the line that connects an input variable of a function or function block and the left bus bar indicates a BOOL input and the line that connects an output variable to the right bus bar indicates a BOOL output.

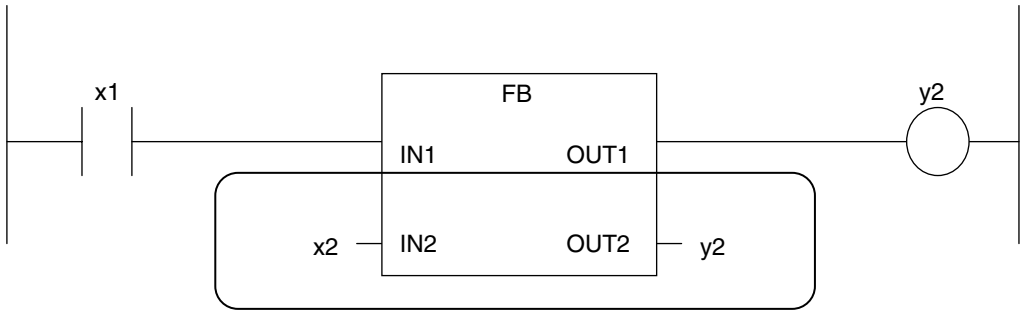
   Example:

   Inputs are connected in the power flow that connects to the left bus bar. Outputs are connected in the power flow that connects to the right bus bar.



2. **Parameter Inputs and Parameter Outputs**

In a ladder diagram, parameter inputs and outputs are specified when the input and output variables of a function or function block are not connected to the left and right bus bars.



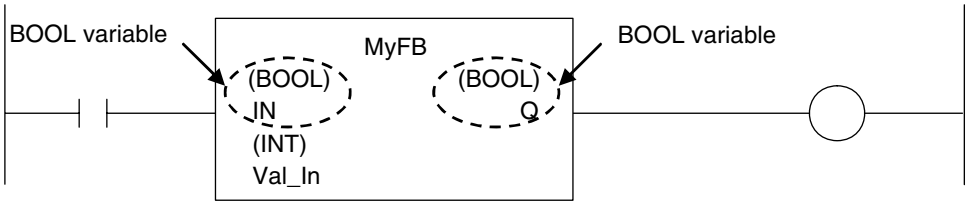
As shown below, you can specify either variables or constants for input and output parameters.

Function/function block variables	Input parameters	Output parameters
Input variables	You can specify variables or constants.	---
Output variables	---	You can specify only variables.
In-out variables	You can specify only variables.	You can specify only variables.

● Number of BOOL Variables

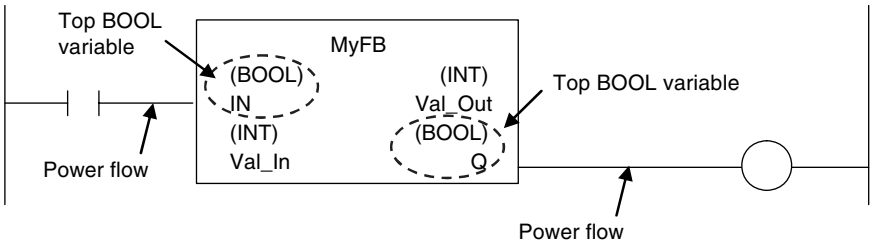
At least one BOOL variable each is required for the input and the output (such as *EN* and *ENO*) of a function or function block.

Example:

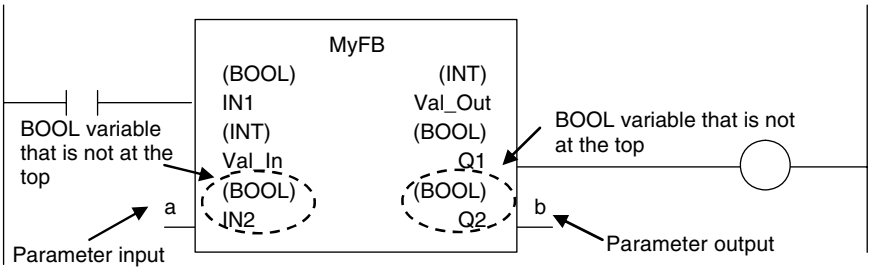


● Connections Based on the BOOL Variable Positions

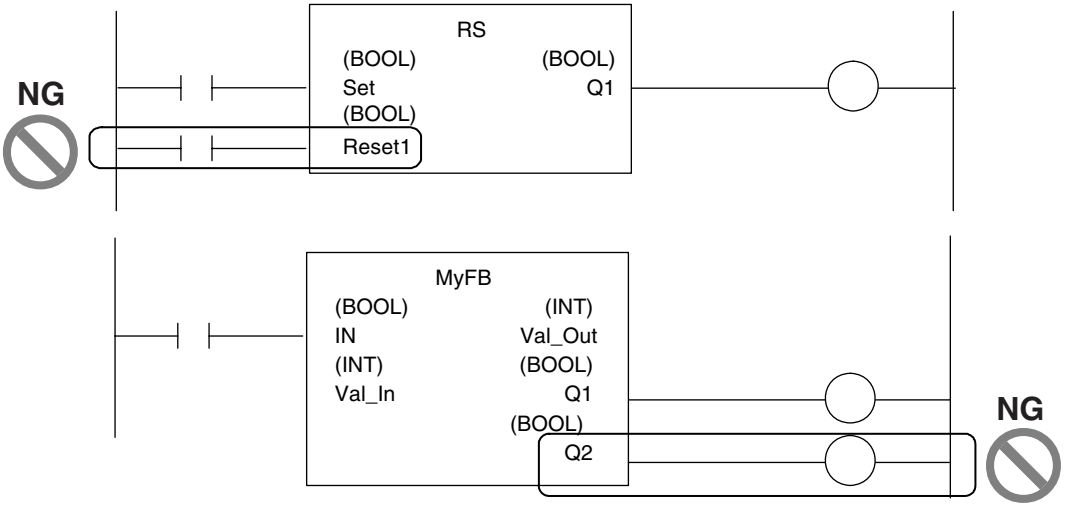
The top BOOL variables are connected to the left and right bus bars. In other words, they become the power flow input and power flow output.



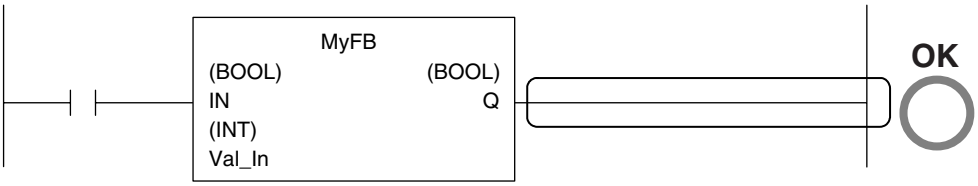
There is only one power flow input and one power flow output for each function or function block. All other BOOL variables that are not at the top are for parameter inputs and parameter outputs.



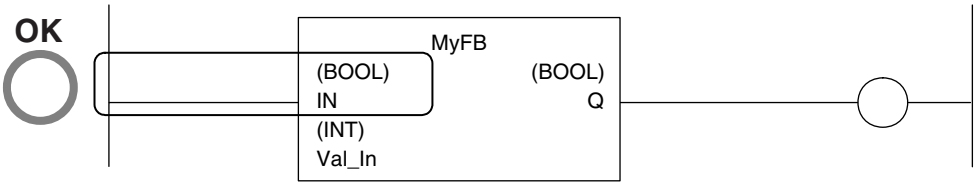
You cannot connect multiple BOOL variables to the left bus bar or the right bus bar as shown below.



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

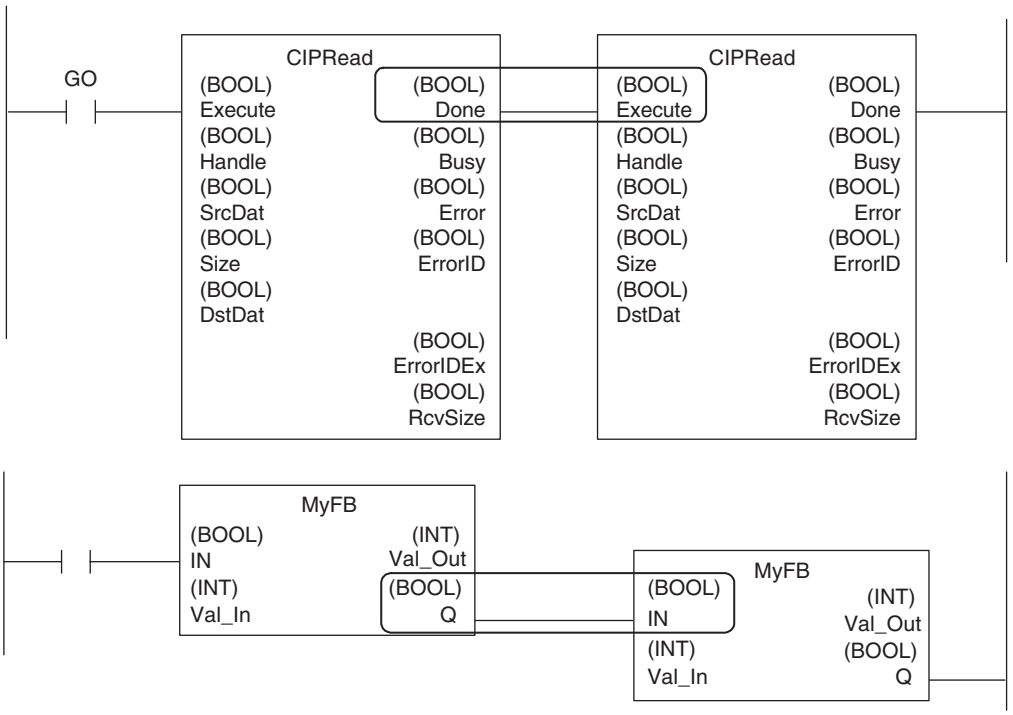


A LD instruction is not necessarily required. You can also connect directly to the left bus bar.



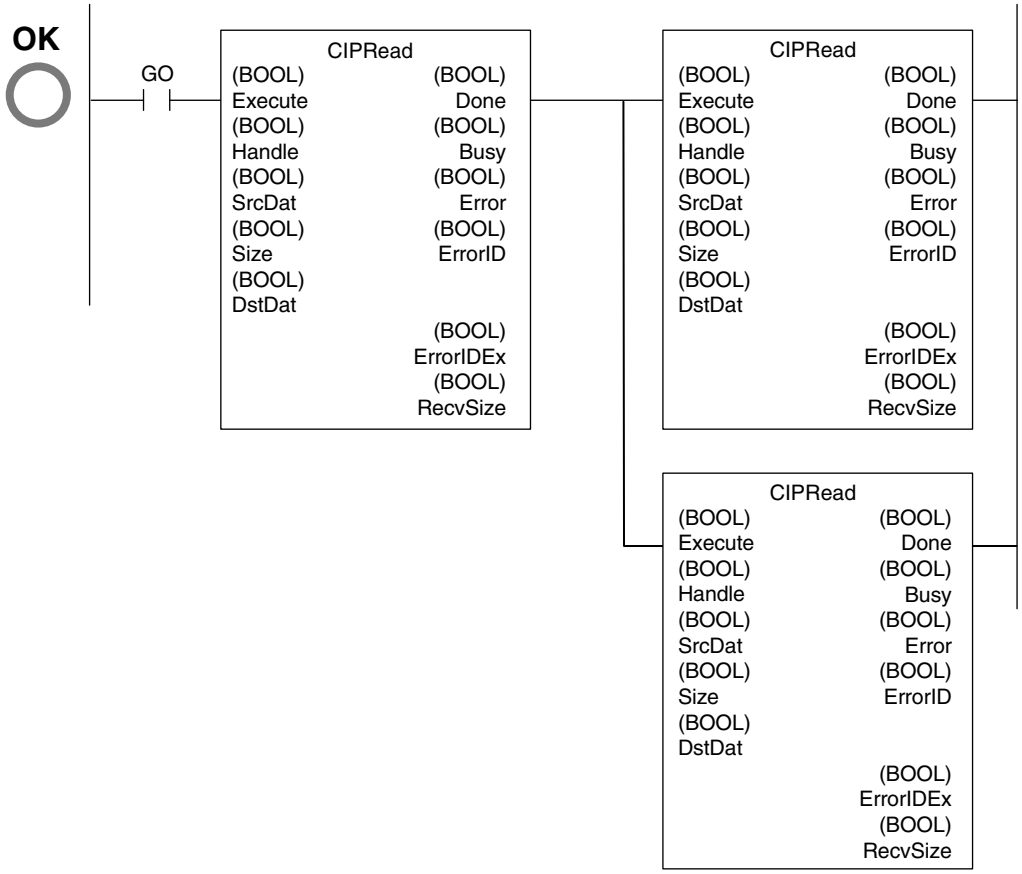
● Cascade Connections

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



- You can branch the power flow output.

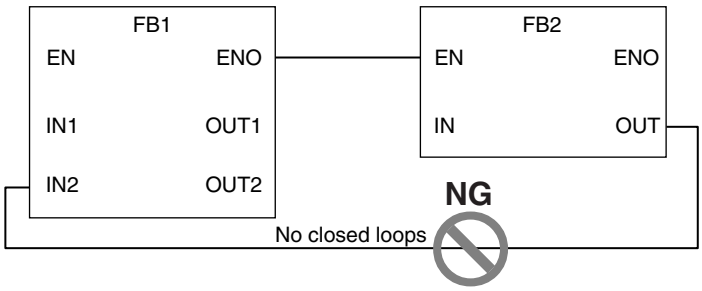
Example:



Restriction

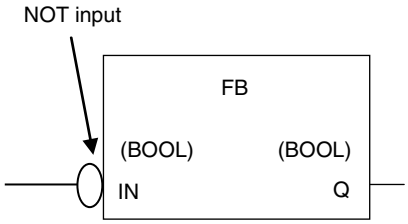
- You cannot create closed loops or intersect connecting lines.

Example:



● Reversing Inputs

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



TRUE→FALSE

FALSE→TRUE

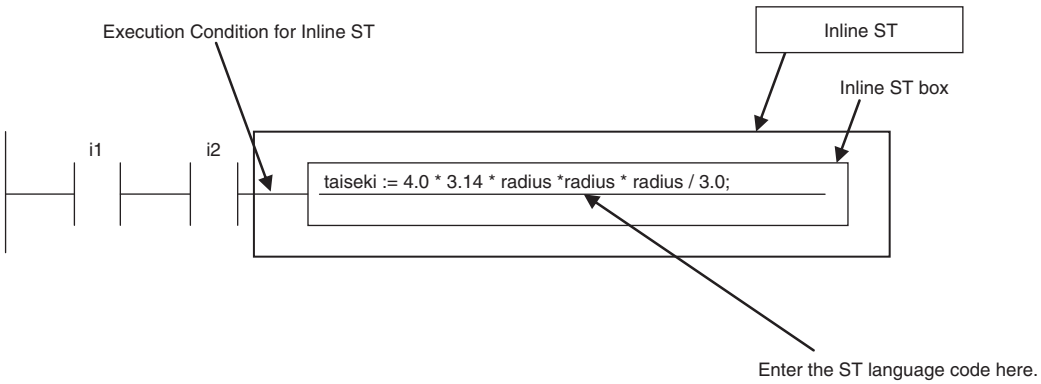
Inline ST

● 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.

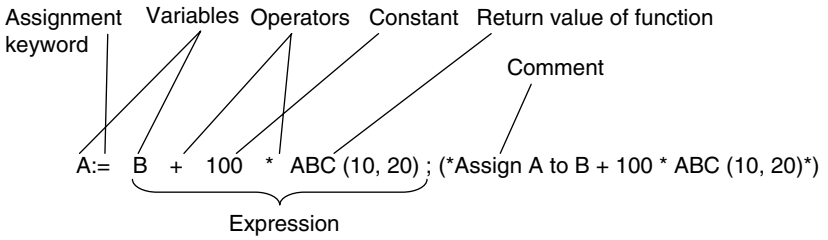
- Loop constructs and control constructs such as IF THEN ELSE are provided.
- You can write programs like high-level languages such as C, and you can include comments to make the program easy to read.

```
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;
```

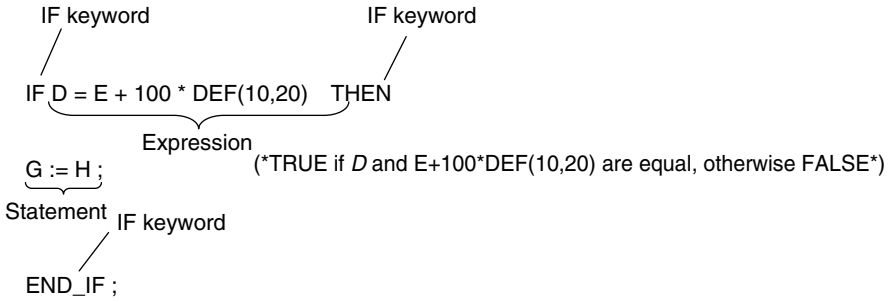
Structure of ST

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: ■ indicates 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* on page 6-84 for restrictions on variable names.

● Text Strings

Refer to *6-3-12 Restrictions on Variable Names and Other Program-related Names* on page 6-84 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 6-121.
RETURN	Return	

Keyword	Meaning	Example
IF	If	IF d e THEN f := 1; ELSIF d = e THEN f :=2; ELSE f := 3; END_IF;
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;

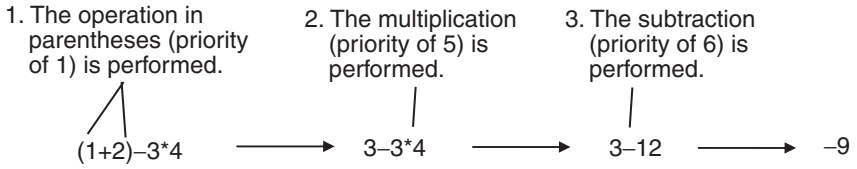
;	Empty statement	Val[i] := i ; (* Empty statement *) WHILE (Var <>0) DO ; (* Empty statement *) END_WHILE;
(* Text *)	Comment	(* Commenting out multiple lines IF MyFun (ErrorCode) = 3 THEN ReturnValue := GetDetail(); END_IF; down to here. *)
//Text	Comment	A := SIN(X) ^ 2 + COS (Y) ^2 + 10; // A := 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: 20000	5
Division	/	100/200 Value: 0.5	
Modulo division	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	
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

Operation	Operator	Notation example and evaluated value	Priority
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:  Incorrect notation   out  := MyFunc( In1:=x1, In2:=MyFunc( In1:=x2, In2:=x3 ) );
          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 manufactures. 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 6-126 for details on implicit casting.

○: Possible
x: A building error will occur.

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	○	○	×	×	×	×	○	○	×
Bit string	○	○	×	×	×	×	○	○	×
Integer	○	○	○	○	○*1	○	○	×	○
Real number	○	○	○	×	○	○	○	×	○
Duration	○	○	×	×	×	×	×	×	○
Date	○	○	×	×	×	×	×	×	×
Time of day	○	○	×	×	×	×	×	×	×
Date and time	○	○	×	×	×	×	×	×	×
Text string	○	○	×	×	×	×*2	×*2	×	×
Enumeration	○	○	×	×	×	×	○	×	×
Structure parent	○	○	×	×	×	×	×	×	×
Array parent	○	○	×	×	×	×	×	×	×

*1. If you select the version 1.15 or earlier in the Select Device Area of the Project Properties Dialog Box on the Sysmac Studio for an NX701 CPU Unit and NJ-series CPU Unit, 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:

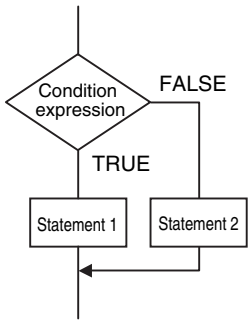
IF, THEN, (ELSE), END_IF

Note: You can omit ELSE.

Statement 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.

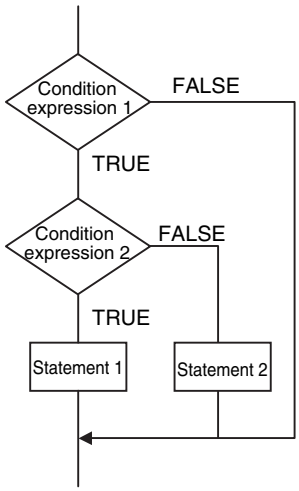
Precautions:

- 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;
```

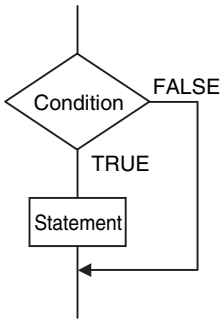
Process Flow Diagram:



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.

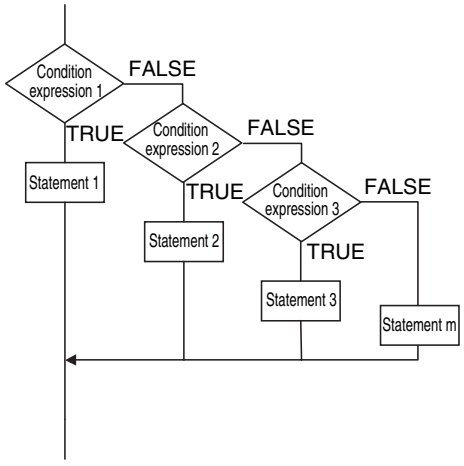
Reserved Words:

IF, THEN, ELSIF, (ELSE), END_IF
Note: You can omit ELSE.

Statement 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 <*conditon_expression_2*> is TRUE, then <*statement_2*> is executed.
If <*condition_expression_2*> is FALSE and <*conditon_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 a BOOL variable (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, the 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.

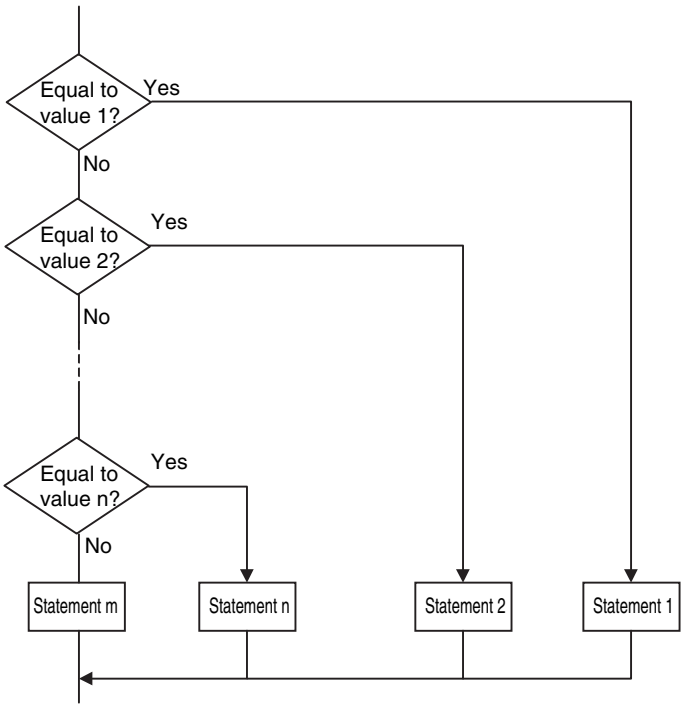
Reserved Words:

CASE

Statement 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:
       X := 10;
    1:
       X := 11;
    ELSE
       X := 12;
END_CASE;
```

Branches depending on the return value of FUN( ).

Data Types That You Can Use in CASE Constructs

○: Supported
×: Not supported

Classification	Data type		<integer_expression>
Basic data types	Integers		○
	Boolean, bit string, real, duration, date, time of day, date and time, or text string data		×
Data type specifications	Array specifications	Arrays	×
		Elements	○ For integers and enumerations only.
Derivative data types	Structures	Structures	×
		Members	○ For integers and enumerations only.
	Unions	Unions	×
		Members	○ For integers and enumerations only.
	Enumerations		○

● 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:

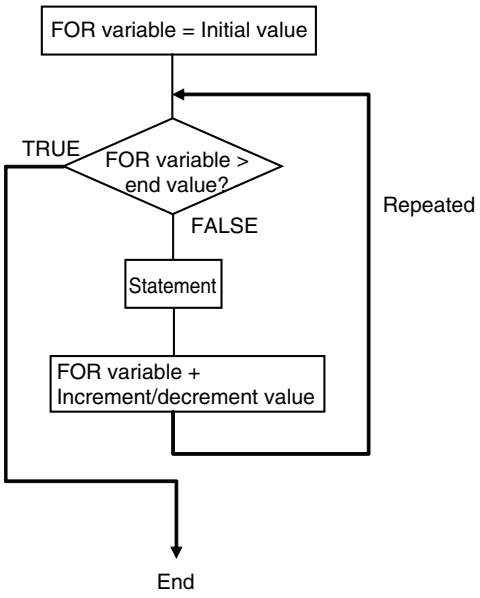
FOR, TO, (BY), DO, END_FOR

Note: You can omit BY.

Statement 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 assigned 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] < MIN 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:

When the data type of variable *i* is USINT, the end value 256 can be expressed. It is treated as 0 and an infinite loop occurs.

```
FOR i := 0 TO 254 BY 2 DO
  INTArray[i] := i;
END_FOR;
```

Version Information

With the Sysmac Studio version 1.08 or higher, you can specify arithmetic expressions for <*end_value*> and <*increment/decrement*>.
However, the evaluation is performed for <*end_value*> or <*increment/decrement*> only before the execution of FOR loop operation. The values of <*end_value*> and <*increment/decrement*> do not change after the FOR loop operation is started.
For example, in the following case, the value of <*end_value*> is 10 and <*increment/decrement*> is 3. Even after the FOR loop operation is started and the values of variable *A* and *C* are changed, the value of <*end_value*> is still 10 and <*increment/decrement*> is still 3.

```
A := INT#1;
B := INT#2;
C := INT#10;

FOR i := 0 TO C BY A+B DO
  INTArray[i] := i;
  A := B + i;
  C := C + i;
END_FOR;
```

If an arithmetic expressions is specified for <*end_value*> or <*increment/decrement*> on the Sysmac Studio version 1.07 or lower, a building error will occur.

Data Types That You Can Use in FOR Constructs

○: Supported
×: Not supported

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		×
	Integers		○
Data type specifications	Array specifications	Arrays	×
		Elements	○ For integers and enumerations only.*2
Derivative data types	Structures	Structures	×
		Members	○ For integers and enumerations only.*2
	Unions	Unions	×
		Members	×
	Enumerations		○*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.

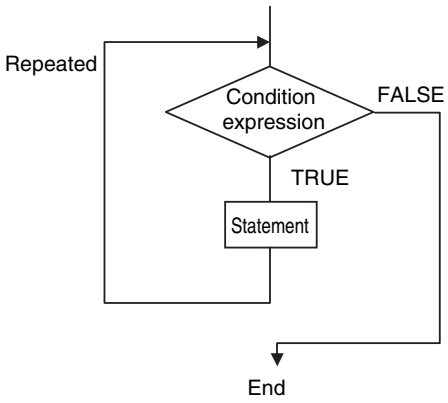
Reserved Words:

WHILE, DO, END_WHILE

Statement 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 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.

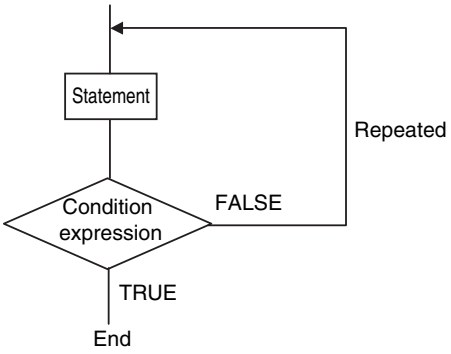
Reserved Words:

REPEAT, UNTIL, END_ REPEAT

Statement 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 whether to 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: 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

Statement 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.

Note 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

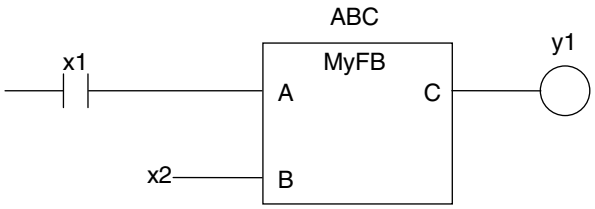
A and B: Input or in-out variable names of called function block

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

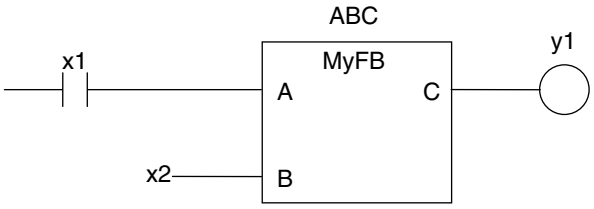


- You can give the arguments and return values in any order.
- You can omit the input variable names and input parameter names. If you omit these names, the values assigned to the input variables for the previous call are assigned to the input variables again. If this is the first time that the function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

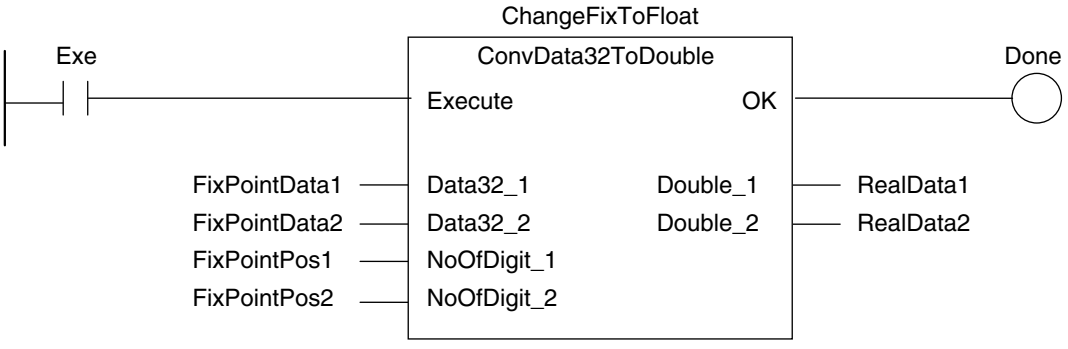
Example
- Programming
- Notation 1

```
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	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
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.

○: Possible (initial used).

x: A building error will occur.

POU type	Variables for the called POU	Notation pattern		Omission
		Parameters included	Example	
FB	Given (notation method 1)	All parameters given	instance(x:=a,y:=b,z:=c);	○
		More than one parameter given	instance(x:=a,y:=b);	
		One parameter given	instance(y:=b);	
		No parameters given	instance(x:=);	×
	Not given (notation method 2)	All parameters given	instance(a,b,c);	○
		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);	

● 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.

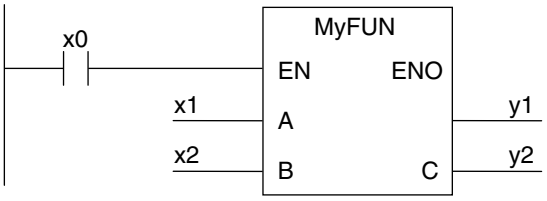
There are two methods of writing this statement, as shown in (1) and (2) below.

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

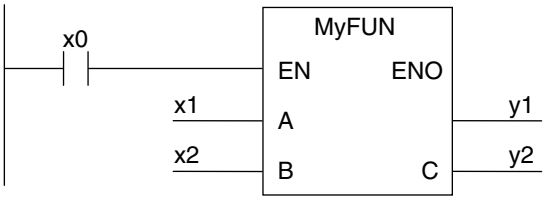
- You can give the arguments in any order.
- You can omit the input variable names and input parameter names. If they are omitted, the input variables are assigned their initial values.
- You can omit *EN* as well. If it is omitted, *EN* is assigned a value of TRUE.

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

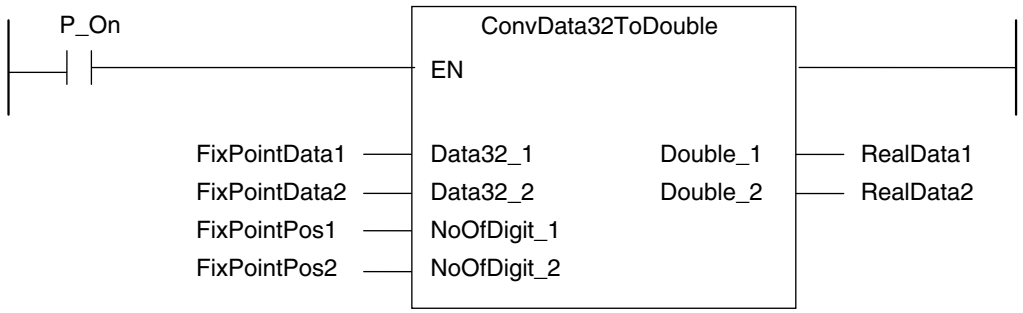
- Programming
- Notation 1

```
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.

Variable name	Data type	Comment
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

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.

○: Possible (initial used).
x: A building error will occur.

POU type	Variables for the called POU	Notation pattern		Omission
		Parameters included	Example	
FB	Given (notation method 1)	All parameters given	FUN(x:=a,y:=b,z:=c);	○
		More than one parameter given	FUN(x:=a,y:=b);	
		One parameter given	FUN(y:=b);	
		No parameters given	FUN(x:=);	×
	Not given (notation method 2)	All parameters given	FUN(a,b,c)	○
		No parameters given	FUN();	
		Only the first parameter given	FUN(a);	×
		One parameter given	FUN(a, ,);	
		More than one parameter given	FUN(a,b);	

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.

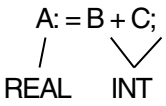
1. When the data types of the operands in the expression on the right side of the assignment statement are not the same

   Example:

```
A: = INT#10 + SINT#2;
```

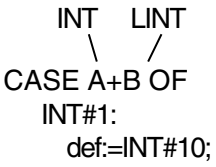
2. When the data types of the operands on the right and left sides of the assignment statement are not the same

   Example:



3. When the data types of the operands in statement are not the same

   Example:



The casting rules are described for the following three cases.

Casting Rules When the Right-hand Side of an Assignment Statement Is an Arithmetic Expression

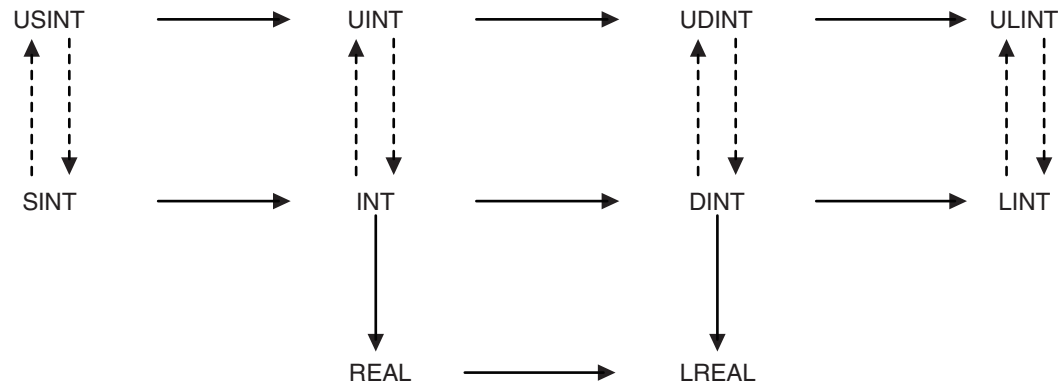
- For the right-hand operand, you can use any combination of the data types that are supported for the operator operand.
- Of the operands on the right side, the operand with the highest rank is considered the data type of the entire side.
  (Refer to the *Data Type Ranking Table* given below for the data type ranks.)

Data Type Ranking Table:
The higher the rank, the larger the range of numerical values that the data type can express.

Rank	Data type
1	SINT
2	USINT
3	INT
4	UINT
5	DINT
6	UDINT
7	LINT
8	ULINT
9	REAL
10	LREAL
11	BYTE
12	WORD
13	DWORD
14	LWORD

Casting Rules When You Assign the Right-hand Value to the Left-hand Side

In the following chart, a cast is performed if an arrow connects the data type of the source to the data type of the assignment destination.
Any combination that is not connected will cause a building error.



When you assign the value, the sign and absolute value of the number do not change.

When you assign the value, the sign and absolute value of the number may change.

Example:
```
intVar := -1;          (* intVar := 16#FFFF *)
uintVar := 1;
uintVar := intVar;     (* uintVar:= 16#FFFF, or –1 was
                          assigned but the result is 65535 *)
```

Even if the arrow does not connect directly to a data type, you can still perform assignments for the data types.
For example, SINT->USINT->UINT->UDINT->ULINT are all connected, so you can write an assignment such as ULINT:=SINT.

Precautions for Correct Use

Observe the following precautions when casting UDINT to ULINT data, DINT to LINT data, or DINT to LREAL data.
All of these are casts from 32-bit data to 64-bit data. If the result of the calculation of the right side of the assignment statement exceeds the range of 32-bit data, the correct value may not be assigned.

Example: For the following assignment statements, the result of the addition in the third statement exceeds the range of 32-bit data. An overflow will result and 0 will be assigned to *LintVar*.

```
UdintVar := UDINT#16#FFFF_FFFF;           // Upper limit of 32-bit data
DintVar := DINT#1;                        // 1
LintVar := (UdintVar + DintVar) / DINT#2; // (Upper limit of 32-bit data+1)/2
```

In a case like this one, convert the data to 64-bit data before you perform the calculation. To do this for the above example, change the assignment status as shown below.

```
LintTmp1 := UDINT_TO_LINT(UDINT#16#FFFF_FFFF); // Convert UDINT to LINT data.
LintTmp2 := DINT_TO_LINT(DINT#1);              // Convert DINT to LINT data.
LintVar := (LintTmp1 + LintTmp2) / DINT#2;
```

Casting Rules in Expressions in Statements

The implicit cast rules for right-hand arithmetic expressions in assignment statements and for assigning the value of the right-hand side to the left-hand side also apply to expressions in statements.

Example:

```
CASE (A+B+C) OF
  Result1:
     to;
  ResultN:
```

```
        to;
    END_CASE;
```

Order of Execution of Functions

The order of execution of functions is not defined for functions in expressions. The order of execution of functions depends on the unit version of the CPU Unit, the version of the Sysmac Studio, and the notation. Precaution is required in cases where the results of an expression may depend on the order of execution of the functions, such as in the following cases.

- Expressions that contain more than one function that access the same global variable
- Expressions that contain a function and a variable whose value is changed by that function

Expressions That Contain More Than One Function That Access the Same Global Variable

In the following example, the order of execution of the three functions is not necessarily the same as the order of execution of the calculations, which is determined by the priority of the operators. Therefore, it is possible that the functions are executed in the following order: FUN2, FUN3, and then FUN1.

```
result := FUN1() + FUN2() * FUN3();
```

If all three of the functions in the above expression access and write the same global variable, the value of the *result* variable may change depending on the order of execution of the functions.
To ensure that the three functions are always executed in the same order, the expression is broken up. The following notation is used to execute the functions in the following order: FUN2, FUN3, and then FUN1.

```
tmp2   := FUN2();
tmp3   := FUN3();
result := FUN1() + tmp2 * tmp3;
```

Expressions That Contain a Function and a Variable Whose Value Is Changed by That Function

The following expression contains a function and a variable whose value is changed by that function.

```
result := varA + FUN4(out => varA);
```

In the above expression, the first element on the right side, variable *varA*, is not necessarily evaluated before FUN4 is executed. Therefore, the value of the *result* variable may change depending on the order of *varA* evaluation and FUN4 execution.
To ensure that that *varA* evaluation and FUN4 execution always occur in the same order, the expression is broken up.
The following notation is used to evaluate *varA* first and then execute FUN4.

```
tmp    := varA;
result := tmp + FUN4(out => varA);
```

The following notation is used to execute FUN4 first and then evaluate *varA*.

```
tmp    := FUN4(out => varA);
result := varA + tmp;
```

Calculation Precision of Expressions with Constants without Data Type Specifications

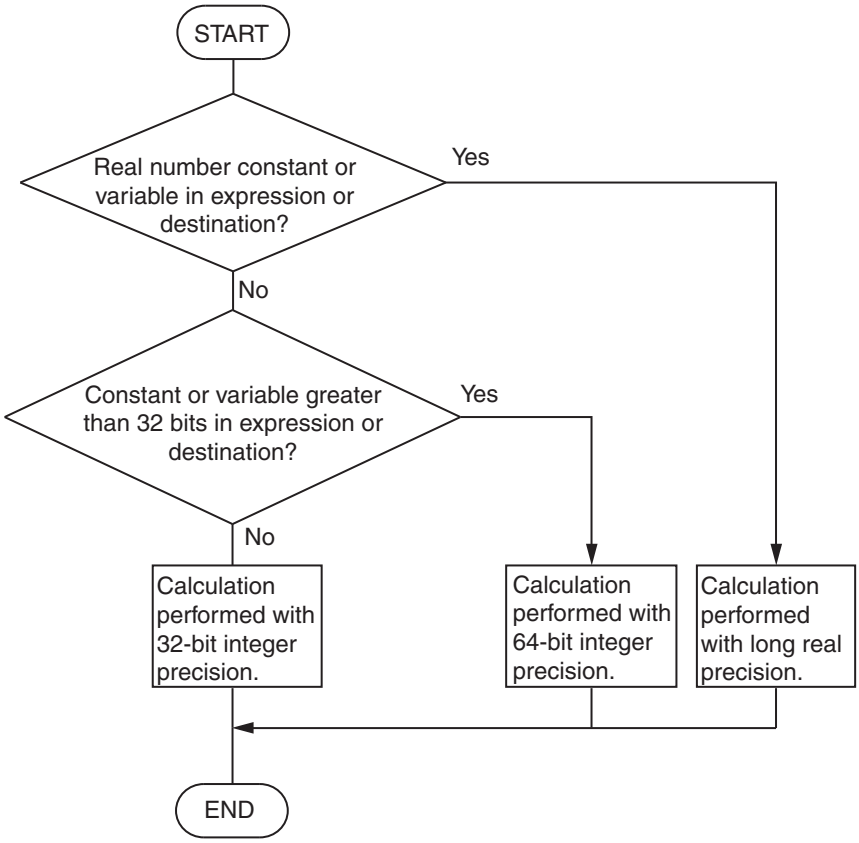
The calculation precision of an expression that contains a constant without a data type specification is automatically determined by the data types of the variables and constants that are given in the expression and destination.

Notation of Constants

If a constant is given without a decimal point, such as 100, it is processed as an integer. If a constant is given with a decimal point, such as 100.0, it is processed as a real number.

Expression Calculation Precision

The calculation precision of an expression is either 32-bit integer, 64-bit integer, or long real precision depending of the data types of the variables and constants given in the expression and destination. The following rules apply to the calculation precision of an expression.



Example:

```
realv := 2 + 3 *4; // The data type of the realv variable is REAL.
```

The *realv* variable is a real number, so the calculation is performed with long real precision. The calculation result is 14.0.

```
realv := 2 + 3.0 *4; // The data type of the realv variable is REAL.
```

The 3.0 constant and the *realv* variable are real numbers, so the calculation is performed with long real precision. The calculation result is 14.0.

```
lintv := 2 + 3 * 4; // The data type of the lintv variable is LINT.
```

There is no constant or variable that is a real number, but the *lintv* variable exceeds 32 bits, so the calculation is performed with 64-bit integer precision. The calculation result is 14.

```
intv := 2 + 3 * 4; // The data type of the intv variable is INT.
```

There is no constant or variable that is a real number and there is no constant that exceeds 32 bits, so the calculation is performed with 32-bit integer precision. The calculation result is 14.

However, the calculation precision of division is determined only by the divisor and dividend. The rules for determining the calculation precision are the same as those in the previous flowchart.

Example:

```
realv := 2 / 3 *4; // The data type of the realv variable is REAL.
```

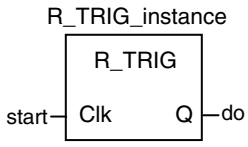
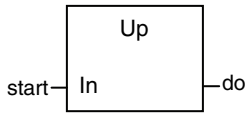
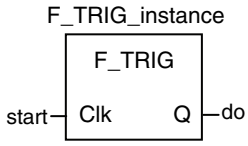
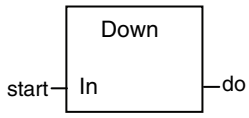
Dividing 2 by 3 does not include an integer that exceeds 32 bits for the divisor or dividend, so the calculation is performed with 32-bit integer precision. The calculation result is 0.
In the next step, the *realv* variable is a real number, so the calculation of 0 * 4 is performed with long real precision. The calculation result is 0.0.

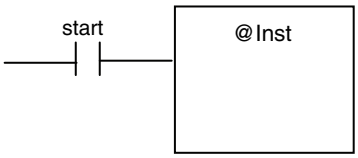
Precautions for Correct Use

The calculation precision of an expression that contains a constant without a data type specification is automatically determined by the notation of the constant and the data types of the variables that are given in the expression. Therefore, calculations may be performed with unintended precision. We recommend that you specify the data type for real numbers, such as REAL#1.0.

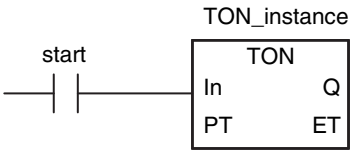
Differences between ST and Ladder Diagrams

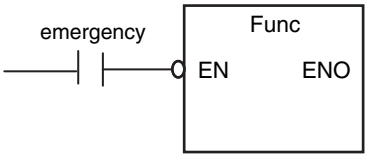
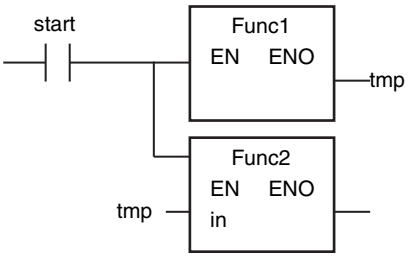
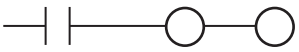
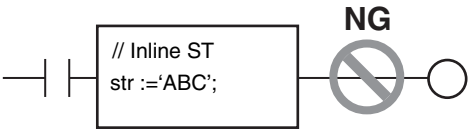
The differences between ST and ladder diagrams are described below.

Item	Ladder diagram	ST (including inline ST)
Input differentiation	**Change to TRUE** • Method 1 start do ----|↑|----------○ • Method 2  • Method 3 	**Change to TRUE** • Method 1 R_TRIG_instance (Clk:=start, Q=>do); * R_TRIG_instance is an instance of the R_TRIG instruction.
	Change to FALSE • Method 1 start do ----|↓|----------○ • Method 2  • Method 3 	**Change to FALSE** • Method 1 F_TRIG_instance (Clk:=start, Q=>do); * F_TRIG_instance is an instance of the F_TRIG instruction.

Instruction differentiation	**Upward Differentiation**
	 
		Upward Differentiation
		There is no equivalent in ST. You must create it in logic.
		Example:
		• Method 1
		R_TRIG_instance (Clk:=start, Q=>do);
		IF (do = TRUE) THEN Inst();
		END_IF;
		• Method 2
		IF (start = TRUE) THEN
		IF (pre_start = FALSE) THEN Inst();
		END_IF;
		END_IF;
		pre_start:=start;// Update previous value.

Item	Ladder diagram	ST (including inline ST)

Instructions that last multiple task periods	With the TON instruction, multiple cycles are required from the start of instruction execution to the end and the instruction is reset when the power flow is FALSE. Therefore, you need to declare only one instance to both execute the instruction and reset it.
	 
		You must declare two instances, one for execution and one to reset, as shown below.
		IF (start = TRUE) THEN
		TON_instance(In:=TRUE, *omitted*); // Start timer.
		ELSE
		TON_instance(In:=FALSE, *omitted*); // Reset timer.
		END_IF;

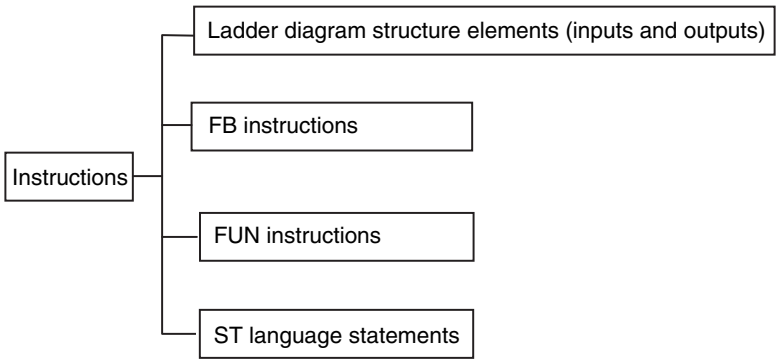
Function/ function block argument reversal specifications	Add a circle to indicate reversal at the intersection of the BOOL argument and the function/function block. 	Add a NOT operator to the argument. * You can add NOT operators to any BOOL variable, not just arguments. IF (NOT emergency) THEN Func(); END_IF;
Multi-stage connections		IF(start=TRUE) THEN Func2(in := Func1()); END_IF;
Post-connecting ladder instructions	You can connect only other Out instructions after an Out instruction. 	You cannot continue the ladder diagram after inline ST. 
Program divisions	You can create sections.	You cannot create sections.

6-6 Instructions

This section describes the instructions that are pre-defined by the NJ/NX-series Controller.
For details on these instructions, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* and *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

6-6-1 Instructions

Instructions are the smallest unit of the processing elements that are provided by OMRON for use in POU algorithms.
Instructions are classified as shown below.



Programs, user-defined functions, and user-defined function blocks consist of these instructions.

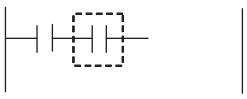
6-6-2 Basic Understanding of Instructions

The fundamental specifications of the instructions follow the specifications of functions and function blocks.
This section describes specifications that are unique to instructions.

Ladder Diagram Structure Elements (Inputs and Outputs)

● Locations

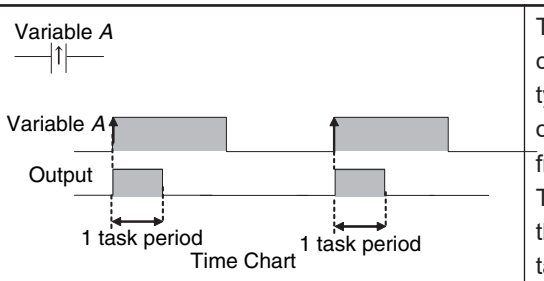
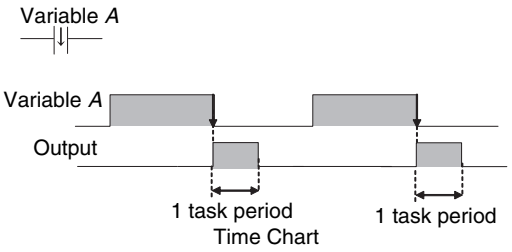
Instructions for ladder diagram inputs and outputs have certain positions where they can be placed, as shown below.

Classification		Locations	Diagram
Input instructions	Logical start	Connected directly to the left bus bar or is at the beginning of an instruction block.	
	Intermediate instructions	Between a logical start and the output instruction.	

Classification	Locations	Diagram
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.
Example:
Execute specifies a change to TRUE or a change to FALSE in the execution condition.
Enable causes the instruction to be executed each task period according to the current execution condition.

Function block instructions are unconditionally executed for as long as the POU that called them is executed.

● Instruction Options

Instruction options cannot be specified.

FUN Instructions

● Execution Conditions

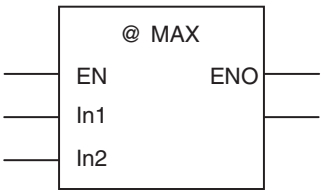
All FUN instructions have *EN* inputs as execution conditions. The FUN instruction is executed each task period as long as *EN* is TRUE.

● Instruction Options

In a ladder diagram, you can add the following instruction options to specify a change to TRUE or a change to FALSE as the execution condition for that instruction. ST statements do not have options.

Instruction options		Symbol	
Differentiation option	Change to TRUE	@	This option creates an upwardly differentiated instruction. The instruction is executed only once when *EN* changes to TRUE.

To add an instruction option, add one of the option symbols listed in the table above before the instruction.
Example:



Information That Applies to Both FB Instructions and FUN Instructions

● Condition Flags

System-defined variables that are assigned values that represent the result of instruction processing are called Condition Flags. The only Condition Flag for an NJ/NX-series Controller is the Carry Flag (P_CY).
The Carry Flag serves the following purposes.

- It shows whether the result of processing an instruction exceeds the range that can be expressed by the data type of the output variable.
- It shows whether an overflow occurred in a bit shift instruction for bit string data. For details, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)*.

6-6-3 Instruction Errors

Instruction errors refer to the errors that occur when an instruction is executed. This section describes when an instruction error occurs, which error is detected as an instruction error, and what operation follow an instruction error, etc.

Timing When Instruction Errors Occur

The timing when instruction errors occur can be divided into the following three cases. Detectable errors and operations following to the errors differ by the timing when instruction errors occur.

- When the values of input parameters or in-out parameters are checked before instruction execution.
- When internal processing is performed during instruction execution.
- When the values of output parameters are checked after instruction execution.

Errors Detected As Instruction Errors

The followings are the errors detected as instruction errors. Different errors are detected depending on the timing when instruction errors occur.

● Errors detected before or after instruction execution

The followings are the errors detected before or after instruction execution.

- Reading or writing an array variable from or to an element beyond the array range.
- Assigning a string that is longer than the defined byte length to a STRING variable.
- Assigning a string that does not end with a NULL character to a STRING variable.
- Dividing an integer variable by 0.

● Errors detected during instruction execution

Errors detected during instruction execution differ by instruction. For details on errors detected in each instruction, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)*.

Operation for Instruction Errors

The operation for the following elements differ depending on whether an instruction error occurs or not: output variable *ENO*, output variable *Error*, output variable *ErrorID*, system-defined variable *P_PRGER*, and events. The details on the operations are described below.

● Output variable *ENO*, output variable *Error*, and output variable *ErrorID*

ENO (enable out), *Error*, and *ErrorID* (error code) are the output variables that indicate whether an error exists or not. Each instruction has different output variables. The meaning of each variable and its value on an instruction error are shown below. The values vary by the timing when an instruction error occurs.

Output variable	Data type	Meaning	Value when an instruction error occurs		
			Before instruction execution*1	During instruction execution	After instruction execution*2
ENO	BOOL	TRUE: Normal end FALSE: Error end, Execution in progress, or Not executed	FALSE	FALSE	TRUE
Error	BOOL	TRUE: Error end FALSE: Error 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 6-140.

● Events

When an instruction error occurs, an event is created for it. For details on events, refer to *Events for Instruction Errors* on page 6-141.

Version Information

A CPU Unit with unit version 1.02 or later is required to create events for instruction errors.

Output Parameters in Ladder Diagrams

The following table shows the values of output parameters when an instruction, user-defined function, or user-defined function block that is created in a ladder diagram ends normally or has an instruction error.

Condition	Type of output parameter	Value of output parameter
Normal end	Power flow output	Values are updated according to the internal algorithm.
	BOOL parameter output	
	Parameter output other than BOOL	
Instruction error	Power flow output	Set to FALSE.
	BOOL parameter output	The previous values are retained.
	Parameter output other than BOOL	

Operation When a Syntax Error Occurs in a POU Written in ST

● Errors in Assignment Statements

When an error occurs in an assignment statement written in ST, that line is not executed.

```
5   a = b / (c + d) + e * f + ABS(g);
6   x := 1;
```

For example, if a division by zero error occurs in (b/(c+d)) on line 5, execution of line 5 is cancelled (the value of a is not changed) and line 6 is executed.

This operation is the same as when the output *ENO* of a user-created function is FALSE.

```
5  a = User-created_function (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.
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.

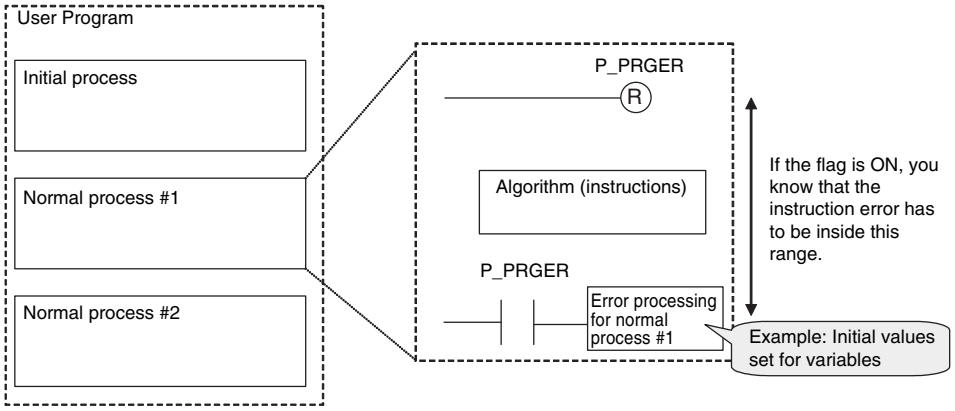
Instruction Error Flag

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.

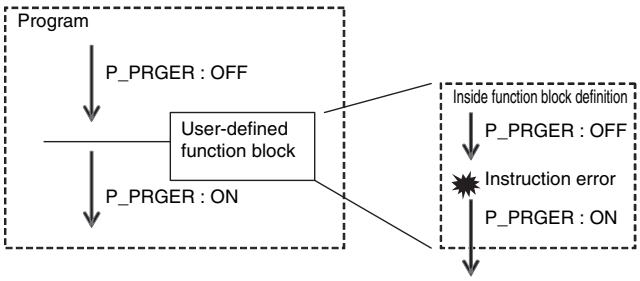
The *P_PRGER* Flag is a local variable for the program. This flag changes to TRUE when an instruction error occurs in the program, and remains TRUE during the next task period.

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 user program changes it back to FALSE.	BOOL	TRUE or FALSE	FALSE	Read/ write

You 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 *8-8 Event Logs* on page 8-81 for the procedure to check events. For information on the events that are created, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)*.

Precautions for Correct Use

- To create events for instruction errors, you must select **Use** for **Event Log Settings** − **Instruction Error Output** on the Sysmac Studio. Refer to *4-2-2 Controller Setup* on page 4-4 and to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for information on the Controller Setup.
- If you change the user program after an instruction error occurs, the information in the event log may no longer be correct.
- If an instruction with an error is executed repeatedly, an instruction error or event is created each time the instruction is executed. This may cause the event log to exceed the maximum number of events. If this occurs, older events are overwritten.

Version Information

- A CPU Unit with unit version 1.02 or later is required to create events for instruction errors.
- A CPU Unit with unit version 1.02 or later and Sysmac Studio version 1.03 or higher are required to specify whether to output instruction errors when they occur.

Additional Information

- If an error occurs in a motion control instruction, two events are created, one for the instruction error and one for the motion control instruction. For details on events for motion control instructions, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.
- Events for motion control instruction are created even if you select **Do not use** for **Event Log Settings** − **Instruction Error Output** in the Controller Setup on the Sysmac Studio.

6-7 Namespaces

This section provides the specifications for namespaces and the procedures to use them. Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for the procedures to manipulate them.

Version Information

A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use namespaces.

6-7-1 Namespaces

Namespaces are a system for grouping function block definitions and other entities to manage them in nested structures. They are similar to grouping files in folders to manage them in a directory structure. If you do not use namespaces, the name of each function block definition or other entity must be unique. If you use namespaces, you can use the same name more than once by setting namespaces. Using namespaces is not required.

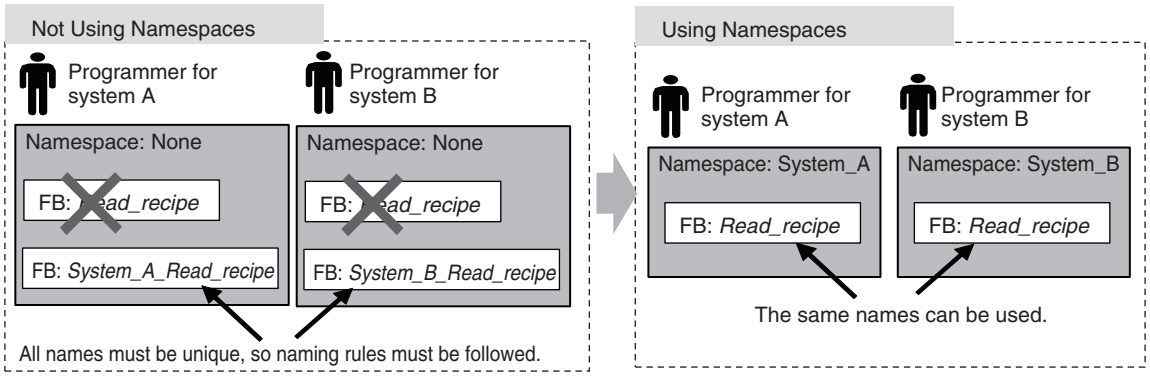
Features of Namespaces

Namespaces provide the following features.

● Preventing Duplicated Names

As long as different namespaces are used, you can use the same name for a function block or other entity more than once.

For example, assume that several systems must be programmed, and that a different programmer will program each of them. Here, it would be likely that the same names would be used for different function block definitions or other entities. If you did not use namespaces, you would have to create naming rules to prevent the duplication of names. However, if you set a different namespace for each system, programming would be possible without worrying about duplicating names with other systems.



6-7-2 Namespace Specifications

This section describes what namespaces can be used for, namespace notation, and namespace declarations.

Namespace Usage

You can use namespaces for the entities that are listed in the following table.

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.

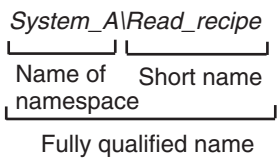
Example:

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:



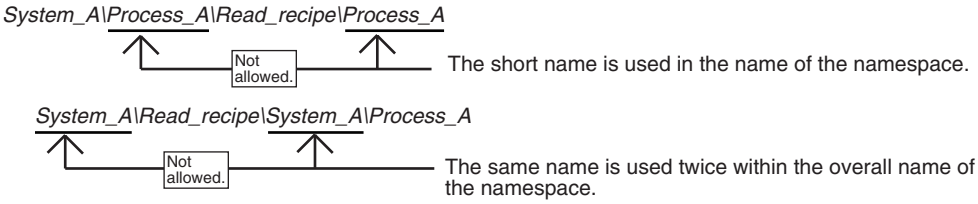
● Restrictions on Namespace Notation

- 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* on page 6-84.
- The following table gives the limits to the number of characters in the names of namespaces.

Name	Maximum size	Character code
Name of namespace	93 bytes	UTF-8
Short name	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.



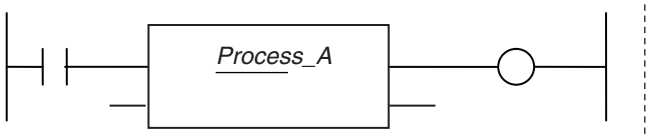
- You can use the short name of a POU definition in the algorithm of a POU definition if it is in the same namespace. However, an error will occur when you build the program if there is a POU definition or data type with the same short name at a higher level in that namespace. For example, assume that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

POU 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.

Example:

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 definition, which is not in the *System_C* namespace. The short name can be given for the *Calculate_upper_limit* function block definition, which is in the *System_C* namespace.

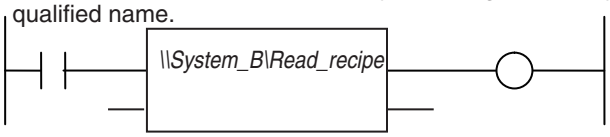
Namespace	Entity	Short name
System_A	Function block definition	*Lifter*
System_B	Function block definition	*Read_recipe*
System_C	Function block definition	*Calculate_upper_limit*

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

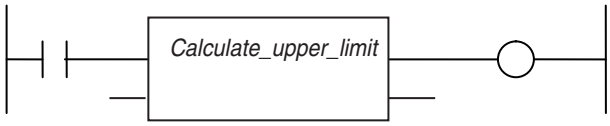
System_C

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)* 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)* for specific procedures.

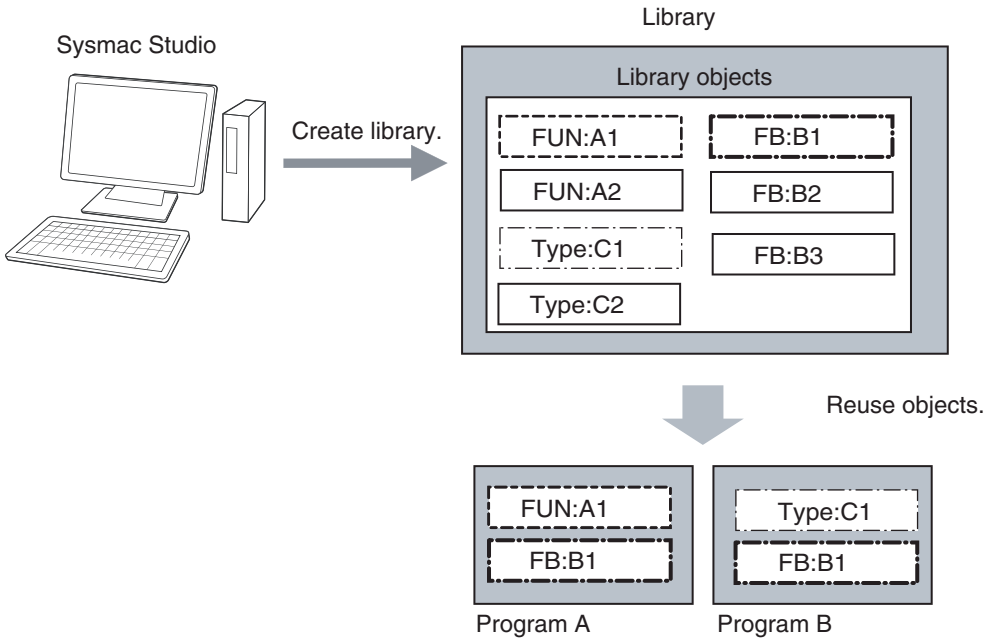
Version Information

A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use libraries.

6-8-1 Introduction to Libraries

A library contains POU definitions and data types in a form that allows you to reuse them as objects in programming. The objects in a library are called library objects.
An NJ/NX-series CPU Unit 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.

Setting	Description
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. *1
Attached files	You can attach one or more files.

*1. 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*1	Structure data types, union data types, and enumeration data types

*1. 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*1	The version of the library object.
Author*1	The creator of the library object. (optional)
Creation date*1	The date that the library object was created.
Update date*1	The date that the object library was last updated.
Comment	A comment on the library object. (optional)

*1. 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* on page 6-142.

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 you use any other functions, function block definitions, or data types.

6-9 Programming Precautions

This section describes precautions for developing a user program.

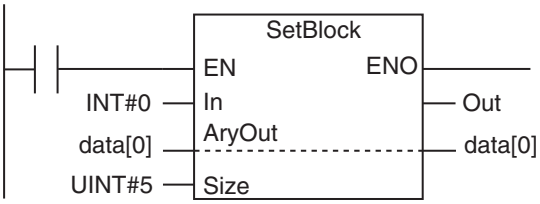
6-9-1 Array Specifications for Input Variables, Output Variables, In-Out Variables

Some instructions handle array variables.

Example:

Data Type Table

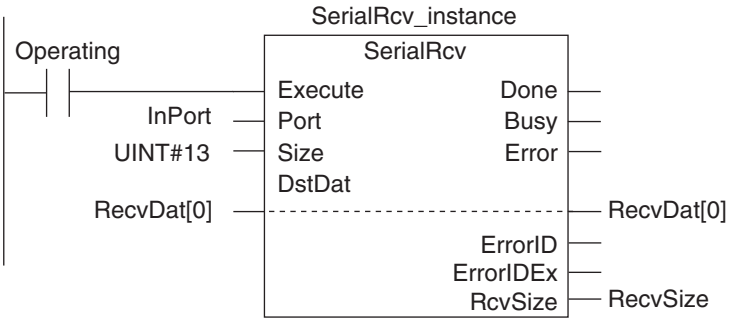
Variable name	Data type
data	ARRAY[0..4] OF INT



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

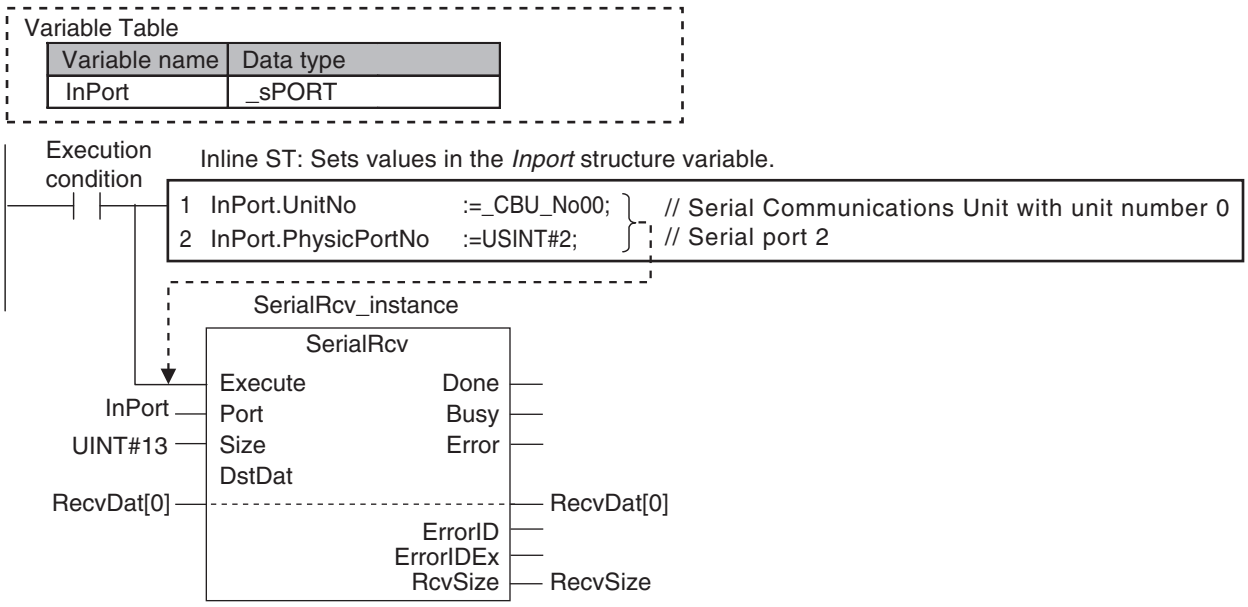
Some instructions have structure variables for input, output, or in-out variables.

Example:



In this case, you must create a structure variable for the input, output, and in-out parameters, then use the MOVE instruction to set the values.

Example:



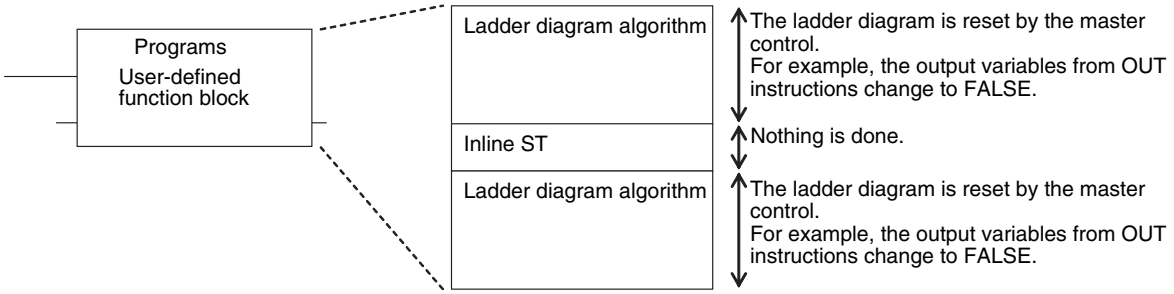
6-9-3 Master Control

Introduction

Master control is used to make output FALSE for all processing between the MC (Master Control Start) instruction and the MCR (Master Control End) instruction.
Master control is useful to control the execution conditions of a relatively long series of instructions.
Refer to information on the MC and MCR instructions in the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for details.

Master Control Programming Languages

You can use master control in ladder diagrams.
You cannot use master control with ST. You also cannot use master control for inline ST inside a ladder diagram.
Example:
Inside a Master Control Region:



Operation of Instructions That Are Reset in a Master Control Region

Refer to information on the MC and MCR instructions in the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for the operation of other instructions in the master control region when master control is reset.

7

Checking Operation and Actual Operation

This section describes the items and procedures for checking the operation of an NJ/NX-series Controller, including offline debugging procedures.

7-1 Overview of Steps in Checking Operation and Actual Operation 7-2

7-2 Offline Debugging 7-3

7-2-1 Features of Simulation 7-3
7-2-2 Simulation Execution 7-3
7-2-3 Setting Up Simulations 7-6

7-3 Checking Operation on the Actual System and Actual Operation 7-8

7-3-1 Checking Operation on the Actual System and Actual Operation 7-8
7-3-2 Downloading the Project 7-8
7-3-3 Checking I/O Wiring 7-9
7-3-4 MC Test Run 7-9
7-3-5 Checking the Operation of the User Program 7-10
7-3-6 Starting Actual Operation 7-11

7-1 Overview of Steps in Checking Operation and Actual Operation

The shaded steps in the overall procedure that is shown below are related to the checking operation and actual operation. In *Step 2-4. Offline Debugging*, a simulation is used to check operation without going online with the Controller. In *Step 5. Checking Operation and Starting Operation on the Actual System*, you go online with the Controller to check the operation of the physical Controller. When checking operation is completed, you start actual operation.
Refer to *1-3 Overall Operating Procedure for the NJ/NX-series* on page 1-23 for the overall procedure.

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

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

Step 3. Mounting and Setting Hardware

Step 4. Wiring

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

7-2 Offline Debugging

This section describes how to use simulation to debug operation offline. You can simulate the operation of an NJ/NX-series Controller on a computer to check the operation of the user program with only the computer. There are also debugging operations that can be used during simulation that are not supported on the physical Controller. This makes user program development and debugging more efficient.

7-2-1 Features of Simulation

In the following way, simulation is more effective than going online with the Controller to debug operation.

- You can use breakpoints, step execution, pausing, and other functions to check program logic.
- You can select only specific programs to simulate to check only those programs.
- You can change the simulation execution speed to check operation at a slower speed than for actual operation.
- You can use the Task Execution Time Monitor to estimate the task execution times.
- You can use debugging programs to manipulate inputs from outside the Controller.

7-2-2 Simulation Execution

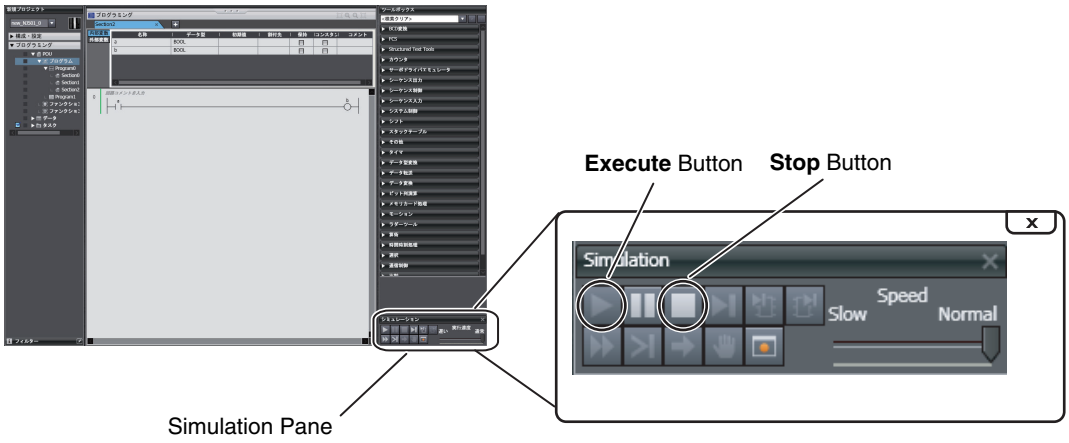
You can do the following for simulations.

- Start and stop the Simulator
- Check the logic of programs.
- Estimate task execution times.
- Use online debugging functions.

Starting and Stopping the Simulator

You perform simulations by starting the Simulator from the Simulation Pane of the Sysmac Studio. After you complete checking operation with the simulation, you stop the Simulator.
The following procedure shows how to start and stop simulations.

1 Select **Simulation Pane** from the View Menu of the Sysmac Studio.
The Simulation Pane is displayed on the lower right of the window.



2 Click the **Execute** Button in the Simulation Pane.
The user program is transferred to the Simulator and the simulation starts. When a simulation starts, the Editors and other parts of the Sysmac Studio window will enter the same state as when the Sysmac Studio is online with the Controller.

3 After you complete checking operation, click the **Stop** Button in the Simulation Pane to stop the Simulator.

Checking the Logic of Programs

You can use simulation debugging to stop the operation of the Simulator or to execute a program one step at a time to check the validity of the program logic. You can perform the following operations with the buttons in the Simulation Pane.

Operation	Description of operation
Breakpoints	Use a breakpoint to specify a location in a program and pause program execution at that location.
Step execution	Use step execution to execute one line of an ST program or one instruction in a ladder diagram program and then pause the Simulator.
Continuous step execution	Use continuous step execution to continually perform step execution at a specified interval.
Pausing	Use pausing to pause execution of the simulation.
Step-in execution	Use step-in execution to perform step execution of source code inside a function or function block.
Step-out execution	Use step-out execution to execute the current function or function block to the end.
One-period execution	Use one-period execution to execute the current task for one period. Execution pauses at the beginning of the program in the next period.
Conditional breakpoints	Use conditional breakpoints to pause the execution of a program at a breakpoint when the specified stopping condition is met.

Estimating Task Execution Times

If you execute the Simulator in Execution Time Estimation Mode, the estimated task execution time from when task execution starts until it stops is displayed on the Task Execution Time Monitor. The

average and maximum estimated task execution times are displayed. Refer to *Task Execution Time Monitor* on page 5-128 for the Task Execution Time Monitor.

Precautions for Correct Use

- Select the relevant hardware revision in the Unit that the hardware revision is displayed.
- The estimated task execution times are not necessarily the same as the actual task execution times on the physical Controller. Depending on the user program, I/O configuration, and whether communications are used, the execution times on the physical Controller may exceed the estimated maximum value. Use them only as guidelines in task design. Always confirm the task execution times while connected to the physical Controller to study the designs and before starting actual system operation.

Online Debugging Functions

With the Simulator, you can use some of the functions for debugging that are supported when you are online with the Controller. The following table shows the differences between online debugging with the Controller or offline debugging with the Simulator. Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on the differences in debugging operations for the Controller and for the Simulator.

(○: Supported, ×: Not supported)

Debugging function	Controller	Simulator
Monitoring	○	○
Monitoring in a Watch Tab Page	○	○
Monitoring in the I/O Map	○	○
Differential monitoring	○	○
Controlling BOOL variables	○	○
Forced Refreshing (TRUE/FALSE/Cancel)	○	○
Changing present values of data	○	○
Clearing all memory	○	×
Cross-reference pop-ups	○	○
Online editing	○	○
Monitoring Controller status	○	×
Monitoring task execution status	○	○
Monitoring axis status (MC Monitor Table)	○	○
Changing the operating mode	○	×
Resetting the Controller	○	×
Data tracing	○	○
Setting triggers	○	○
Setting variables to sample	○	○
Starting and stopping tracing	○	○
Displaying trace results	○	○
Exporting trace results	○	○
Creating 3D equipment models	○	○
Displaying digital and analog charts	○	○
Displaying 3D axis paths	○	○
Monitoring task execution times	○	○
Estimating execution processing times	×	○

Debugging function	Controller	Simulator
Debugging with program simulations	×	○
Setting simulation programs	×	○
Changing the simulation speed	×	○
Setting breakpoints	×	○
Step execution	×	○
Troubleshooting	○	○
Monitoring error information	○	○
Displaying error logs	○	○
Setting event tables	○	○
Monitoring user memory usage	○	○
Clock information settings	○	×
Releasing access rights	○	×

7-2-3 Setting Up Simulations

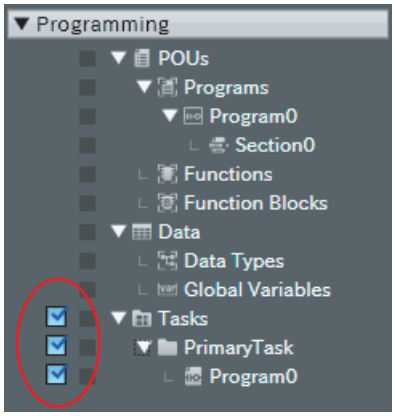
You set the following for simulations.

- 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 Multi-view Explorer 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.

4. 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 Checking Operation on the Actual System and Actual Operation

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. *1 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)*

*1. You cannot connect a computer to an NX701 CPU Unit with hardware revision A or later, NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit via USB, because it does not provide a peripheral USB port. Refer to *10-2 Connection with Sysmac Studio* on page 10-9 for the procedure to connect to the Sysmac Studio.

Step 2. Checking Operation on the Controller	Reference
1. In PROGRAM mode, check the I/O wiring by using forced refreshing of real I/O 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* on page 7-9 *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* *7-3-5 Checking the Operation of the User Program* on page 7-10

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.

Precautions for Correct Use

For CPU Units with unit version 1.40 or later, the Controller reset is required in any of the following cases. Transfer the project according to the message on the Sysmac Studio.
- When an attempt is made to transfer a project with project unit version 1.40 or later to a CPU Unit where a project with project unit version earlier than 1.40 is stored.
- When an attempt is made to transfer a project with project unit version earlier than 1.40 to a CPU Unit where a project with project unit version 1.40 or later is stored.
- After the Clear All Memory operation is performed on a CPU Unit where a project with project unit version 1.40 or later is stored, without executing the Controller reset, an attempt is made to transfer a project with project unit version earlier than 1.40.

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 *8-7-1 Forced Refreshing* on page 8-61 for information on forced refreshing.

7-3-4 MC Test Run

The MC Test Run function is used mainly to perform the following operations from the Sysmac Studio without a user program.
- Checking wiring: You can monitor Servo Drive connector I/O signals and Servo Drive status.
- Checking the operation and direction of the motor: You can turn ON the Servo and jog axes.
- Checking electronic gear settings: You can perform relative positioning, and check and change travel distances.
- Checking homing: You can check the homing operation.

Connect online to the Controller from the Sysmac Studio and perform the MC Test Run on the MC Test Run Tab Page.
For details, refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)*.

Use the following procedure.

1 After you complete the necessary wiring, connect the Sysmac Studio online to the Controller.

2 Create axes, assign the axes, and set the following axis parameters.
Axis parameter settings required for an MC Test Run: Unit of Display, Command Pulses Per Motor Rotation, Travel Distance Per Motor Rotation, Maximum Velocity, Maximum Jog Velocity, Maximum Acceleration Rate, Maximum Deceleration Rate, Software Limit Function Selection, Software Limits, and Count Mode

3 Open the MC Test Run Tab Page and perform the following.
Example:

- Monitoring and checking wiring
- Jogging to check the direction of the motor
- Check travel distances for relative positioning (electronic gear settings).
- Confirming the homing operation

7-3-5 Checking the Operation of the User Program

To check the operation of the user program on the actual system, change the operating mode of the CPU Unit to RUN mode. You can use the following to check operation.

- Checking the operation of the user program
- Correcting the user program with online editing
- Checking the operation of the user program with data tracing

Checking the Operation of the User Program

You can perform the following to check the operation of the user program.

1 Monitor the execution status of the user program.

2 Check the operation by changing the status of program inputs and program outputs, and the values of variables.

● Monitoring the Execution Status of the User Program

You can monitor the TRUE/FALSE status of program inputs and outputs and the present values of variables in the Controller. You can monitor the status on the Ladder Editor, Watch Tab Page, or I/O Map of the Sysmac Studio.

● Checking the Operation by Changing the Status of Program Inputs and Program Outputs, and the Values of Variables

You can change the TRUE/FALSE status of program inputs and outputs and the present values of variables in the user program to see if the user program operates as designed. Use forced refreshing to change the status of program inputs and program outputs. Use one of the methods to change the present values of variables.

Refer to *8-7-1 Forced Refreshing* on page 8-61 and *8-7-2 Changing Present Values* on page 8-65 for details.

Correcting the User Program with Online Editing

You can use online editing to correct a user program that you determined needs to be corrected while checking operation. You can use online editing to change a user program without stopping the operation of the CPU Unit.

Refer to *8-7-3 Online Editing* on page 8-67 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 *8-7-4 Data Tracing* on page 8-68 for details.

7-3-6 Starting Actual Operation

Change the operating mode to RUN mode to start actual operation. Check the user program, data, and parameter settings sufficiently for proper execution before you use them for actual operation.

8

CPU Unit Functions

This section describes the functionality provided by the CPU Unit.

8-1 Data Management, Clock, and Operating Functions 8-3
8-1-1 Clearing All Memory 8-3
8-1-2 Clock 8-3
8-1-3 RUN Output 8-6

8-2 Management Functions for NX Units 8-7
8-2-1 NX Bus Function Module 8-7
8-2-2 Mounting Settings of NX Units on the CPU Unit 8-11
8-2-3 Restarting NX Units on the CPU Unit 8-14
8-2-4 Checking Wiring for NX Units on the CPU Unit 8-15
8-2-5 Fail-soft Operation for NX Units on the CPU Unit 8-16
8-2-6 Monitoring Total Power-ON Time for NX Units on the CPU Unit 8-19
8-2-7 Clearing All Memory for Each NX Unit on the CPU Unit 8-20

8-3 Management Function for X Bus Units 8-22
8-3-1 X Bus Function Module 8-22
8-3-2 X Bus Unit Common Function Module 8-23
8-3-3 X Bus Unit Mounting Settings 8-24
8-3-4 X Bus Unit Setting Synchronization Function 8-26
8-3-5 Clearing All Memory for Each X Bus Unit 8-26

8-4 Management Functions for CJ-series Units 8-27
8-4-1 Basic I/O Units 8-27
8-4-2 Special Units 8-28

8-5 SD Memory Card Operations 8-30
8-5-1 SD Memory Card Operations 8-30
8-5-2 Specifications of Supported SD Memory Cards, Folders, and Files 8-31
8-5-3 SD Memory Card Operation Instructions 8-34
8-5-4 FTP Client Communications Instructions 8-34
8-5-5 FTP Server 8-35
8-5-6 File Operations from the Sysmac Studio 8-35
8-5-7 SD Memory Card Life Expiration Detection 8-35
8-5-8 List of System-defined Variables Related to SD Memory Cards 8-36
8-5-9 SD Memory Card Self-diagnostic Functions 8-37
8-5-10 Exclusive Control of File Access in SD Memory Cards 8-39

8-6 Security 8-40
8-6-1 Authentication of User Program Execution IDs 8-41
8-6-2 User Program Transfer with No Restoration Information 8-44
8-6-3 Overall Project File Protection 8-45
8-6-4 Data Protection 8-46

8-6-5 Secure Communication Function ... 8-48
8-6-6 Secure Socket Services ... 8-48
8-6-7 TCP/UDP Port Close Function ... 8-49
8-6-8 Packet Filter ... 8-49
8-6-9 Complete Data Erasure Function ... 8-49
8-6-10 Operation Authority Verification ... 8-51
8-6-11 User Authentication ... 8-53
8-6-12 CPU Unit Write Protection ... 8-57
8-6-13 Firmware Update Prohibition ... 8-58
8-6-14 CPU Unit Names and Serial IDs ... 8-59
8-6-15 Access log ... 8-60

8-7 Debugging ... 8-61
8-7-1 Forced Refreshing ... 8-61
8-7-2 Changing Present Values ... 8-65
8-7-3 Online Editing ... 8-67
8-7-4 Data Tracing ... 8-68
8-7-5 Differential Monitoring ... 8-75
8-7-6 Automation Playback ... 8-80

8-8 Event Logs ... 8-81
8-8-1 Introduction ... 8-81
8-8-2 Detailed Information on Event Logs ... 8-83
8-8-3 Controller Events (Controller Errors and Information) ... 8-87
8-8-4 User-defined Events (User-defined Errors and Information) ... 8-88

8-9 Changing Event Levels ... 8-95
8-9-1 Applications of Changing Event Levels ... 8-95
8-9-2 Events for Which the Event Level Can Be Changed ... 8-95
8-9-3 Procedure to Change an Event Level ... 8-95

8-10 Displaying Firmware Update Log ... 8-98

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 offset in the CPU Unit to the defaults from the Sysmac Studio. This is called the *Clear All Memory* operation.

Precautions for Correct Use

- The Clear All Memory operation can be performed only in PROGRAM mode.
- You cannot execute the *Clear All Memory* operation when write protection of the CPU Unit is set in the security functions.
- Do not turn OFF the power supply to the CPU Unit during the Clear All Memory operation.
- The client private key, client certificate and secure socket setting for the secure socket communications are out of the target of the Clear All Memory operation.
  Use the Secure Socket Configuration commands to initial the client private key, client certificate and secure socket setting that are used for the secure socket communications.

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

● Operations from the Sysmac Studio

Connect the Sysmac Studio to the CPU Unit online, and select the **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

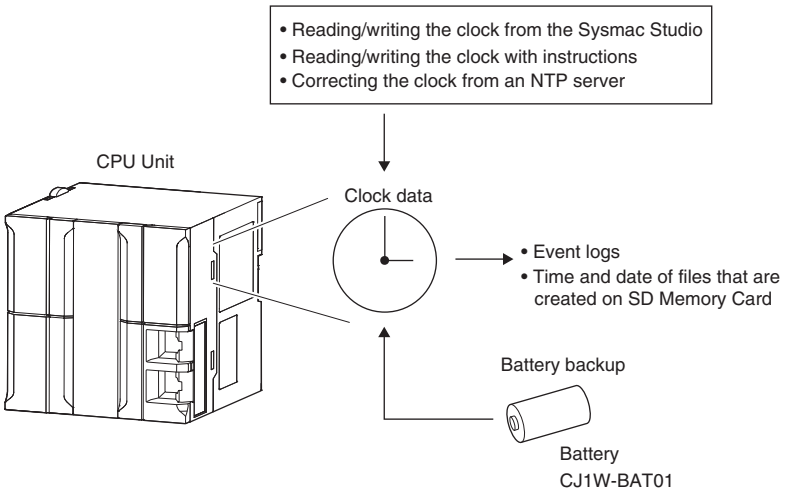
Overview

A clock (RTC) is built into the CPU Unit.

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

The following functions are supported.

- Reading/writing the clock from the Sysmac Studio
- Reading/writing the clock with instructions
- Reading the clock from system-defined variables (Writing is not possible.)
- Correcting the clock from an NTP server



Precautions for Correct Use

- The clock data is retained by the Battery when the power is turned OFF. If the battery is not installed, the clock data is not correct when the power is turned ON. If the battery is not installed, reset the clock data from an NTP server over an EtherNet/IP network after the power is turned ON.
- The clock data is retained by a built-in capacitor when the battery-free operation is used in the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units. When the power of the device is turned OFF if the retention time in the built-in capacitor is exceeded, the clock data are initialized. If you use the clock data in the event log and other functions, specify the clock data when you turn ON the power supply every time.

● Clock Data Range

- NX-series CPU Units:1970-01-01 to 2069-12-31 (January 1, 1970 to December 31, 2069).
- NJ-series CPU Units:1970-01-01 to 2106-02-06 (January 1, 1970 to February 6, 2106).

● Setting the Time Zone and the Local Time

Before you use the Controller for the first time, set the time zone and local time in the clock data. You can set the time zone and local time from the Sysmac Studio in the Controller Clock Dialog Box.

The clock data that is read by the EtherCAT slaves and CJ-series CPU Units from the CPU Unit and the clock data that is set are the local times in the time zone.

Additional Information

When a Battery is not mounted or when the Battery voltage is low, the time zone setting is retained, but the clock data is not retained and will not be correct.

Setting the Clock Data

Use one of the following methods.

● Changing Clock Data from the Sysmac Studio

You can use the Sysmac Studio to synchronize the clock data of the built-in clock with the clock on the computer.

● Changing Clock Data with Instructions

You can use the SetTime instruction to set the clock data.

● Changing the Clock Data from an NTP Server

You can use an NTP server on EtherNet/IP to set the clock data.

Correcting the Clock from an NTP Server

● Application

In a network system, the clock data must be shared by the entire system. NTP is supported to enable easy time synchronization.

● Specifications

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

Reading the Clock Data

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

● Reading the Clock Data from Instructions

You can use the GetTime instruction to read the clock data from the user program.

● Reading the Clock from System-defined Variables (Writing Is Not Possible)

You can use the following system-defined variable to read the clock data.
_CurrentTime (System Time)

● Sysmac Studio Procedure

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

Logging

When you change the clock data, an event is recorded in the event log.
However, nothing is recorded in the event log if the time is corrected for the NTP.

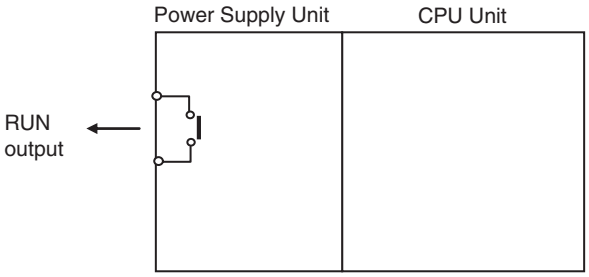
Related System-defined Variables

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

8-1-3 RUN Output

Introduction

The RUN output on NX-PA9001, NX-PD7001, or NJ-P□3001 Power Supply Unit is ON while the CPU Unit is operating.



The RUN output operates as shown in the following table.

Status	Operation
During RUN mode	ON
Startup state (until RUN mode is entered according to the Startup Mode setting)	OFF
During PROGRAM mode	
When a major fault level Controller error occurs	

The ratings of the RUN output on NX-PA9001, NX-PD7001, or NJ-P□3001 Power Supply Unit are as follows:

Item	Description
Contact form	SPST-NO
Switching capacity	2 A at 250 VAC for resistive load 0.5 A at 120 VAC for inductive load 2 A at 24 VDC for resistive load

Application

- Obtain a signal to notify the host that the CPU Unit is functioning normally and is currently operating.
- Synchronize the completion of startup of more than one CPU Unit
- Release interlocks when the CPU Unit starts operation.

Precautions for Safe Use

It takes up to approximately 10 to 20 seconds to enter RUN mode after the power is turned ON. The outputs during this time behave according to the slave or Unit specifications.
Use the RUN output on the Power Supply Unit, for example, to implement fail-safe circuits so that external devices do not operate incorrectly.

8-2 Management Functions for NX Units

This section describes the management functions used for NX Units on the NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit.
You can use NX Units only on the NX502, NX102, and NX1P2 CPU Units.

8-2-1 NX Bus Function Module

The NX Bus Function Module performs processing such as a management of event logging, management of status, and I/O refreshing for the NX Units that are connected to the NX bus of the CPU Unit as a master of the NX bus (hereafter NX bus master).

For I/O data that are handled by the NX Bus Function Module, there are two kinds of I/O data, the status of NX Units managed by the NX Bus Function Module as the NX bus master and I/O data for individual NX Units. The variables are the assignable I/O ports for both of them.

There are two kinds of variables to access I/O data, device variables that are assigned to I/O ports and system-defined variables for the NX Bus Function Module.

The following describes the status of NX Units managed by the NX Bus Function Module as the NX bus master, I/O data for individual NX Units, assigning device variables to I/O ports, and programming sample using device variables.

Status of NX Units Managed by the NX Bus Function Module as the NX Bus Master

For the status of NX Units managed by the NX Bus Function Module as the NX bus master, you can use only device variables assigned to I/O ports to access, only system-defined variables to access, or both of device variables and system-defined variables to access.

● A List of Status of NX Units Managed by the NX Bus Function Module as the NX Bus Master

Name	I/O port	System-defined variable
NX Unit Registration Status	NX Unit Registration Status	_NXB_UnitRegTbl
NX Unit Message Enabled Status	NX Unit Message Enabled Status	_NXB_UnitMsgActiveTbl
NX Unit I/O Data Active Status	NX Unit I/O Data Active Status	_NXB_UnitIOActiveTbl
NX Unit Error Status	NX Unit Error Status	_NXB_UnitErrFlagTbl
Time Stamp of Synchronous Input	Time Stamp of Synchronous Input	---
Time Stamp of Synchronous Output	TimeStamp of Synchronous Output	---
NX Bus Function Module Error Status	---	_NXB_ErrSta
NX Bus Function Module Master Error Status	---	_NXB_MstrErrSta
NX Bus Function Module Unit Error Status	---	_NXB_UnitErrStaTbl

● Descriptions of Status of NX Units Managed by the NX Bus Function Module as the NX Bus Master

Name	Description
NX Unit Registration Status	This status tells whether the NX Units are registered in the Unit configuration. Each bit has the following meaning. TRUE: Registered FALSE: Not registered If the Unit configuration information is registered, the status is TRUE for each Unit that is registered. If the Unit configuration information was automatically created (with only the actual Unit configuration information and no registered information), the status is FALSE for all Units. The status is TRUE for NX Units that are set as unmounted Units. Each bit is updated at the following times. • If the Unit Configuration Information Is Registered: The status changes to TRUE when the system is started. The status changes to FALSE when the configuration information is cleared. • If the Unit Configuration Information Is Automatically Created: The status changes to TRUE when the configuration information is confirmed. The status is always FALSE if the Unit configuration information is automatically created.
NX Unit Message Enabled Status	This status tells whether the NX Units can process message communications. Each bit has the following meaning. TRUE: Message communications possible. FALSE: Message communications not possible. The status says that message communications are enabled for NX Units that meet the following conditions. • The comparison shows no differences (only if the Unit configuration information is registered). • The NX Unit does not have a WDT error. The status is FALSE for NX Units that are set as unmounted Units. Each bit is updated when the message communications status changes on the corresponding NX Unit.
NX Unit I/O Data Active Status	This status tells whether the NX Units can process I/O data communications. Each bit has the following meaning. TRUE: The I/O data in the NX Unit can be used for control. FALSE: The I/O data in the NX Unit cannot be used for control. The status is FALSE for NX Units that are set as unmounted Units. Each bit is updated when the operating status changes on the corresponding NX Unit. If both of NX Unit Registration Status and NX Unit I/O Data Active Status are TRUE, the target NX Units operate normally.

Name	Description
NX Unit Error Status	This status tells whether an error exists on the NX Units. Each bit has the following meaning. TRUE: Error FALSE: No error Each bit is set to TRUE when the level of the error is as follows: • Major fault • Partial fault • Minor fault • Observation The status is FALSE for NX Units that are set as unmounted Units. Each bit is updated at the following times. The status changes to TRUE when an error occurs. The status changes to FALSE when the error is reset. Even if the cause of the error has been removed, you must reset the error for the status to change to FALSE.
Time Stamp of Synchronous Input	This time stamp tells when a synchronous input occurred in the NX Unit that supports synchronous I/O refreshing. The unit is ns.
Time Stamp of Synchronous Output	This time stamp tells when a synchronous output occurred in the NX Unit that supports synchronous I/O refreshing. The unit is ns.
NX Bus Function Module Error Status	Gives the NX Bus Function Module error status. This status collects the NX Bus Function Module Master Error Status and NX Bus Function Module Unit Error Status for all NX Units.
NX Bus Function Module Master Error Status	Gives the status of errors that are detected in the NX Bus Function Module of the CPU Unit. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits.
NX Bus Function Module Unit Error Status	Gives the status of errors that are detected in the NX Unit on the CPU Unit. This status is given as an array of WORD data. The subscript of the array corresponds to the NX Unit number. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits.

Refer to *A-7-5 NX Bus Function Module, Category Name: _NXB* on page A-111 for details on the related system-defined variables.

● Update Timing of Status of NX Units Managed by the NX Bus Function Module as the NX Bus Master

Name	I/O port for an NX Unit	System-defined variable
NX Unit Registration Status	Not synchronized with task execution	Not synchronized with task execution
NX Unit Message Enabled Status	Not synchronized with task execution	Not synchronized with task execution
NX Unit I/O Data Active Status	Synchronized with primary periodic task	Synchronized with primary periodic task
NX Unit Error Status	Synchronized with primary periodic task	Synchronized with primary periodic task
Time Stamp of Synchronous Input	Synchronized with primary periodic task	---
Time Stamp of Synchronous Output	Synchronized with primary periodic task	---
NX Bus Function Module Error Status	---	Not synchronized with task execution
NX Bus Function Module Master Error Status	---	Not synchronized with task execution
NX Bus Function Module Unit Error Status	---	Not synchronized with task execution

I/O Data for Individual NX Units

I/O data are determined by the model number of the NX Unit and the functionality. You can use only device variables that are assigned to an I/O port of an NX Unit to access I/O data.

Refer to the user's manual for the specific NX Units for details on I/O data for individual NX Units.

Assigning Device Variables to I/O Ports

When you create the Unit configuration information on the Sysmac Studio, the status of NX Units managed by the NX Bus Function Module as the NX bus master and I/O data for NX Units mounted on the CPU Unit are automatically registered as I/O ports.
The variables that are assigned to I/O ports for status and I/O data are device variables.

● I/O Port Names

The status of NX Units managed by the NX Bus Function Module as the NX bus master is given as the following six kinds of I/O port names for each NX Unit.

Name	I/O port name	Data type
NX Unit Registration Status	Device name + NX Unit Registration Status	BOOL
NX Unit Message Enabled Status	Device name + NX Unit Message Enabled Status	BOOL
NX Unit I/O Data Active Status	Device name + NX Unit I/O Data Active Status	BOOL
NX Unit Error Status	Device name + NX Unit Error Status	BOOL
Time Stamp of Synchronous Input	Device name + Time Stamp of Synchronous Input	ULINT
Time Stamp of Synchronous Output	Device name + TimeStamp of Synchronous Output	ULINT

Example for NX Unit Registration Status with a device name N1:
N1 NX Unit Registration Status
Example for Time Stamp of Synchronous Input with a device name N2:
N2 Time Stamp of Synchronous Input

I/O port names are determined by the model number of the NX Unit and the functionality for I/O data for NX Units mounted on the CPU Unit.

Example for a Digital Input Unit:
Input Bit 00
Example for an Analog Output Unit:
Ch1 Analog Output Value

● Registering Device Variables

You assign device variables to I/O ports in the I/O Map of the Sysmac Studio. The device variables that you create are registered in the variable table.
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

Refer to *Device Variable Attributes* on page 3-12 for details on device variable attributes.

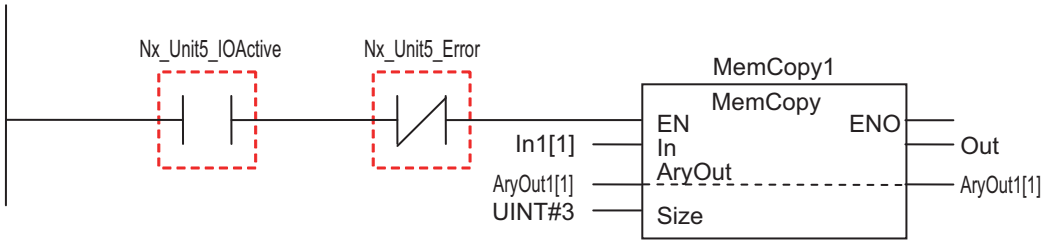
Programming Sample Using Device Variables

The following describes a programming sample when using the NX Unit device variables.

● Testing the Validity of I/O Data for Individual NX Units

A sample programming that determines whether NX Unit (NX Unit number: 5) I/O data on the CPU Unit is valid is given below. The NX Unit I/O Data Active Status and the NX Unit Error Status are used.

The I/O data is manipulated in the NX Unit (NX Unit number: 5) if I/O data is valid.

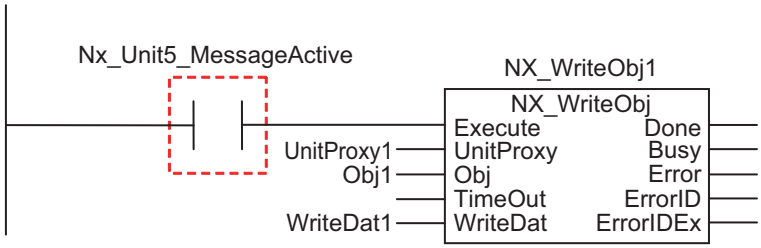


Nx_Unit5_IOActive: This is a device variable to indicate the NX Unit I/O Data Active Status with NX Unit number 5.

Nx_Unit5_Error: This is a device variable to indicate the NX Unit Error Status with NX Unit number 5.

● Testing Whether Individual NX Units Can Process Message Communications

A sample programming that confirms whether NX Unit (NX Unit number: 5) on the CPU Unit can process message communications is given below. The NX Unit Message Enabled Status is used. The Write NX Unit instruction is used to send a message to the NX Unit (NX Unit number: 5) if message communications is enabled.



Nx_Unit5_MessageActive: This is a device variable to indicate the NX Unit Message Enabled Status with NX Unit number 5.

8-2-2 Mounting Settings of NX Units on the CPU Unit

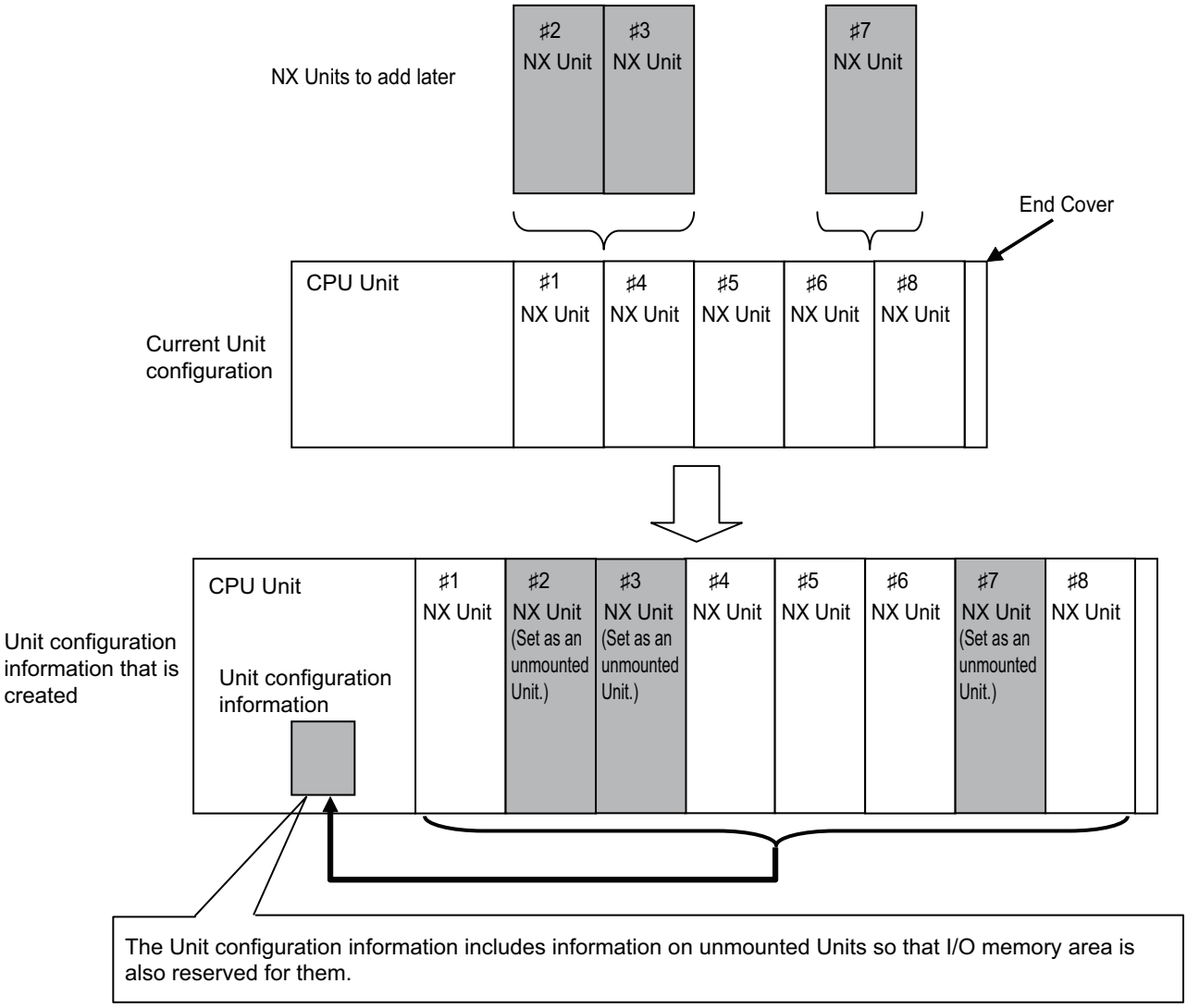
This section describes the mounting settings of NX Units on the CPU Unit.

Overview of Function

You can use this function to register NX Units that will be added to at a later time in the Unit configuration information as unmounted Units. With this function, you can create the program in advance for NX Units that are not mounted to the actual configuration.

You can use this function even if a specific Unit is temporarily unavailable such as when commissioning the system.

- I/O memory area is reserved for these unmounted NX Units in the same way that it is reserved for NX Units that are mounted to the actual configuration.
- Unmounted NX Units are also assigned NX Unit numbers. This prevents the NX Unit numbers of other NX Units on the CPU Unit from changing when you change the setting of an NX Unit that is not mounted to the setting of an NX Unit that is mounted to the actual configuration.



Operating Specifications for NX Units That Are Set as Unmounted Units

The operating specifications for NX Units that are set as unmounted Units are given in the following table.

Item	Operation
Bandwidth reservation for I/O refresh data with the EtherCAT master	Bandwidth is reserved. The input value is always 0.
I/O refreshing with the EtherCAT master	The I/O is not refreshed.
Detection of events	Events are not detected.
Assignment of NX Unit numbers to NX Units	Unit numbers are not assigned because the Units do not exist.

Item	Operation
Message communications	Not possible because the Units do not exist.
Transfers for the synchronization function of the Sysmac Studio	Not applicable.
Transfer of the Unit operation settings	Not applicable.
Sysmac Studio Controller backup function	Not applicable.
SD Memory Card backup function	Not applicable.
Instructions	Parameters cannot be read or written. An instruction error will occur.
Clearing all memory	Not applicable.
Reading/writing setting information through backup/restore operations	Not applicable.
Reading event logs	Not applicable.
Notification of status information	Not applicable.

NX Units that are set as unmounted Units are included in the calculations for total power consumption and total Unit width when the Unit configuration is created on the Sysmac Studio.

Precautions for Safe Use

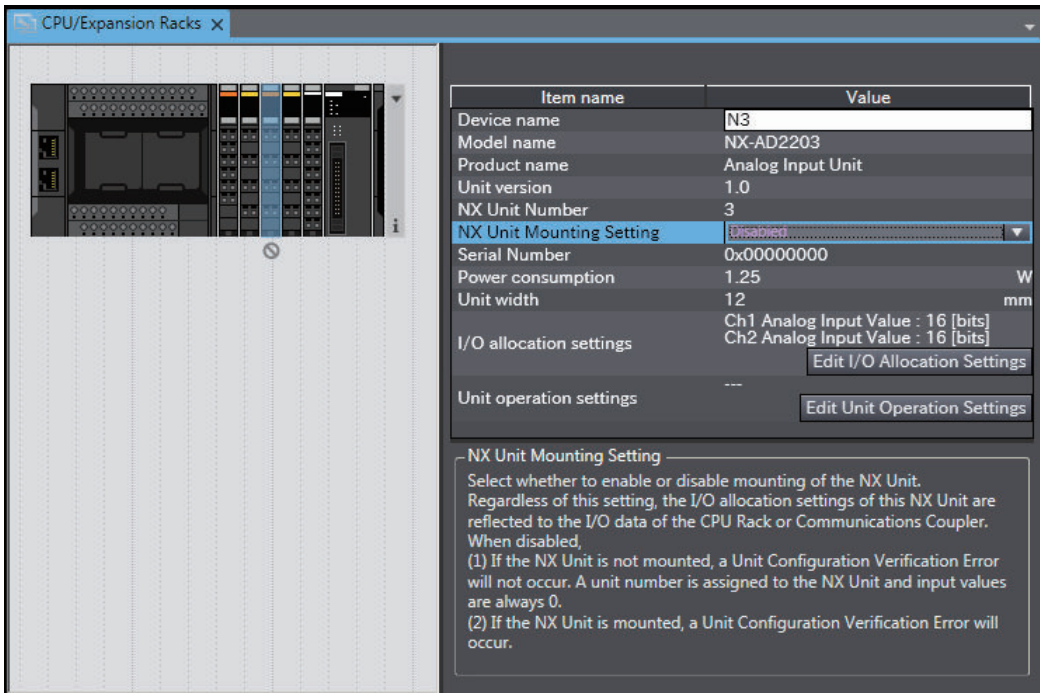
Check the user program, data, and parameter settings for proper execution before you use them for actual operation.

Precautions for Correct Use

When you mount an NX Unit that was set as an unmounted Unit, a Unit Configuration Verification Error will occur.

Setting NX Units as Unmounted Units

In the CPU and Expansion Racks Tab Page on the Sysmac Studio, select the target NX Unit and set the **NX Unit Mounting Setting** to **Disabled**. The selected NX Unit is set as an unmounted Unit. After you change the settings for any NX Units, always transfer the Unit configuration information to the CPU Unit.



8-2-3 Restarting NX Units on the CPU Unit

This section describes restarting an NX Unit on the NX1P2 CPU Unit. The restart function is used to enable values that are set for the NX Unit without cycling the power supply to the Controller.

Types of Restarts

The following table gives the types of restarts for individual NX Units.

Type	Function
Restarting NX Bus Function Module	All NX Units on the CPU Unit are restarted.
Restarting Individual NX Units	The specified NX Unit is restarted.

Restarting NX Bus Function Module

All NX Units on the CPU Unit are restarted.
Specify the NX Bus Function Module and then execute the restart. The restart of the NX Bus Function Module does not affect the task execution of the CPU Unit. Also an error will not occur.

The methods for restarting are listed below.

- Sysmac Studio
- RestartNXUnit (Restart NX Unit) instruction

Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for details on the RestartNXUnit instruction.

Restarting Individual NX Units

One specified NX Unit is restarted.
The methods for restarting an NX Unit are listed below.

- Sysmac Studio
- RestartNXUnit (Restart NX Unit) instruction

Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for details on the RestartNXUnit instruction.
The NX Bus Function Module and all NX Units that were not specified for restarting continue to operate.

Restarting an NX Unit

Place the Sysmac Studio online. In the CPU and Expansion Racks Tab Page, right-click the CPU Unit and select **Restart for NX Bus/NX Unit** – **Yes**. All NX Units on the CPU Unit are restarted.
In the same way, right-click the NX Unit to restart and select **Restart for NX Bus/NX Unit** – **Yes**. The specified NX Unit is restarted.

8-2-4 Checking Wiring for NX Units on the CPU Unit

This section describes how to check the wiring for NX Units on the NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit.

You can use the Sysmac Studio to check the wiring between NX Units on the CPU Unit and I/O devices during system commissioning even if the user program is not created.

Preparation

Create the NX Unit configuration information and check whether the Unit configuration on the Sysmac Studio agrees with the actual Unit configuration if the NX Unit configuration information was transferred to the CPU Unit.
If the NX Unit configuration information was not transferred to the CPU Unit, go online, right-click anywhere in the CPU and Expansion Racks Tab Page and select **Compare and Merger with Actual Unit Configuration**. The configuration is automatically configured.

How to Check the Wiring

Item to check	Checking method
Checking inputs	Place the Sysmac Studio online and you can monitor the present values of Unit I/O ports in the I/O Map.
Checking outputs	Place the Sysmac Studio online, and forced refreshing or set/reset Unit I/O ports in the I/O Map.

Refer to *8-7-1 Forced Refreshing* on page 8-61 for information on forced refreshing.

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

Additional Information

- For Position Interface Units and other NX Units that are assigned to axes, use the MC Test Run of the Sysmac Studio. Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for details.
- Use the I/O checking of EtherCAT Slave Terminals to check the wiring of NX Units on the EtherCAT Coupler Unit. Refer to the *NX-series EtherCAT Coupler Units User's Manual* (Cat. No. W519) for details.

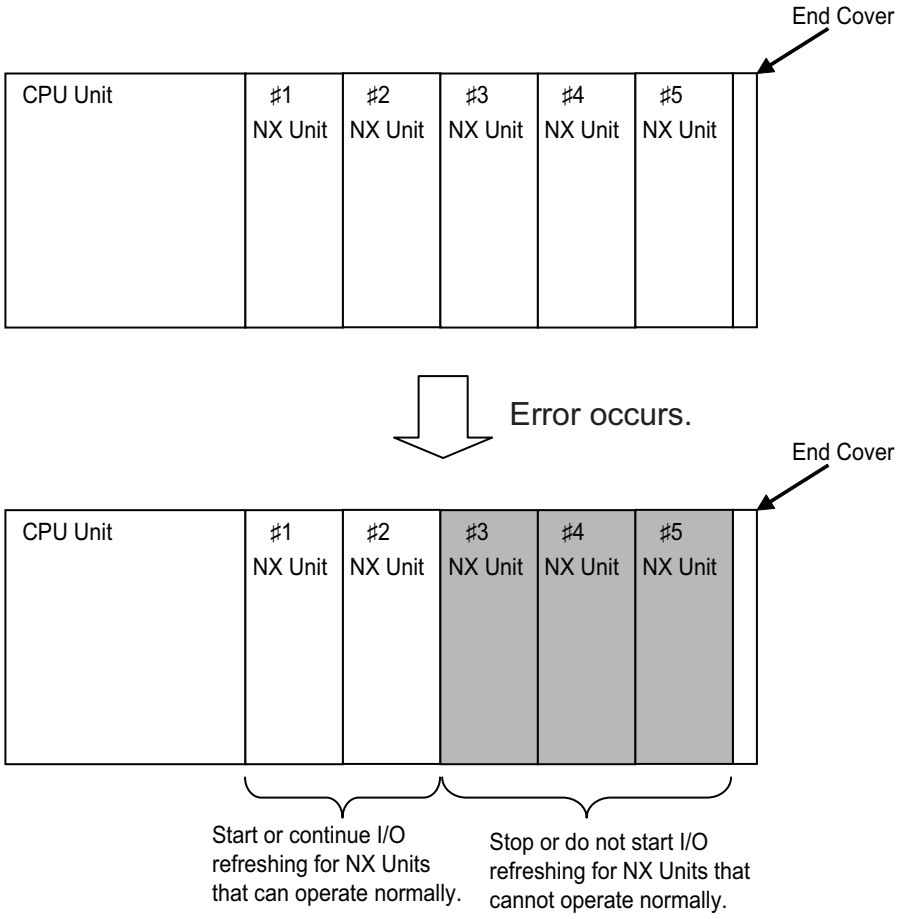
8-2-5 Fail-soft Operation for NX Units on the CPU Unit

This section describes the fail-soft operation for the NX Bus Function Module of the CPU Unit.
Only the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units have the NX Bus Function Module.

This function allows the NX Bus Function Module to start or continue I/O refreshing only with the NX Units on the CPU Unit that can operate normally when an error occurs for the NX Bus Function Module.

For example, you can use this function in the following cases.

- When it is dangerous to stop all NX Units on the CPU Unit at once.
- To continue the operation of the NX Units on the CPU Unit until the system can be stopped safely through the user program or user operation.
- To not stop all devices, i.e., to continue operation for only some devices



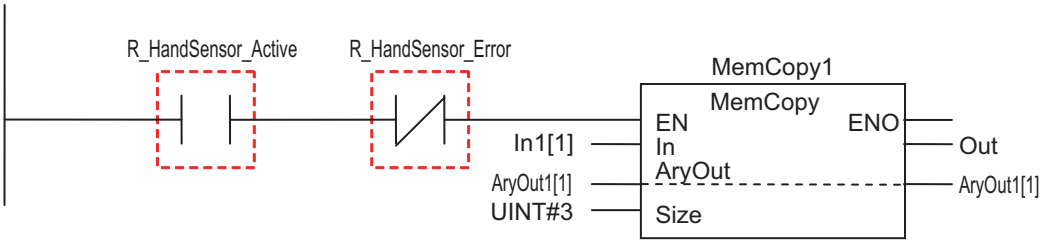
Precautions for Safe Use

If you change the fail-soft operation setting, the output status when the error occurs may also change. Confirm safety before you change the setting.

Precautions for Correct Use

If you use fail-soft operation, write programming to determine whether Unit I/O data is valid. Without such programming, the user program cannot distinguish between Units for which I/O refreshing is continued and Units for which I/O refreshing is stopped.
To determine whether Unit I/O data is valid, use the variables to indicate the NX Unit I/O Data Active Status and NX Unit Error Status from the NX Unit device variables.

A sample programming that determines whether NX Unit (NX Unit number: 5) I/O data on the CPU Unit is valid is given below.



R_HandSensor_Active: This is a device variable to indicate the NX Unit I/O Data Active Status with NX Unit number 5.
R_HandSensor_Error: This is a device variable to indicate the NX Unit Error Status with NX Unit number 5.

Operations for Errors

The following table describes the operation of the NX Bus Function Module when the NX Bus Function Module is used with and without fail-soft operation.

Operating status	Operation when an error occurs while starting	Operation when an error occurs during normal operation
With fail-soft operation	The NX Bus Function Module starts I/O refreshing for the NX Units that can operate normally. It does not start I/O refreshing for NX Units that cannot operate normally.	The NX Bus Function Module continues I/O refreshing for the NX Units that can operate normally. It stops I/O refreshing for NX Units that cannot operate normally.
Without fail-soft operation*1	The NX Bus Function Module does not start I/O refreshing for any of the NX Units.	The NX Bus Function Module stops I/O refreshing for any of the NX Units.

*1. When an error occurs, I/O refreshing for the NX Units on the CPU Unit that is not started, i.e., I/O refreshing for the NX Units on the Controller that is stopped is called “entire stop”.

Except for the I/O refreshing, the operation when an error occurs for the NX Bus Function Module is the same regardless of whether fail-soft operation is used. Specifically, error notification is provided and errors are recorded in the event log.

Setting Fail-soft Operation

● Using Fail-soft Operation

To enable fail-soft operation, select the CPU Unit in the CPU and Expansion Racks Tab Page on the Sysmac Studio and set the **Fail-soft Operation Setting** to **Fail-soft operation**.
The default for the **Fail-soft Operation Setting** for the NX Bus Function Module is **Fail-soft operation**.

● Not Using Fail-soft Operation

To disable fail-soft operation, select the CPU Unit in the CPU and Expansion Racks Tab Page on the Sysmac Studio, and set the **Fail-soft Operation Setting** to **Stop**.

Precautions for Correct Use

- After you change the setting, always transfer the changed settings to the CPU Unit.
- Refer to *4-2-4 Configuration and Setup for NX Units* on page 4-13 for information on the Unit configurations and settings for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

Errors to Which Fail-soft Operation Applies

The following errors are examples of the errors to which fail-soft operation applies.

- NX Bus Communications Error
- Registered NX Unit Not Mounted
- NX Unit Communications Timeout
- NX Unit Initialization Error
- NX Unit Startup Error

Even if you enable **Fail-soft operation**, the NX Bus Function Module may not start I/O refreshing for all of the NX Units when the CPU Unit is started, depending on the cause of the error.
Refer to *Causes of Unit Configuration Verification Errors and Error Operation* on page 8-18 for details on the operation for different error causes.

Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for the errors to which fail-soft operation applies.
If an error occurs to which fail-soft operation does not apply, the NX Bus Function Module will stop I/O refreshing for all NX Units even if you enable fail-soft operation.

Causes of Unit Configuration Verification Errors and Error Operation

Even if you enable **Fail-soft operation**, I/O refreshing may not start depending on the cause of the error when the CPU Unit starts.
Examples are provided below.

Example of Unit configuration information and actual configuration		NX Unit numbers					Description of configuration	Operation when CPU Unit starts
		1	2	3	4	5		
Unit configuration information		A	B	C	D	E*1	The following models of Units are mounted after the CPU Unit in the order given on the left: A, B, C, D, and E.	---
Actual configuration	Case 1	A	B	C	---	---	Unit D is not mounted.	I/O refreshing does not start for NX Unit numbers 1, 2, and 3 because fail-soft operation is enabled.
	Case 2	A	C	D	---	---	Unit B is not mounted.	I/O refreshing does not start for any of the NX Units.
	Case 3	A	B	D	C	---	Unit C and D are mounted in reverse order.	I/O refreshing does not start for any of the NX Units.
	Case 4	A	B	C	D	D	An extra Unit D is mounted for NX Unit number 5.	I/O refreshing does not start for any of the NX Units.
	Case 5	A	B	C	F	---	Unit F is mounted for NX Unit number 4, but it does not exist in the Unit configuration information.	I/O refreshing does not start for any of the NX Units.
	Case 6	A	B	C	D	E	Unit E is mounted for NX Unit number 5 even though its NX Unit Mounting Setting is set to *Disable*.	I/O refreshing does not start for any of the NX Units.

*1. Unit E has the **NX Unit Mounting Setting** set to **Disable**.

8-2-6 Monitoring Total Power-ON Time for NX Units on the CPU Unit

This section describes how to monitor the total power-ON time for NX Units on the CPU Unit.
Each of the NX Units on the CPU Unit records the total time that the Unit power supply is ON to it. You can display these times on the Sysmac Studio.

Specifications of Monitoring Total Power-ON Times

The specifications of monitoring the total power-ON times are given in the following table.

Item	Specification
Display unit	• When total power-ON time is less than 1 hour : Minutes • When total power-ON time is 1 hour or longer : Hours
Update interval	• When total power-ON time is less than 24 hours : 10 minutes • When total power-ON time is 24 hours or longer : 1 hour
Measurement error	1 hour/month max.
Default setting	0 minutes

Checking Total Power-ON Times

You can use the Sysmac Studio to check the total power-ON times of NX Units on the CPU Unit.

Or, you can use the following special instruction to check the total power-ON times.
- NX_ReadTotalPowerOnTime instruction

This section describes how to check the total power-ON times with the Sysmac Studio. Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for details on the NX_ReadTotalPower-OnTime instruction.

● Checking Total Power-ON Times with Sysmac Studio

You can use the Production Information on the Sysmac Studio to check.

1 Go online.

2 Right-click **CPU Rack** under **Configurations and Setup** - **CPU/Expansion Racks** in the Multi-view Explorer and select **Production Information**.
The Production Information Dialog Box is displayed.
You can check the total power-ON times of each NX Unit when you change the view to the production information details.

● Display When Times Cannot Be Recorded

If the total power-ON time cannot be recorded because of a non-volatile memory hardware error, the total power-ON time is displayed as *Invalid record* on the Sysmac Studio.

● Display for Units That Do Not Support Monitoring the Total Power-ON Time

If a Unit does not support monitoring the total power-ON time, the total power-ON time for the Unit is displayed as "---" on the Sysmac Studio.

● Display When Reading the Time Failed

If reading the time failed, the total power-ON time is displayed as "---" on the Sysmac Studio.

8-2-7 Clearing All Memory for Each NX Unit on the CPU Unit

This section describes how to clear all memory for each NX Unit on the CPU Unit.
The various settings in NX Units on the CPU Unit are initialized to the factory settings.
The NX Unit restarts after Clear All Memory operation.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on how to execute Clear All Memory operation for NX Units.

Precautions for Correct Use

- NX Units mounted on the NX1P2 CPU Unit cannot perform Clear All Memory operation individually. Perform the Clear All Memory operation for the CPU Unit.
- The function to restart the NX Unit after clearing all memory has restrictions on the support and versions for NX Units. Refer to *Support Functions of the CPU Units and Restrictions on the NX Units* in the *NX-series Data Reference Manual (Cat. No. W525)* for details on the restrictions.

The following table shows the status of each data type when the Clear All Memory operation is performed for each NX Unit.

Type of data	Status after Clear All Memory operation
Unit operation settings	It will be factory setting.
Unit application data	For NX Units with Unit application data, check the operation specifications at the time of Clear All Memory operation in the user's manual for the NX Units.
Event logs	If you select to clear event logs at execution, the logs will be cleared.

8-3 Management Function for X Bus Units

This section describes the management functions used for X Bus Units.
You can use X Bus Units only on the NX502 CPU Units.

8-3-1 X Bus Function Module

The X Bus Function Module performs processing such as a management of status and I/O refreshing for the X Bus Units that are connected to the X Bus of the CPU Unit as a master of the X Bus (hereafter X Bus master).

Device variables assigned to the I/O ports are used to access the I/O data. Refer to *8-3-2 X Bus Unit Common Function Module* on page 8-23.

The following describes the status of X Bus Units managed by the X Bus Function Module as the X Bus master.

Status of X Bus Units Managed by the X Bus Function Module as the X Bus Master

For the status of X Bus Units managed by the X Bus Function Module as the X Bus master, you can use system-defined variables to access.

● A List of Status of X Bus Units Managed by the X Bus Master

Name	System-defined variable
X Bus Unit Registration Status	_XBU_RegTbl
X Bus Function Module Error Status	_XBU_ErrSta
X Bus Function Module Master Error Status	_XBU_MstrErr
X Bus Function Module Unit Error Status	_XBU_UnitErr
X Bus Function Module Unit Error Status Table	_XBU_UnitErrTbl

● Descriptions of Status of X Bus Units Managed by the X Bus Master

Name	Description
X Bus Unit Registration Status	This status tells whether the X Bus Units are registered in the Unit configuration.
X Bus Function Module Error Status	Gives the status of errors that are detected in the X Bus Function Module.
X Bus Function Module Master Error Status	Gives the status of errors that are detected in the X Bus master.
X Bus Function Module Unit Error Status	Gives the collective value of all *X Bus Function Module Unit Error Status Table*.
X Bus Function Module Unit Error Status Table	Gives the status of errors that are detected in the X Bus Unit.

Refer to *A-7-6 X Bus Function Module, Category Name: _XBU* on page A-114 for details on the related system-defined variables.

8-3-2 X Bus Unit Common Function Module

The X Bus Unit Common Function Module, a function module of the X Bus Unit, performs processing such as a management of X Bus Unit common information, i.e. total power-ON time and unit version, etc., management of event logging, management of status, and I/O refreshing with the CPU Unit.

For I/O data that are handled by the X Bus Unit Common Function Module , there are two kinds of I/O data, X Bus Unit common I/O data and I/O data unique to individual X Bus Units. The variables are the assignable I/O ports for both of them.

Device variables assigned to the I/O ports are used to access these I/O data.
Refer to *A-17-1 X Bus Unit Common Function Module* on page A-289 for details on device variables for X Bus Unit common I/O data.
The following describes the I/O data for individual X Bus Units and assigning device variables to I/O ports.

I/O Data for Individual X Bus Units

I/O data are determined by the model number of the X Bus Unit and the functionality.

Refer to the user's manual for the specific X Bus Unit for details on I/O data unique to individual X Bus Units.

Assigning Device Variables to I/O Ports

When you create the Unit configuration information on the Sysmac Studio, the X Bus Unit common I/O data are automatically registered as I/O ports.
The variables that are assigned to these statuses and I/O data are device variables.

● I/O Port Names

Refer to *Variable name* in the table in *A-17-1 X Bus Unit Common Function Module* on page A-289 for the X Bus Unit common I/O port names.

An example for I/O port name: PowerOnHour (Total Power ON Time)

● Registering Device Variables

You assign device variables to I/O ports in the I/O Map of the Sysmac Studio. The device variables that you create are registered in the variable table.
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

Refer to *Device Variable Attributes* on page 3-12 for details on device variable attributes.

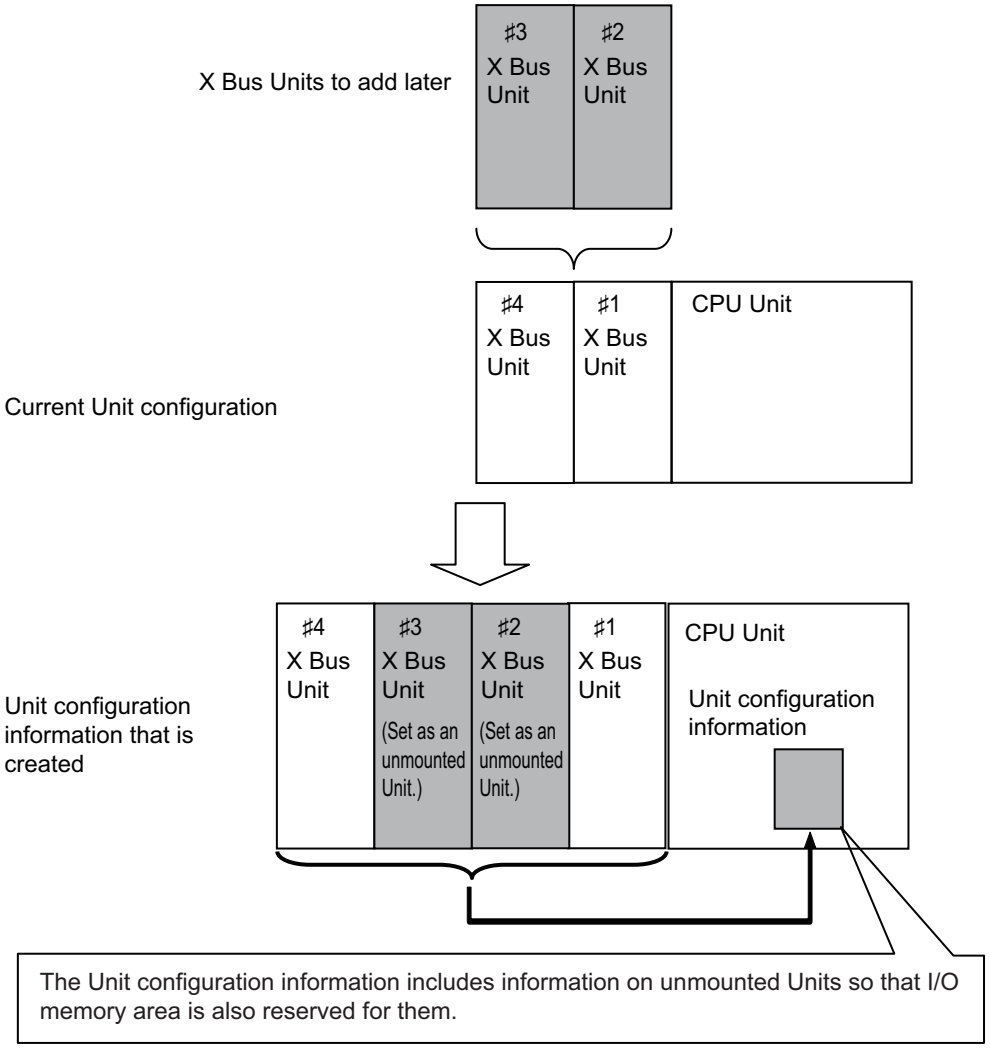
8-3-3 X Bus Unit Mounting Settings

This section describes the mounting settings of X Bus Units.

Overview of Function

You can use this function to register X Bus Units that will be added to at a later time in the Unit configuration information as unmounted Units. With this function, you can create the program in advance for X Bus Units that are not mounted to the actual configuration.
You can use this function even if a specific Unit is temporarily unavailable such as when commissioning the system.

- I/O memory area is reserved for these unmounted X Bus Units in the same way that it is reserved for X Bus Units that are mounted to the actual configuration.
- Unmounted X Bus Units are also assigned X Bus Unit numbers. This prevents the X Bus Unit numbers of other X Bus Units on the CPU Unit from changing when you change the setting of an X Bus Unit to mount it to the actual configuration.



Operating Specifications for X Bus Units That Are Set as Unmounted Units

The operating specifications for X Bus Units that are set as unmounted Units are given in the following table.

Item	Operation
System-defined variables for X Bus Units	Bandwidth is reserved. The input value is always 0.
Variable refreshing with the CPU Unit	The variables are not refreshed.
Detection of events	Events are not detected.
Message communications	Not possible because the Units do not exist.
Transfers for the synchronization function of the Sysmac Studio	Not applicable.
Sysmac Studio Controller backup function	Not applicable.
SD Memory Card backup function	Not applicable.
Instructions	Parameters cannot be read or written. An instruction error will occur.
Clearing all memory	Not applicable.
Reading/writing setting information through backup/restore operations	Not applicable.
Reading event logs	Not applicable.
Notification of status information	Not applicable.

Precautions for Safe Use

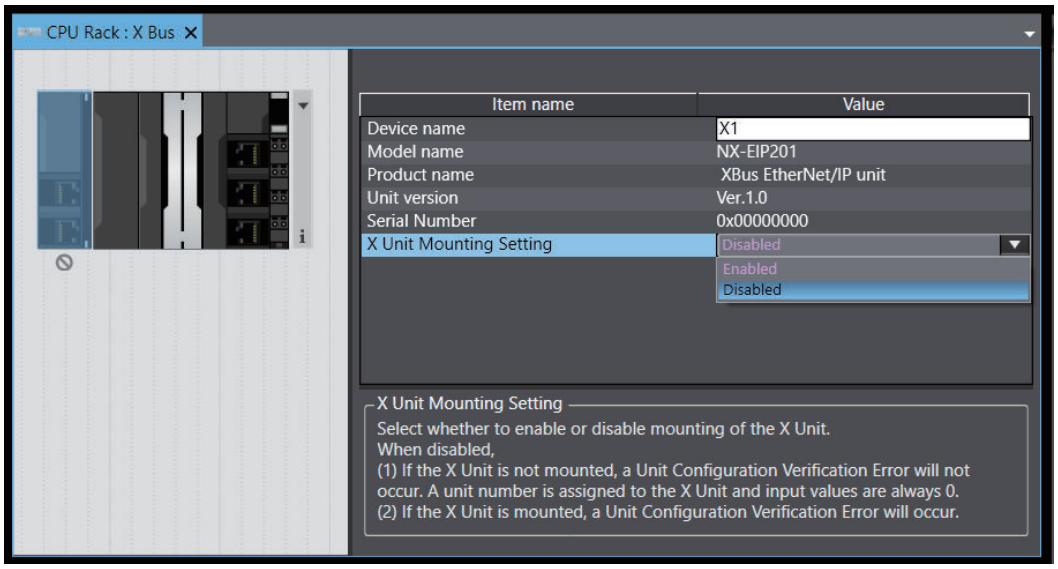
Check the user program, data, and parameter settings for proper execution before you use them for actual operation.

Precautions for Correct Use

When you mount an X Bus Unit that was set as an unmounted Unit, a Unit Configuration Verification Error will occur.

Setting X Bus Units as Unmounted Units

In the CPU Rack: X Bus Tab Page on the Sysmac Studio, select the target X Bus Unit and set the **Unit Mounting Setting** to **Disabled**. The selected Unit is set as an unmounted Unit. After you change the settings for any Units, always transfer the Unit configuration information to the CPU Unit.



8-3-4 X Bus Unit Setting Synchronization Function

This section describes the setting synchronization function for each X Bus Unit.

The individual setting synchronization function can transfer (between a computer and a Unit) and verify the settings of each X Bus Unit.

Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for operating procedures of transferring and comparing settings.

Precautions for Correct Use

When you transfer the settings from a computer to a Unit, if the settings other than the X Bus Unit settings differ from Sysmac Studio setting values and the actual X Bus Unit setting values, the settings cannot be transferred.

8-3-5 Clearing All Memory for Each X Bus Unit

This section describes how to clear all memory for each X Bus Unit.

The individual Clear all memory function can clear the settings of each X Bus Unit.

After the Clear All Memory operation is executed, the cleared X Bus Units operate with the default settings.

Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on how to clear all memory for each X Bus Unit.

Precautions for Correct Use

- The conditions under which the Clear All Memory operation for each X Bus Unit can be executed are as follows.
  a) The operating mode of the CPU Unit is PROGRAM mode.
  b) The X Bus Unit configuration in the CPU Unit matches the actual X Bus Unit configuration.
- During the Clear All Memory operation, the I/O refreshing for the CPU Unit and all X Bus Units is stopped.

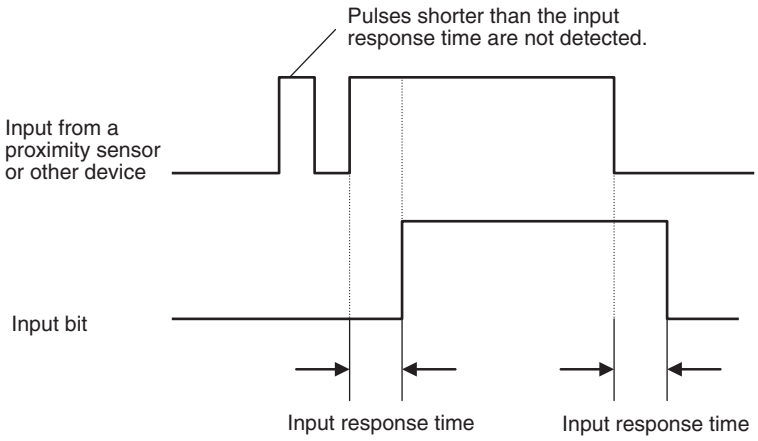
8-4 Management Functions for CJ-series Units

This section describes the management functions used for Units in the Controller.
You can use CJ-series Units only with NJ-series CPU Units.

8-4-1 Basic I/O Units

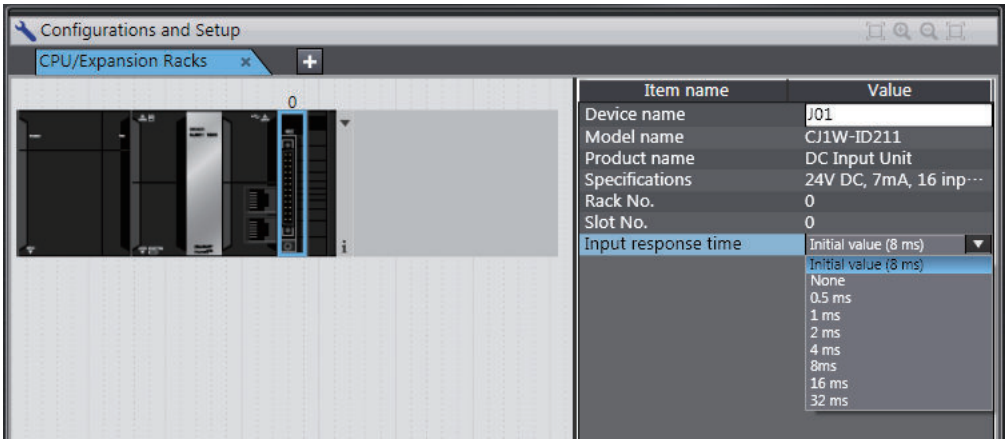
Introduction

You can increase the input response time to reduce chattering and the effects of external noise. You can decrease the input response time to enable detection of shorter input pulses. Do not set the ON response time or OFF response time to less than the refresh time.



Setting Methods

From the Multiview Explorer of the Sysmac Studio, double-click **CPU/Expansion Racks** under **Configurations and Setup**. Then select the input response times in the Unit information for the Basic I/O Units.



You must do either of the following to enable the settings.

- Cycle the power supply to the Controller.
- Reset the Controller (the entire CPU Unit) from the Sysmac Studio.

Related System-defined Variables

The set values for the input response times of the Basic Input Units are output to the following system-defined variable.

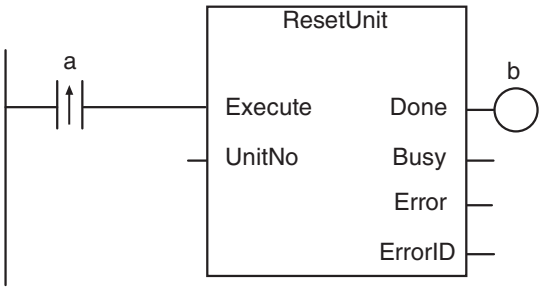
Variable name	Meaning	Function	Data type	R/W
_CJB_InRespTm	Basic Input Unit Input Response Times	Contains the response times of the Basic I/O Units in 0.1-ms increments.	ARRAY [0..3, 0..9] OF UINT	R

8-4-2 Special Units

Restarting Special Units

You can restart a Special Unit (Special I/O Unit or CPU Bus Unit) to enable values that are set for it. If you restart a Special Unit, you do not have to cycle the power supply to the Controller.
Execute the following ResetUnit (Restart Unit) instruction to restart Special Units.

Instruction name	Instruction	Description
Restart Unit	ResetUnit	Restarts the CPU Bus Unit or Special I/O Unit.



The ResetUnit instruction restarts a Special Unit across multiple task periods when execution condition *a* changes to TRUE. If the restart ends normally, the output variable *Done* (normal end) changes to TRUE and variable *b* therefore changes to TRUE.

If Special Unit settings are changed in any of the following ways, you must restart the Special Unit or cycle the power supply to the Controller.

- Editing from the Special Unit Settings Tab Page of the Sysmac Studio
- Editing from the I/O Map or Watch Tab Page
- Setting the user program

● Related System-defined Variables

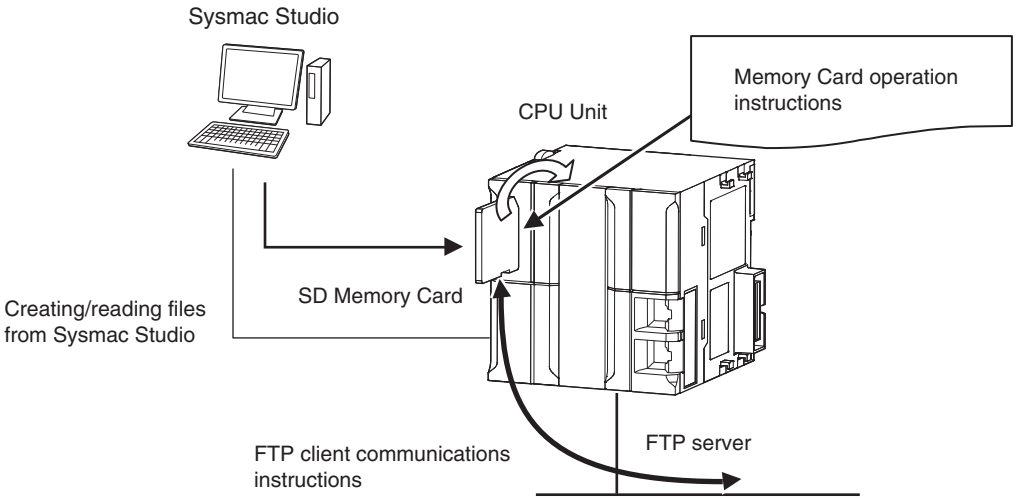
Variable name	Meaning	Function	Data type	R/W
_CJB_CBU00InitSta to _CJB_CBU15InitSta	CPU Bus Unit Initializing Flags	The corresponding variable is TRUE during initialization of the CPU Bus Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units.	BOOL	R
_CJB_SIO00InitSta to _CJB_SIO95InitSta	Special I/O Unit Initializing Flags	The corresponding variable is TRUE during initialization of the Special I/O Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units.	BOOL	R
_CJB_CBU00Restart to _CJB_CBU15Restart	CPU Bus Unit Restart Bits	The CPU Bus Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the CPU Bus Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Bit to TRUE with an instruction, the restart process begins from refresh processing in the next task period.	BOOL	RW
_CJB_SIO00Restart to _CJB_SIO95Restart	Special I/O Unit Restart Bits	The Special I/O Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the Special I/O Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Bit to TRUE with an instruction, the restart process begins from refresh processing in the next task period.	BOOL	RW

8-5 SD Memory Card Operations

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

8-5-1 SD Memory Card Operations

The NJ/NX-series CPU Unit supports the following functions for SD Memory Cards.



Function	Introduction
SD Memory Card operation instructions	You can access SD Memory Cards from instructions in the user program.
FTP Client communications instructions[*1]	You can use these instructions to transfer files via FTP from the CPU Unit to computers or Controllers at Ethernet nodes.
FTP server	You can use FTP commands from an FTP client on the Intranet to read and write large files in the SD Memory Card through EtherNet/IP.
File operations from the Sysmac Studio	You can perform file operations from the Sysmac Studio for the SD Memory Card inserted in the CPU Unit. You can perform file operations for Controller files in the SD Memory Card and save standard document files on the computer.
SD Memory Card life expiration detection	Notification of the expiration of the life of the SD Memory Card is provided in a system-defined variable and event log.
SD Memory Card backup, automatic transfer from SD Memory Card, and program transfer from SD Memory Card[*2]	You use the SD Memory Card inserted in the CPU Unit to backup, restore, and verify user programs and data in the Controller. For details, refer to *9-2 SD Memory Card Backups* on page 9-15, *9-4 Automatic Transfers from SD Memory Cards* on page 9-37, and *9-5 Program Transfer from SD Memory Card* on page 9-39 for details.
Safety Data Logging[*3]	You can record I/O data for the NX-SL5□□□ that is mounted to the NX bus of the NX502 CPU Unit or NX102 CPU Unit in a chronological order into an SD Memory Card. Refer to the *NX-series Safety Control Unit User's Manual (Cat. No. Z930-E1-12 or later)* for details.

Function	Introduction
Secure socket communications logging*4	You can record parameters, starting, ending, and errors of a TLS session. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for details.

*1. A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the FTP client communications instructions.

*2. A CPU Unit with unit version 1.11 or later and Sysmac Studio version 1.15 or higher are required to use the function.

*3. An NX502 CPU Unit with unit version 1.60 or later, an NX102 CPU Unit with unit version 1.31 or later, or an NX-SL5□□□ Safety CPU Unit with unit version 1.3 or later and Sysmac Studio version 1.24 or higher are required to use the Safety Data Logging.

*4. An NX502 CPU Unit with unit version 1.60 or later, an NX102-□□00 CPU Unit with unit version 1.46 or later, an NX102-□□20 CPU Unit with unit version 1.37 or later, or an NX1P2 CPU Unit with unit version 1.46 or later and Sysmac Studio version 1.46 or later are required to use the secure socket communications logging.

8-5-2 Specifications of Supported SD Memory Cards, Folders, and Files

SD Memory Card Specifications

The NJ/NX-series Controllers support both SD cards and SDHC cards. However, operation was confirmed only for the OMRON SD Memory Card given in the following table. Correct operation may not be possible if you use any other SD or SDHC card.

Model	Card type	Capacity [GB]	Formatting	Number of rewrites (per block)	Write protection	Weight
HMC-SD291	SD card	2	FAT16	60,000 times	You can write-protect the SD Memory Card with a hardware switch on the Card.	Approx. 2g
HMC-SD292				50,000 times		
HMC-SD491	SDHC card	4	FAT32	60,000 times		
HMC-SD492				100,000 times		
HMC-SD1A1		16		60,000 times		
HMC-SD1A2				100,000 times		

The system-defined variable *_Card1Err* (SD Memory Card Error Flag) changes to TRUE (observation level) in the following cases.

- When there is a format error

If an error occurs, the SD PWR indicator on the front of the CPU Unit goes out, and accessing the SD Memory Card will not be possible.

Precautions for Correct Use

The SD Memory Card you can use depends on the model and version of the Controller.

CPU models other than NX102-□□□□

Unit version	HMC-SD291	Type:HMC-SD491	Type:HMC-SD1A1	Type:HMC-SD292	Type:HMC-SD492	Type:HMC-SD1A2
Ver.1.00 to Ver.1.20	*1	*3	*2	*1 *4	*4	*2
Ver.1.21 to Ver.1.23		Can be used without restrictions.		*1 *4	*4	
Ver.1.24 to Ver.1.29				*1	Can be used without restrictions.	
Ver.1.40 to Ver.1.42				*1 *4	*4	
Ver.1.43 or later				*1	Can be used without restrictions.	

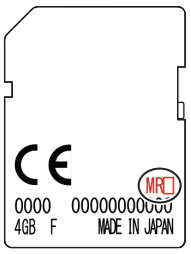
NX102-□□□□

Unit version	HMC-SD291	HMC-SD491	HMC-SD1A1	HMC-SD292	HMC-SD492	HMC-SD1A2
Ver.1.30 to Ver.1.31	Can be used without restrictions.		*2	*4		*2
Ver.1.32 to Ver.1.35			Can be used without restrictions.	*4		
Ver.1.36 to Ver.1.39				Can be used without restrictions.		
Ver.1.40 to Ver.1.42				*4		
Ver.1.43 or later				Can be used without restrictions.		

*1. You cannot use an HMC-SD291 or HMC-SD292 for the NJ 501-□□□□ CPU Unit with the hardware revision "A" and unit version 1.15 or later.

*2. Cannot be used.

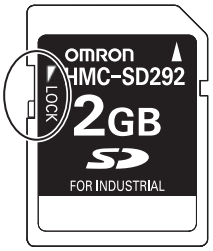
*3. You cannot use a SD Memory Card with "MR□" written on the back for the NJ501-□□□□ CPU Unit with unit version 1.12 or earlier or the NX701-□□□□ CPU Unit with unit version 1.13 or earlier.



*4. *8-5-7 SD Memory Card Life Expiration Detection* on page 8-35 cannot be used.

Additional Information

Write Protection Key
You will not be able to write to the SD Memory Card if the key is set to the LOCK position. Use this setting to prevent overwriting.



Folder and File Specifications

Character Restrictions

Object named by user	Usable characters	Reserved words	Multibyte character compatibility	Case sensitivity	Maximum size (without NULL)
Volume label	0 to 9, A to Z, and a to z, as well as % - _ @ ! ‘ ( ) ~ = # & + ^ [ ] { } , . ; and single-byte kana*1	CON, PRN, AUX, CLOCK$, NUL, COM0, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT0, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9	Not supported.*2	Case insensitive	11 bytes
Directory name	0 to 9, A to Z, and a to z, as well as $ % ' - _ @ ! ` ( ) ~ = # & + ^ [ ] { } , . ; and single-byte kana				65 bytes
File names					65 bytes

*1. You cannot begin volume label names with a space.
*2. Even if the computer supports multibyte characters (e.g., for Japanese), you cannot use them in the CPU Unit.

Subdirectory Levels

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

Maximum Number of Stored Files

The number of files that you can store on an SD Memory Card depends on the directory level in which you store the files. The maximum number of files for each is given in the following table. However, the values in the table assume that 8.3 filename is used. If you use long file names, the maximum number of stored files is less than the value given in the following table.

Directory level	Format	Maximum number of stored files
Root directory	**FAT16**	511
	FAT32	65,533

Directory level	Format	Maximum number of stored files
Subdirectory	**FAT16, FAT32**	65,533

● Maximum Size of One File

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

8-5-3 SD Memory Card Operation Instructions

You can perform various operations on the SD Memory Card by using the following instructions.

Instruction name	Instruction	Description
Read Variable from File	**FileReadVar**	The FileReadVar instruction reads the contents of a binary file on the SD Memory Card and stores it to the specified variable. You can specify array and structure variables.
Write Variable to File	**FileWriteVar**	The FileWriteVar instruction writes the value of a specified variable to a binary file in the SD Memory Card. You can specify array and structure variables. If the directory specified for the file name does not exist, it is created.
Open File	**FileOpen**	The FileOpen instruction opens the specified file.
Close File	**FileClose**	The FileClose instruction closes the specified file.
Seek File	**FileSeek**	The FileSeek instruction sets a file position indicator in the specified file.
Read File	**FileRead**	The FileRead instruction reads the data from the specified file.
Write File	**FileWrite**	The FileWrite instruction writes data to the specified file.
Get Text String	**FileGets**	The FileGets instruction reads a text string of one line from the specified file.
Put Text String	**FilePuts**	The FilePuts instruction writes a text string of one line to the specified file.
Delete File	**FileRemove**	The FileRemove instruction deletes the specified file from the SD Memory Card.
Change File Name	**FileRename**	The FileRename instruction changes the name of the specified file or directory.
Copy File	**FileCopy**	The FileCopy instruction copies the specified file to a different file.
Create Directory	**DirCreate**	The DirCreate instruction creates a directory in the SD Memory Card.
Delete Directory	**DirRemove**	The DirRemove instruction deletes a directory from the SD Memory Card.

8-5-4 FTP Client Communications Instructions

FTP client communications instructions are used to transfer files via FTP from an NJ/NX-series CPU Unit to computers or Controllers at Ethernet nodes. The files on the SD Memory Card are read and written when the following instructions are executed.

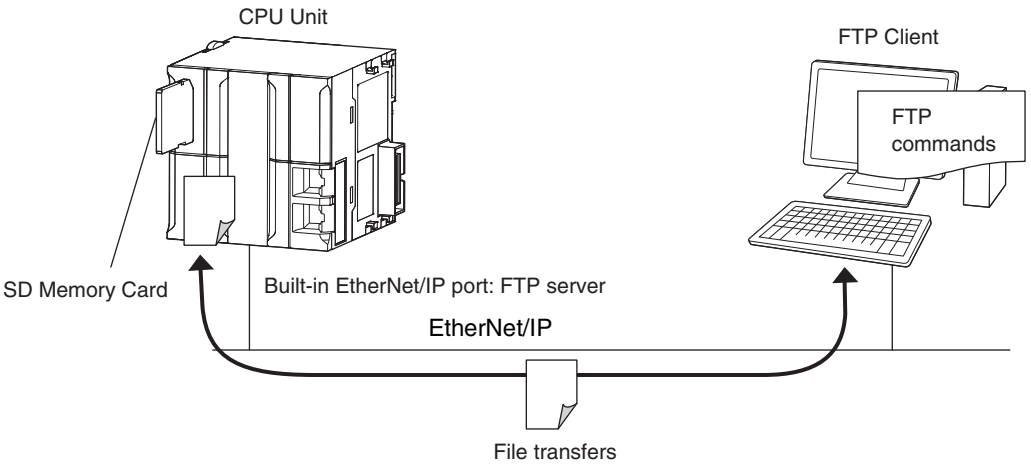
Instruction name	Instruction	Description
Put File onto FTP Server	**FTPPutFile**	The FTPPutFile instruction uploads one or more files in the FTP client's SD Memory Card to the FTP server.
Get File from FTP Server	**FTPGetFile**	The FTPGetFile instruction downloads one or more files from the FTP server to the FTP client's SD Memory Card.

Version Information

A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the FTP client communications instructions.

8-5-5 FTP Server

You can read and write files on the SD Memory Card via EtherNet/IP by sending FTP commands to the built-in EtherNet/IP port from an FTP client.
Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for details.



8-5-6 File Operations from the Sysmac Studio

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

8-5-7 SD Memory Card Life Expiration Detection

You can determine the remaining life of the SD Memory Card before the Card becomes physically deteriorated.
You can determine the remaining life of the SD Memory Card with the following functions.

- System-defined variable *_Card1Deteriorated* (SD Memory Card Life Warning Flag)
- SD Memory Card Life Exceeded (Observation) record in the event log

The life of the SD Memory Card is checked when the power is turned ON and periodically while the SD Memory Card is inserted.

When the end of the life of the SD Memory Card is detected, save the data on the SD Memory Card and replace the SD Memory Card.

Version Information

Combination of the CPU Unit version and SD Memory Card determines whether the SD memory card life expiration detection function can be used or not. Refer to *8-5-2 Specifications of Supported SD Memory Cards, Folders, and Files* on page 8-31.

8-5-8 List of System-defined Variables Related to SD Memory Cards

The following system-defined variables show the status of the SD Memory Card.

Variable name	Meaning	Function	Data type
_Card1Ready	SD Memory Card Ready Flag	TRUE when the SD Memory Card is recognized. FALSE when an SD Memory Card is not recognized. TRUE: The Card can be used. FALSE: The Card cannot be used.	BOOL
_Card1Protect	SD Memory Card Write Protected Flag	TRUE when the SD Memory Card is write-protected. TRUE: Write protected. FALSE: Not write protected.	BOOL
_Card1Err	SD Memory Card Error Flag	TRUE when an unusable SD Memory Card is inserted or a format error occurs. TRUE: There is an error FALSE: There is no error	BOOL
_Card1Access	SD Memory Card Access Flag*1	TRUE during SD Memory Card access. TRUE: Card is being accessed. FALSE: Card is not being accessed.	BOOL
_Card1Deteriorated	SD Memory Card Life Warning Flag	TRUE when the life of the SD Memory Card is exceeded. TRUE: The life of the Card has been exceeded. FALSE: The Card can still be used.	BOOL
_Card1PowerFail	SD Memory Card Power Interruption Flag*2	TRUE when the power supply to the Controller was interrupted during access to the SD Memory Card. TRUE: Power was interrupted during SD Memory Card access. FALSE: Normal	BOOL
_Card1Capacity	SD Memory Card Storage Capacity	Show the total capacity of the inserted SD Memory Card. The unit of capacity is MiB (1 MiB=1,048,576 bytes). The value is updated every 60 seconds. If the SD PWR indicator is not lit, such as when an SD Memory Card is not inserted, the value is "0".	UDINT

Variable name	Meaning	Function	Data type
_Card1Used	SD Memory Card Storage Usage	Show the usage of the inserted SD Memory Card. The unit of capacity is MiB (1 MiB=1,048,576 bytes). The value is updated every 60 seconds. If the SD PWR indicator is not lit, such as when an SD Memory Card is not inserted, the value is "0".	UDINT

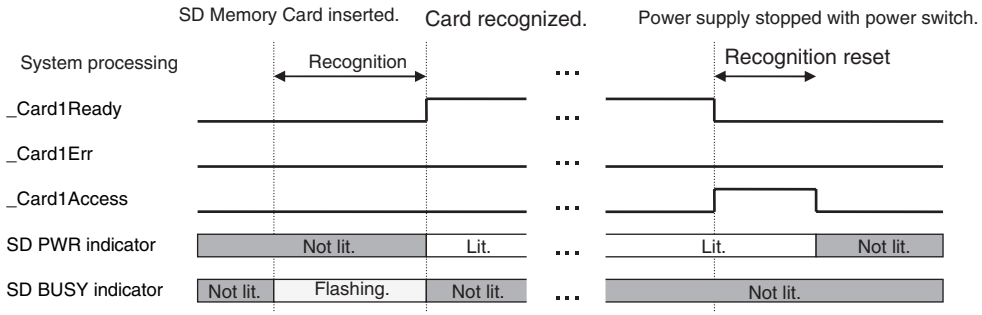
*1. Precaution When Using SD Memory Card Access Flag (*_Card1Access*)
The SD Memory Card Access Flag is intended for use in notifying external devices. The status of access to the SD Memory Card is not updated in realtime. Because of this, do not use the flag in the user program. Because the status of access to the SD Memory Card is not shown in realtime, it may cause unexpected Controller operation if you use it in the user program.

*2. Precautions When Using the SD Memory Card Power Interruption Flag (*_Card1PowerFail*)
If the SD Memory Card Power Interruption Flag is TRUE, check to see if the correct file is in the SD Memory Card and to see if the SD Memory Card operates properly.
If the correct file is missing or the SD Memory Card does not operate properly, download the correct file to the SD Memory Card again. Cycle the power supply to the Controller or reset the Controller, and then see if the SD Memory Card operates properly.
When it is finished, change SD Memory Card Power Interruption Flag to FALSE. (*_Card1PowerFail* does not change to FALSE automatically.)

Note Refer to *9-2 SD Memory Card Backups* on page 9-15 for the system-defined variables that are used with the SD Memory Card backup function.

Additional Information

SD Memory Card Recognition and Unmounting Timing Chart



8-5-9 SD Memory Card Self-diagnostic Functions

You can perform self-diagnosis on the inserted SD Memory Card when the power supply is turned ON.

You can select whether to perform self-diagnosis when the power is turned ON in the **Operation Settings** of the **Controller Setup** under the **Configurations and Setup** from the Sysmac Studio as shown below.

- File system check
- Check equivalent to CHKDSK
- Restoration attempt when check fails

Access point	Setting group	Setting	Description	Set value
Operation Settings, Operation Settings Tab, Basic Settings	SD Memory Card Settings	Memory Card Diagnosis at Startup*1	Sets whether to execute self-diagnosis (file system check and restoration) on the inserted SD Memory Card when the power is turned ON.	Do not check. Check.

*1. Self-diagnosis is not executed if write protection is set on the SD Memory Card itself.

● Results of Self-diagnosis

Case	Indicators			Error type	Correction	Remarks
	RUN	SD PWR	SD BUSY			
Self-diagnosis in progress	Flashing	Not lit	Lit	---	---	---
When self-diagnosis found no problems	---	Lit	Not lit	Normal	None	---
The format of the SD Memory Card is not correct.	---	Not lit	Not lit	Observation	Use the Sysmac Studio to format the SD Memory Card.	---
An error was detected during the file system check and the file system was automatically restored.	---	Lit	Flashes during restore operation. Not lit after restore operation is completed.	Observation	Use file operations in the Sysmac Studio or insert the SD Memory Card into the computer to check whether any files were deleted by the restore operation.	If a corrupted file is detected, an attempt is made to restore the file.
The SD Memory Card failed.	---	Not lit	Not lit	Observation	Replace the SD Memory Card.	---

Precautions for Safe Use

If the recovery function is activated at startup, time is required to enter RUN mode. During that time, outputs will be OFF and external communications are not performed.
Use the RUN output on the Power Supply Unit, for example, to implement fail-safe circuits so that external devices do not operate incorrectly.

Precautions for Correct Use

Never interrupt the power supply to the Controller during SD Memory Card access. That includes when SD Memory Card self-diagnosis at startup is enabled.
An attempt is made by the SD Memory Card restoration function to restore any corrupted files. If the restoration fails, these files may be deleted automatically at startup.

8-5-10 Exclusive Control of File Access in SD Memory Cards

If the same file on the SD Memory Card is accessed from different sources, unintended operations such as reading a file while it is being written or writing a file while it is being read may occur.
Therefore, it is necessary to perform exclusive controls in order to prevent multiple accesses ("reading and writing data" or "writing and writing data") to the same file simultaneously. It is not necessary to perform exclusive controls for "reading and reading data".
When you use a combination of operations that requires exclusive controls, always execute the later processing only after checking that the first processing is finished.
Note that the exclusive controls are performed automatically for the file accesses with more than one instruction.

When the following functions are used, an access to files on the SD Memory Card will occur.

- SD Memory Card operation instructions and FTP client instructions
- Backup, restore and verification operations with the SD Memory Card
- File operations in the SD Memory Card from the Sysmac Studio
- Backup, restore and verification operations from the Sysmac Studio
- FTP server

8-6 Security

This section describes the security functions that are supported by the NJ/NX-series Controller.

To protect your assets, you can use security functions to protect the user program and data in the Controller. To prevent incorrect operation, you can use security functions to restrict operations on the Sysmac Studio.

The NJ/NX-series Controller supports the following security functions.

Application	Security function	Outline of function	Reference
Prevention of the theft of assets	Authentication of user program execution IDs	You can ensure that a user program cannot be operated on another CPU Unit even if the user program is copied.	*8-6-1 Authentication of User Program Execution IDs* on page 8-41
	User program transfer with no restoration information	You can transfer the user program to the CPU Unit without the source code. This prevents anyone from displaying the user program on another computer even if they upload it.	*8-6-2 User Program Transfer with No Restoration Information* on page 8-44
	Overall project file protection	You can place a password on a project file to protect your assets.	*8-6-3 Overall Project File Protection* on page 8-45
	Data protection*1	You can place protection on part of the data in a project file to protect your assets.	*8-6-4 Data Protection* on page 8-46
	Secure communication*2	The communication data between Sysmac Studio and the Controllers is encrypted, and the data is transmitted and received with a hash value. It can prevent data from being eavesdropped or tampered with by a third party.	*8-6-5 Secure Communication Function* on page 8-48
	Secure socket services	You can use secure socket communication encrypted by TLS (Transport Layer Security).	*8-6-6 Secure Socket Services* on page 8-48
	TCP/UDP port close function	You can close TCP/ UDP port of the built-in EtherNet/IP port to prevent unnecessary packets from being received.	*8-6-7 TCP/UDP Port Close Function* on page 8-49
	Packet Filter*2	You can restricts access from external devices by filtering IP packets through the reception process of the built-in EtherNet/IP port.	*8-6-8 Packet Filter* on page 8-49
	Complete data erasure function*3	You can erase all the user-set data in the CPU Unit so that they cannot be restored.	*8-6-9 Complete Data Erasure Function* on page 8-49
Prevention of incorrect operation	Operation authority verification	You can set operation authorities to restrict the operations that can be performed on the CPU Unit from the Sysmac Studio.	*8-6-10 Operation Authority Verification* on page 8-51
	User authentication*2	You can use this function to authenticate each user when Sysmac Studio is going online with the Controller and restrict operation according to the user's privileges.	*8-6-11 User Authentication* on page 8-53

Application	Security function	Outline of function	Reference
	CPU Unit write-protection	You can prevent rewriting data in the Controller from the Sysmac Studio.	*8-6-12 CPU Unit Write Protection* on page 8-57
	Firmware update prohibition*3	You can prevent the firmware update for the CPU Unit.	*8-6-13 Firmware Update Prohibition* on page 8-58
Prevention of incorrect connections	CPU Unit names	You can check to see if the CPU Unit name and serial ID on the computer and in the Controller are the same to prevent going online with the wrong Controller.	*8-6-14 CPU Unit Names and Serial IDs* on page 8-59
Prevention of repudiation	Access log	Operations that the user performed to the Controller using the tool are registered as access logs. If the user authentication function is used, the user name is registered in the access log that is important for security. You can see who performed, when, and what operations, which can be used as a repudiation preventative measure when problems occur.	*8-6-15 Access log* on page 8-60

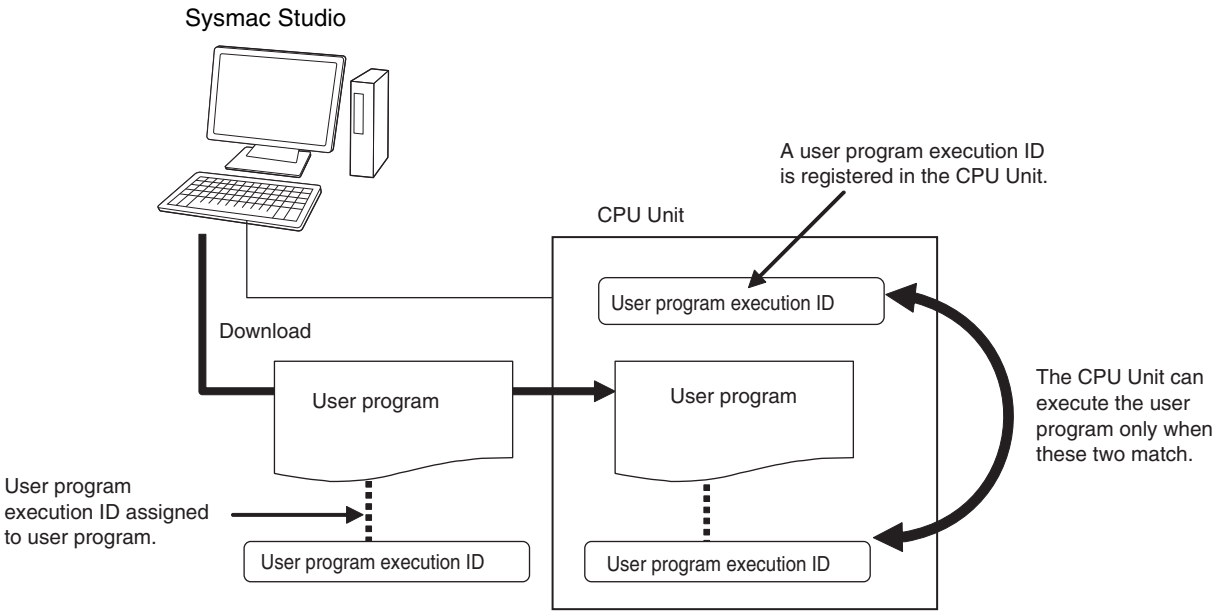
*1. A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required.
*2. This function is available in the following CPU Units of the stated unit versions.
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

*3. This function is available in the following CPU Units of the stated unit versions.
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.60 or later
- NX701 CPU Unit: Version 1.32 or later
- NX502 CPU Unit: Version 1.60 or later

8-6-1 Authentication of User Program Execution IDs

Introduction

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



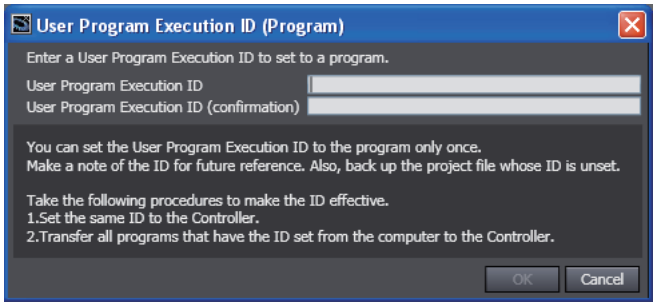
You can therefore prevent different CPU Units (hardware) from executing a user program.

In contrast to the protection function, you can still display and edit the user program even if a user program execution ID is set.

Operating Procedure

1 Always backup the project files before you assign a user program execution ID.

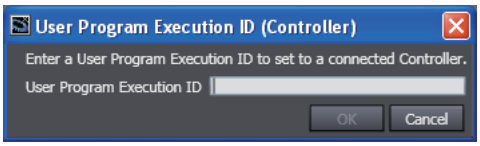
2 Assign the user program execution ID to the user program offline from the Sysmac Studio.



Precautions for Correct Use

After you assign a user program execution ID to a user program, you cannot change or delete the ID.
To use a different ID, read the project file without an ID that was backed up in step 1, above, and assign another user program execution ID. To delete the ID, use the project file without an ID that was backed up in step 1, above.

3 Connect the Sysmac Studio online and register the user program execution ID that was set in step 2 in the CPU Unit.



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

4 Transfer the user program with the same user program execution ID to the CPU Unit.
If the user program execution ID in the user program does not match the user program execution ID in the CPU Unit or if one of them does not have an ID, an ID Verification Error (major fault level Controller error) occurs when you attempt to change to RUN mode and the CPU Unit will not operate.

Precautions for Correct Use

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

Operation When an ID Verification Error Occurs

● When the User Program Execution ID in the CPU Unit Is Incorrect or Not Registered

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

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

2 Cycle the power supply to the Controller, or reset the CPU Unit from the Sysmac Studio.

● When the User Program Execution ID Is Not Assigned to the User Program or Is Incorrect

1 Read the backed up project file from the Sysmac Studio, and assign the correct user program execution ID.

2 Connect the Sysmac Studio to the CPU Unit online and transfer the user program.

3 Cycle the power supply to the Controller, or reset the Controller from the Sysmac Studio.

Other Situations

To Delete the User Program Execution ID Assigned to the User Program

Read the backed up project file in the Sysmac Studio.

To Delete the User Program Execution ID from the CPU Unit

Connect the Sysmac Studio to the CPU Unit online and perform the Clear All Memory operation.

To Check the User Program Execution ID Assigned to the User Program

For security, the user program execution ID that is assigned to the user program cannot be checked from the Sysmac Studio. Read the backed up project file in the Sysmac Studio and set the user program execution ID again.

To Check the User Program Execution ID in the CPU Unit

For security, the user program execution ID that is set in the CPU Unit cannot be checked from the Sysmac Studio. Perform the Clear All Memory operation and register the correct user program execution ID.

Specifications

● User Program Execution ID Verification Specifications

Timing of Verification

At startup, the CPU Unit compares the user program execution ID that is registered in the CPU Unit with the user program execution ID that is assigned to the user program.

Verification Conditions

The conditions for verifications are given in the following table.
“A” and “B” indicate the IDs.

User program execution ID that is registered in the CPU Unit	User program execution ID that is assigned to the user program	Error	Operation
A	A	None	Possible.
None	None		
None	A	ID Verification Error	Not possible.
A	None		
A	B		

Operation When the IDs Do Not Match

When the IDs do not match, an ID Verification Error (major fault level Controller error) occurs, and the CPU Unit does not operate.
However, to reset the error you must cycle the power supply to the Controller or reset the Controller from the Sysmac Studio.

● User Program Execution ID Character Specifications

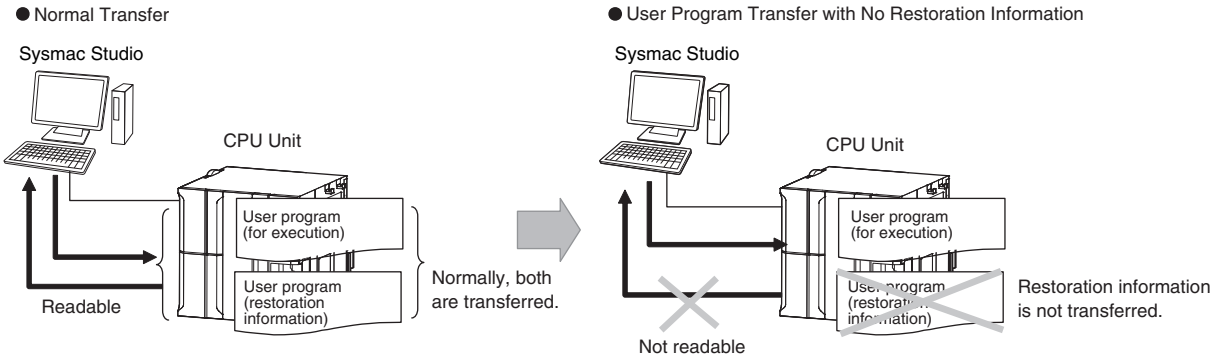
Usable characters	Case sensitivity	Maximum size (without NULL)
0 to 9, A to Z, and a to z	Case sensitive	8 to 32 characters

8-6-2 User Program Transfer with No Restoration Information

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

Introduction

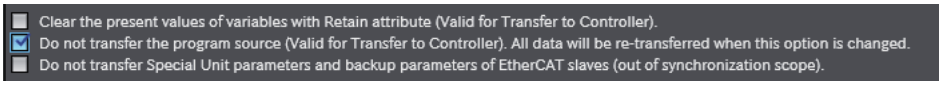
Normally, when you transfer the user program from the Sysmac Studio to the CPU Unit, information is transferred to restore it.
This function does not transfer information for restoration. That makes it impossible to read the user program.



This function is used to prevent theft of user program data when on-site maintenance of the user program is not required.

Operating Procedure

When you transfer the user program to the CPU Unit, select the *Do not transfer program source* Check Box in the Synchronization Window of the Sysmac Studio and then click the **Transfer to Controller** Button.

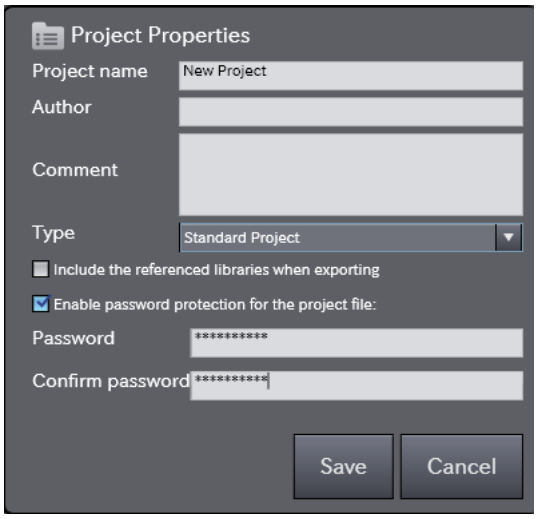


8-6-3 Overall Project File Protection

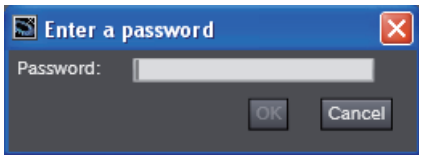
You can place a password on a project file to protect your assets.

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.

8-6-4 Data Protection

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

Version Information

A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use data protection.

Introduction

You can place protection on part of the data in a project file to restrict access to that data.
You can select any of three levels of access restrictions when you set protection.
Protection must be temporarily cleared to access the restricted data. The length of time for which protection is cleared depends on the operation that you use.

Protected Data

Protection can be set for the following data.

- 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 three levels of access restrictions. Only change protection can be set for cam profiles.

- 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.

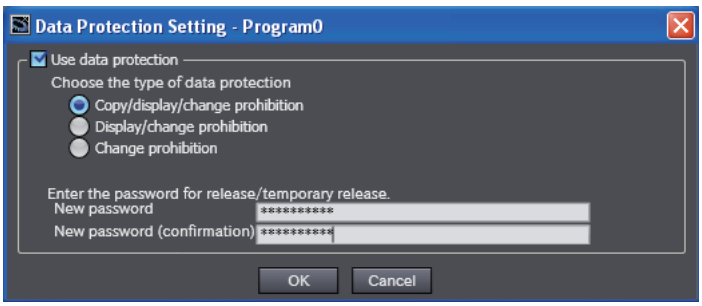
(○: Restricted. ---: Not restricted.)

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	○	○	---	While the project is open
Printing the data	○	○	---	
Changing the data	○	○	○	
Copying the data	○	---	---	Protection must be temporarily cleared for each operation
Displaying basic comparison results	---	---	---	This operation is not restricted.
Displaying detailed comparison results	○	○	---	While the project is open
Jumping from an event log or cross-reference to the data	○	○	---	
Registering objects in a library	○	---	---	Not possible to clear protection temporarily.

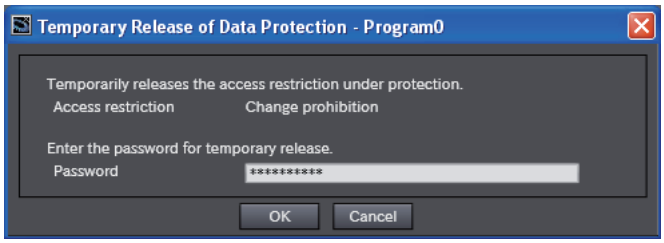
Operating Procedure

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.

8-6-5 Secure Communication Function

The secure communication function encrypts communication data between Sysmac Studio and the Controller and transmits and receives the data with a hash value. It can prevent data from being eavesdropped or tampered with by a third party.
Refer to *10-2-3 Secure Communication* on page 10-12 for more information on this function.

Version Information

Secure communication is available in the following CPU Units of the stated versions.
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

8-6-6 Secure Socket Services

The secure socket services perform secure socket communication encrypted using TLS (Transport Layer Security).
The CPU Unit can be used as a client to connect to cloud and on-premises servers via TCP/IP and exchange messages.
Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for more information on this function.

Version Information

An NX102-□□00 CPU Unit with unit version 1.46 or later or an NX102-□□20 CPU Unit with unit version 1.37 or later, and Sysmac Studio version 1.46 or higher are required to use secure socket services.
An NX1P2-□□□□□□ CPU Unit with unit version 1.46 or later and Sysmac Studio version 1.46 or higher are required.
An NX502- □□□□ CPU Unit with unit version 1.60 or later and Sysmac Studio version 1.54 or higher are required.

8-6-7 TCP/UDP Port Close Function

If you select the **Do not use** Option for the server function, such as FTP server, that is incorporated in the built-in EtherNet/IP Port, you can close the TCP/UDP port used by the server function. This allows you to open only TCP/UDP ports that you want to use in the system.
For details on the server function of the built-in EtherNet/IP port, refer to *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* .

8-6-8 Packet Filter

This function restricts access from external devices by filtering IP packets in the reception process of the built-in EtherNet/IP port.
While TCP/UDP port close function closes TCP/UDP ports to disable the corresponding server function, Packet Filter allows more detailed settings, such as allowing only certain clients to use specific server functions.
Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for more information on this function.

Version Information

The following CPU Units of the stated unit versions and Sysmac Studio version 1.50 or higher are required to use the Packet Filter.
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

8-6-9 Complete Data Erasure Function

The complete data erasure function is used to erase all the user-set data in the CPU Unit.
Since the data in the memory is not completely erased with the Clear all memory function, the data may be restored.
The complete data erasure function prevents data restoration by writing random data to unused areas of the built-in non-volatile memory that stores user programs and various settings after clearing all memory.
It will take a few minutes until the processing is completed because a random data is written to the unused areas of the built-in non-volatile memory.

Version Information

The following CPU Units of the stated unit versions and Sysmac Studio version 1.53 or higher are required to use the complete data erasure function.
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.60 or later
- NX701 CPU Unit: Version 1.32 or later
- NX502 CPU Unit: Version 1.60 or later

Precautions for Correct Use

When you perform a complete data erasure, the client private key, client certificate, and secure socket settings for the secure socket communications will also be erased.

Operating Procedure

1 Execute Clear All Memory operation.
Select the **Erase the data completely** Option in the **Memory All Clear** Dialog Box on the Sysmac Studio and click the **OK** Button.

2 Check the indicators on the front panel of the CPU Unit to see that complete data erasure has been completed.
After the complete data erasure is completed, the CPU Unit indicators indicate the normal end or error end of the complete data erasure. Refer to *Indicator Specifications* on page 8-50 for specifications of indicators on the front panel of the CPU Unit when a complete data erasure is completed.
After the complete data erasure ended normally, the CPU Unit starts in the factory default condition when the power supply to the CPU Unit is cycled.

Indicator Specifications

The indicators on the front panel of the CPU Unit can be used to check whether a complete data erasure is in progress, the process has been completed normally, or the process has terminated abnormally.

● During Complete Data Erasure Process

The following table gives the specifications of indicators when complete data erasure is being executed.

CPU Unit operation status indicator

After the RUN: Not lit, ERR: Not lit, and BUSY: Flashing state, then the following state appears.

Indicator	Color	Behavior
RUN	Green	Flashing
ERROR	Red	Not lit
BUSY	Yellow	Flashing

Operation status indicators of built-in EtherNet/IP port 1

Indicator	Color	Behavior
NET RUN	Green	Each indicator lights up in turn for 1 second in the order of LINK/ACT → NET ERR → NET RUN.
NET ERR	Red	
LINK/ACT	Yellow	

● Normal End

The following table gives the specifications of indicators when complete data erasure ended normally.

CPU Unit operation status indicator

Indicator	Color	Behavior
RUN	Green	Flashing
ERROR	Red	Not lit
BUSY	Yellow	Not lit

Operation status indicators of built-in EtherNet/IP port 1

Indicator	Color	Behavior
NET RUN	Green	Lights.
NET ERR	Red	Lights.
LINK/ACT	Yellow	Lights.

● Error End

The following table gives the specifications of indicators when complete data erasure ended in an error.

CPU Unit operation status indicator

Indicator	Color	Behavior
RUN	Green	Not lit
ERROR	Red	Lit

Operation status indicators of built-in EtherNet/IP port 1

Indicator	Color	Behavior
NET RUN	Green	OFF
NET ERR	Red	OFF
LINK/ACT	Yellow	Not lit

Actions for Error End

If complete data erasure ended in an error, cycle the power supply to the CPU Unit. If the error end persists, consult your OMRON representative.

8-6-10 Operation Authority Verification

Introduction

Online operations are restricted by operation rights to prevent damage to equipment or injuries that may be caused by operating mistakes.
Examples are shown below.

- I/O Monitor: Writing, forced refreshing, etc.
- Controller operations: Changing the operating mode, online editing, MC Test Run, etc.

You can register passwords for operation authority for each CPU Unit in the Sysmac Studio. If a correct password is entered when an online connection is made to a Controller, the online operations for the operation authority category for the password that was entered will be allowed.

The Administrator sets a password for each operation authority. Users are notified of the operation authority name and password according to their skills.

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

Operating Procedure

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.

Type	Password
Administrator	Required.
Designer[*1]	Optional[*2] Whether a password is required is determined by the default operation authority that is set in the **Setting of Operation Authority** Dialog Box.
Maintainer	
Operator[*1]	
Observer[*1]	Not required.

*1. A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required.
*2. Whether a password is required is determined by the default operation authority that is set in the **Setting of Operation Authority** Dialog Box. A password must be entered to perform operations that require an operation authority that is higher than the default operation authority. A password is not required to perform operations that require an operation authority that is equal to or lower than the default operation authority.

● Examples of Online Operations for Operation Rights

Examples of the online operations that are allowed for each operation authority are given below.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details.
(OK: Operation possible, VR: Verification required for each operation, NP: Operation not possible)

Status monitor (example)	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 for troubleshooting	OK	OK	OK	VR	NP

Controller operations (examples)	Administrator	Designer	Maintainer	Operator	Observer
Starting or restarting an MC Test Run	OK	OK	VR	NP	NP
User program execution IDs for Controllers	OK	NP	NP	×	NP
CPU Unit write-protection	OK	OK		NP	NP

● Password Specifications

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

8-6-11 User Authentication

Outline

The user authentication function assigns individual user names and passwords to users of the CPU Unit and identifies who operated the CPU Unit online. When an online connection is made to the CPU Unit, the user will be asked to enter user name and password. Online connections cannot be made unless the user name and password match the predefined settings.
Furthermore, for each user, operation authority of any of administrator, designer, maintainer, operator and observer is assigned. This ensures that users can operate the CPU Unit online within the authorities assigned to them. Refer to the *8-6-10 Operation Authority Verification* on page 8-51 for more information on operation authority verification.
To place the CPU Unit online for the first time or to do so after clearing all memory, you need to be registered as an administrator. Unless being registered as the administrator, you cannot place the Unit online.

The user authentication function records user names in several event logs. This is useful to prevent denial when a security problem occurs. For more information on events for which user names are logged, refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)*.

You can transfer and save the user account settings, including the user names, passwords, and operation authority properties of the user in the CPU Unit. Therefore, you can use the user authentication function even when another computer is connected to the CPU Unit.
However, user account settings are not included in the Sysmac Studio project. Therefore, you must configure the user authentication settings for each CPU Unit that you want to use with user authentication.

Version Information

- The user authentication function is available in version 1.50 or higher of Sysmac Studio.
- User authentication is available in the following CPU Units of the stated unit versions.
  a) NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
  b) NX701 CPU Unit: Version 1.29 or later
  c) NX502 CPU Unit: Version 1.60 or later

Operating Procedure

Entering your user name and password when connecting Sysmac Studio online to the CPU Unit will allow you to operate within the permissions assigned to you. Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for specific procedures.

Precautions for Correct Use

If user account settings is not enabled on the CPU Unit, then if a third party enables user account settings on the CPU Unit, online operations will be disabled unless you enter the user name and password. Make an agreement with the user in advance whether to set up user account settings.

User Account Settings

To use the user authentication function, it is necessary to set the user account settings in advance. User account settings assign each user a user name, password, and type of operation authority.
You can also set the operation lock. Operation lock is a function that requires you to re-enter the user name and password if you do not operate the support software for a set period of time after connecting the support software online to the Controller.
You can set an expiration date for the password. If your password expires, you will be prompted to change your password when you connect Sysmac Studio online.
You can also set how many days in advance to notify that the password will expire. On the set date, you will be prompted to renew your password when you connect the Sysmac Studio online.

Major category	Minor category	Description	Setting range	Default
User account	Number of users*1	Number of users to register	1 to 64	---
	User name	User name	1 to 16 characters*2	---
	Password	Password	8 to 32 characters*2	---
	Operation authority*3	Operation authority to assign to users	Administrator, Designer, Maintainer, Operator, and Observer	Administrator
	Enable/Disable	Whether to enable or disable user authentication for the user	Enable or Disable	Enable
Operation lock	Enable/Disable	Setting about whether to enable or disable operation locking	Enable or Disable	Disable
	Inactive time	Set time for operation lock	1 to 360 minutes	10 minutes
Password	Valid for	Password validity period	14 to 730 days	None Can be used without time limit
	Expiration notice	How many days before to notify that the password will expire.	1 to 30 days	No prior notice

*1. You cannot register more than one same user name.
*2. Usable characters include 0 to 9, A to Z, and a to z.
*3. At least one *Administrator* must be registered to transfer the user account settings to the CPU Unit.

The user account settings must be made for each CPU Unit. They are not saved in the Sysmac Studio project file. These settings can only be executed by users with administrator privileges, except for initial settings.
For the details on the settings, refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*.

Other Specifications

This section describes other specifications of user authentication function.

● Adding and Deleting Users

Administrators can register and delete users in the user account settings of the Sysmac Studio. However, Administrators cannot delete their own user registration. If an administrator wants to delete his or her own user registration, please ask another administrator to delete it. Deleting a user registration during online operation will force the user's online connection to be disconnected.
For details on user registration and deletion of users, refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*.

● Checking User Information

Administrators, designers, and maintainers can see all user names and operation authorities registered in the CPU Unit. The password cannot be checked.
All users can see their own user name and operation authority. The password cannot be checked.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for the procedure to see the user name and operation authority.

● Changing User Account Settings

The specifications for changing user account settings are as follows.

Items to be changed	Users who can change	Specification
Username	Administrator	It is not possible to modify the user name. Delete the registered user name and register a new user name.
Password	All users	• The user can change his or her own password from the Password change Dialog Box. • Changing password will force the online connection to be disconnected.
	Administrator	• The administrator can change passwords of all users from the User account settings Dialog Box. • Changing a user password during online operation will force the user's online connection to be disconnected.
Operation authority	Administrator	The administrator can change operation authorities of all users in the User account settings Dialog Box
Operation lock	Administrator	The administrator can change the setting for operation lock from the User account settings Dialog Box.
Entire user account settings	Administrator	• Changing the user account setting will force the user's online connection to be disconnected. • Performing clear all memory operation clears the entire user account settings saved in the CPU Unit. When the entire user account settings are cleared, user authentication will be disabled.

● Clearing User Account Settings

You can clear the user account settings in any of the following ways. Perform this to prevent information leaks when you dispose of the CPU Unit.

- Perform Clear All Memory operation.
- Specify an early project unit version that does not support user authentication function to the CPU Unit.

● What to Do If You Forget Your Password?

If a user forgets his or her password, there is no way to find it. Also, if you do not have administrator operation privileges, you cannot perform Clear All Memory operation.

In a case Administrator forgets his or her password, use one of the following troubleshooting method.

Troubleshooting	State after troubleshooting
Prepare the backup file without user account settings beforehand, and restore the settings using the SD Memory Card backup function.	User authentication is disabled.
Prepare the backup file, which contains user account settings and Administrator's user name and password settings, in advance, and use it for restoring the settings using the SD Memory Card backup function.	The settings in the backup file are overwritten and applied.
Start the CPU Unit in Safe Mode and connect the Sysmac Studio online. Perform Clear All Memory operation according to **When You Cannot Log In** in the Authentication (Safe Mode) dialog that appears. For information on how to start in Safe Mode, refer to the description on *DIP switch* settings in the hardware user's manual for each CPU Unit.	The user account settings are cleared together with a user program, etc.

Precautions for Correct Use

- The backup file must be prepared beforehand for the above actions. Note that these backup files cannot be created after the Administrator forgets the password.
- If you execute the above measures, the user program before execution will be lost.

Availability of User Authentication Function Depending on the Combination of Versions

The following table shows the availability of user authentication function for different project unit versions of the CPU Unit and Sysmac Studio versions.

		CPU Unit			
		Unit version supporting user authentication function			Earlier version not supporting user authentication function
		Project unit version not set (Factory default or memory all cleared)	Project unit version supporting user authentication function	Earlier project unit version not supporting user authentication function	All project unit versions
Sysmac Studio	Ver.1.50 or higher	Possible.*1	Possible.	Not possible.	Not possible.
	Lower than Ver.1.50	Not possible.*1	Not possible.		

*1. When user authentication is enabled when the project unit version is not set (factory default or memory all cleared), Sysmac Studio version 1.50 will be connected after user authentication. However, Sysmac Studio lower than version 1.50 can connect without user authentication. Set the project unit version of the CPU Unit that supports user authentication function.

Version Information

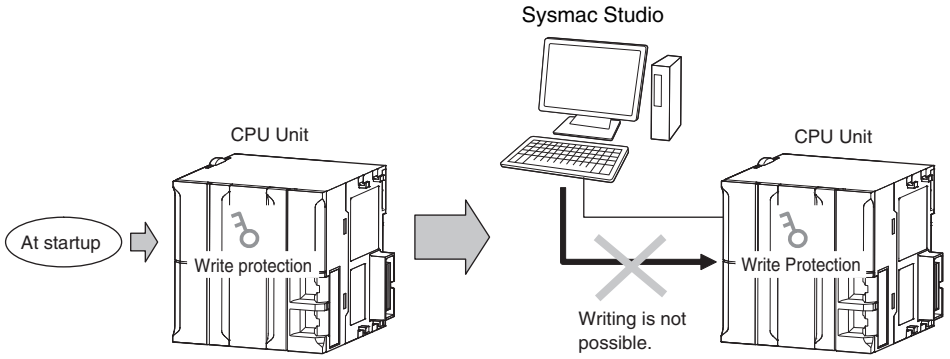
User authentication is available in the following CPU Units of the stated unit versions.
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

8-6-12 CPU Unit Write Protection

This function disables the ability to write data to the CPU Unit 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.

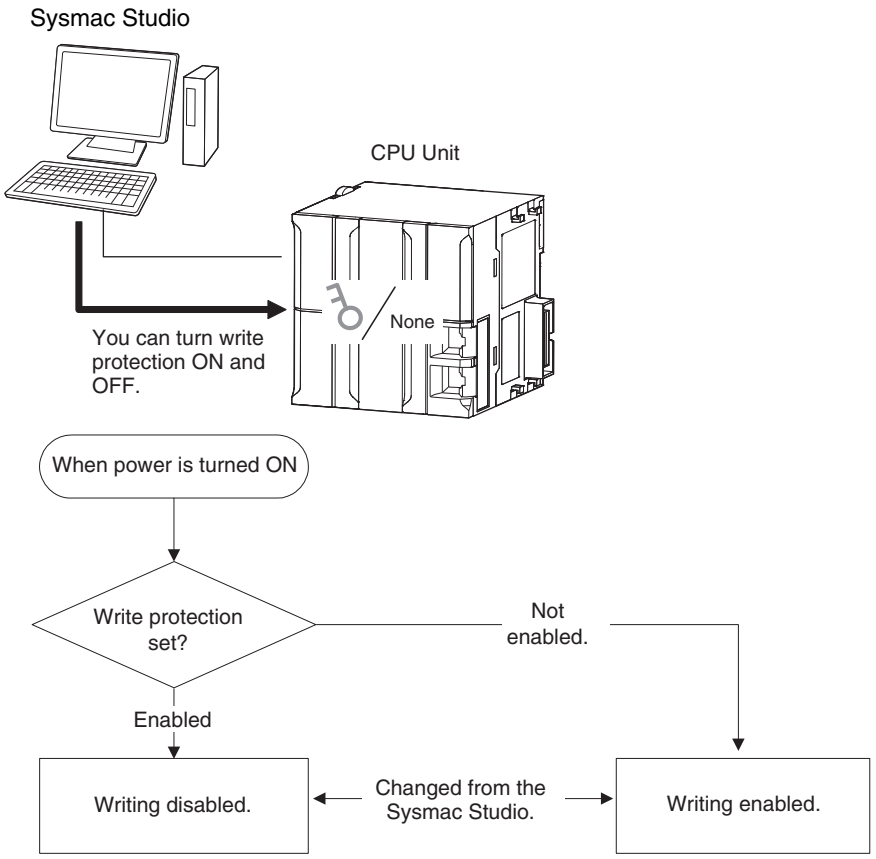


Set whether to automatically enable write protection when the power supply is turned ON in the **Operation Settings** under the **Configurations and Setup** - **Controller Setup** of the Sysmac Studio.

Access point	Setting group	Setting	Description	Set value
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.



8-6-13 Firmware Update Prohibition

Contact your OMRON representative for more information on the firmware update.
The firmware update prohibition prevents the firmware from being updated. Set whether firmware update is prohibited or not on the Sysmac Studio. For details on the settings, refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*.

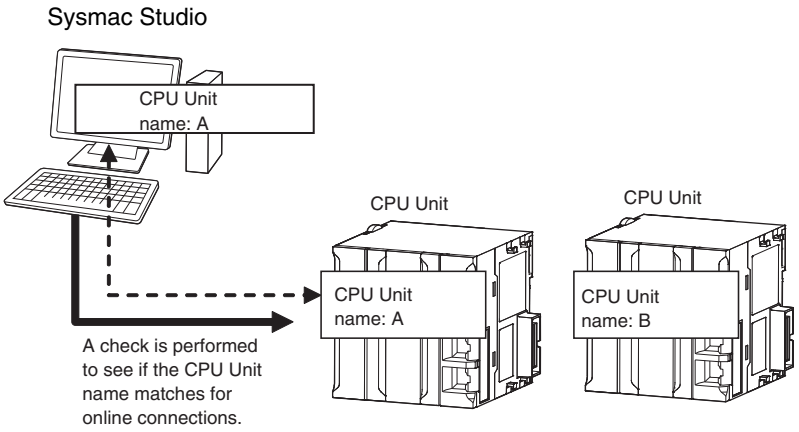
Version Information

The following CPU Units of the stated unit versions and Sysmac Studio version 1.53 or higher are required to use the firmware update prohibition function.
- NJ-series, NX502, NX102, and NX1P2 CPU Unit: Version 1.60 or later
- NX701 CPU Unit: Version 1.32 or later

8-6-14 CPU Unit Names and Serial IDs

Introduction

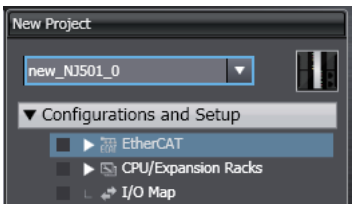
Register a CPU Unit name in the CPU Unit.
When going online to a CPU Unit from the Sysmac Studio, the CPU Unit name in the project is compared to the name of the CPU Unit being connected to.
This helps prevent incorrect connections to the CPU Unit from the Sysmac Studio. It is particularly effective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



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

Setting Methods

1 Set the CPU Unit name when you create a project on the Sysmac Studio.
The CPU Unit name is displayed as shown below.



To change the name, right-click the Controller icon and select **Rename**.

2 When you first connect to the CPU Unit online, the Sysmac Studio prompts you to store the CPU Unit name in the CPU Unit.

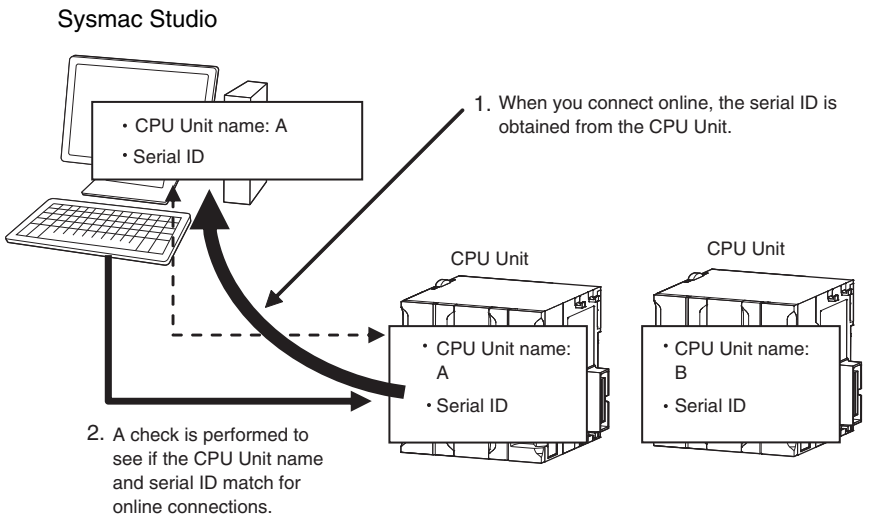
3 After that, when you connect to the CPU Unit online, the Sysmac Studio refers to the CPU Unit name in the project and the CPU Unit name of the CPU Unit you connect to. A warning dialog box is shown if they do not match, and you are asked whether to continue to connect.

Additional Information

You can name EtherNet/IP ports in the Network Configurator.

Serial IDs

When the Sysmac Studio first connects online, you can obtain the serial ID from the CPU Unit's production information and store it in the project.
After that, when the Sysmac Studio connects online, both the CPU Unit name and the serial ID are compared. This enables stricter verification of the CPU Unit.



8-6-15 Access log

Operations that the user performed to the Controller using the tool are registered as access logs. If the user authentication function is used, the user name is registered in the access log that is important for security.
This allows you to see who performed, when, and what operations, and can be used as a repudiation preventative measure when problems occur.
For the access log to register, refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)*.

8-7 Debugging

This section describes debugging.

The NJ/NX-series Controller provides the following debugging operations.

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

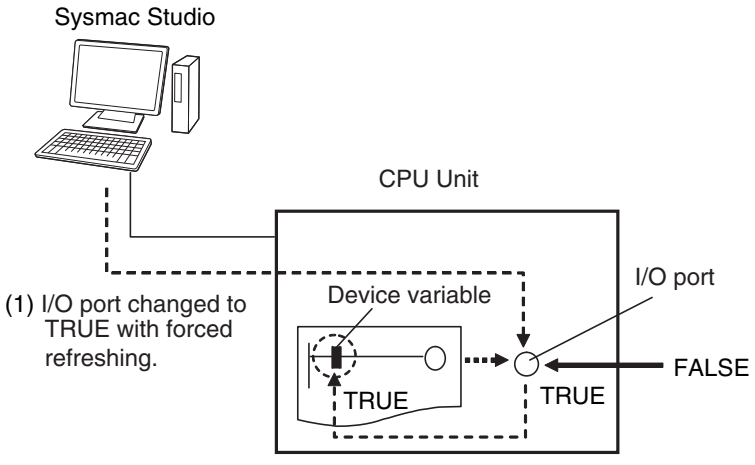
8-7-1 Forced Refreshing

Introduction

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 8-64 for information how forced refreshing is retained or cleared according to changes in CPU Unit status.)
All forced refreshing is cleared when a fatal error occurs, when a Clear All Memory operation is performed, when the operating mode is changed, when power is interrupted, or when the project is downloaded.

● Inputs

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

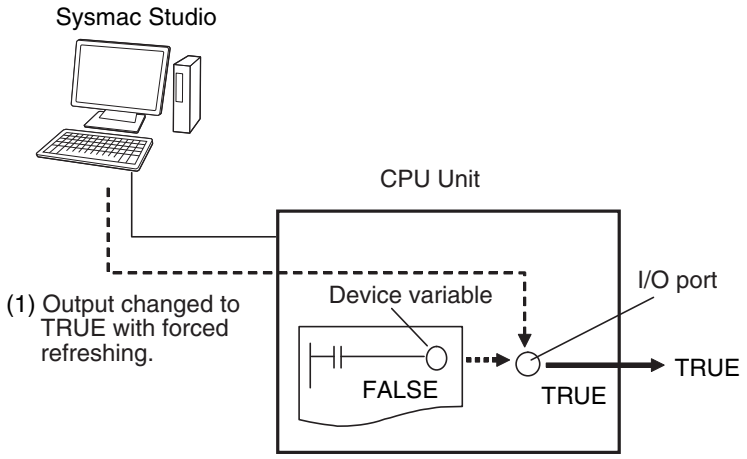


(2) I/O port and device variable change to TRUE.

● 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 and memory used for CJ-series Units.

- I/O ports for EtherCAT slaves
- I/O ports from data handled by the NX Bus Function Module of the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit
- I/O ports from data handled by the X Bus Function Module of the NX502 CPU Unit
- I/O ports for I/O built in the NX1P2 CPU Unit
- I/O ports for Analog I/O Option Boards in the NX1P2 CPU Unit
- I/O ports for CJ-series Basic I/O Units
- I/O ports for CJ-series Special Units
- I/O bits for DeviceNet or CompoNet slaves that are specified for AT specifications from variables

If you execute forced refreshing from the Ladder Editor or the Watch Tab Page, the status of the I/O port or memory element for a CJ-series Unit will change via the variable.

Note For I/O ports from data handled by the NX Bus Function Module of the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, only the I/O ports for I/O data for individual NX Units are applicable. The I/O ports for status of NX Units managed by the NX Bus Function Module as the NX bus master are not applicable.

Note You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

Number of Simultaneous I/O for Forced Refreshing

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

- CJ-series Units: 64 points total

• EtherCAT slaves: 64 points total

The number of external I/O points are given for the above limits. For example, if more than one variable is assigned the same external I/O point as the AT specifications, it is counted as only one point.

Application

● Inputs

- To apply a simulated input signal to debug the user program
- To create a status that would occur only when a failure occurs (e.g., two exclusive bits turning ON or OFF at the same time)

● Outputs

- To turn outputs ON and OFF to check wiring
- To intentionally turn OFF an output you do not want to operate regardless of results of user program execution

Operating Procedure

Operations can be performed from the following panes.

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

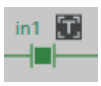
● Procedure for Forced Refreshing from Ladder Editor

1 Select **Monitor** from the Controller Menu. The monitor turns ON.

2 Double-click the ladder program, ladder function, or ladder function block under **Programming** in the Multiview Explorer.
The rungs are displayed on the Ladder Editor in monitor status.

3 Right-click the input or output and select **Forced Refreshing** − **TRUE**. The input or output is forced to TRUE. Right-click the input or output and select **Forced Refreshing** − **FALSE**. The input or output is forced to FALSE.

4 The input or output in the Ladder Editor changes to TRUE or FALSE and the execution condition changes accordingly.
A mark that indicates that the input or output has forced status is displayed as shown below.



Ladder diagram

The TRUE or FALSE mark for forced status indicates the status that was specified for forced refreshing. It does not indicate the current value of the input or output.

Forced status mark	Operation
	TRUE specified with forced refreshing
	FALSE specified with forced refreshing

Additional Information

If there are other variables that are assigned the same memory address as one that is specified as the AT specification of a variable for which forced refreshing is specified, the forced status mark is displayed for all of the variables with that AT specification.

Affect of Operating Modes and Power Interruptions

● Operating Modes for Forced Refreshing

You can execute forced refreshing in either PROGRAM mode or RUN mode.

Forced refreshing is not possible while there is a major fault level Controller error.

● Status of Forced Refreshing during Operating Mode Changes or Power Interruptions

By default, the forced refreshing is cleared when the operating mode changes between RUN mode and PROGRAM mode and when the power is interrupted.

Holding/Clearing Forced Refreshing

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

Change in status		Forced refreshing status
When power is turned ON		Cleared
When operating mode changes	**RUN to PROGRAM mode** **PROGRAM to RUN mode**	Cleared
After downloading		Cleared
When a major fault level Controller error occurs		Cleared
During online editing		Retained

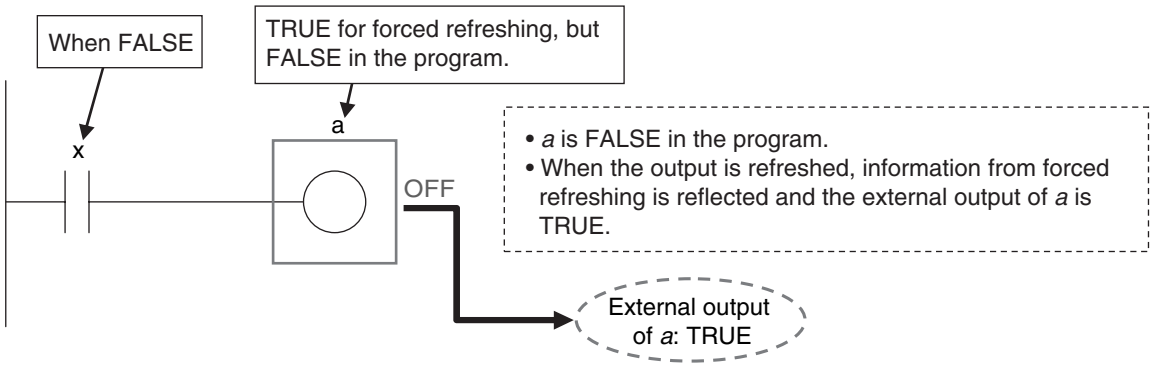
Programming Precautions for Forced Refreshing

If forced refreshing is set in the user program, the status of variables for which forced refreshing is specified are overwritten by the user program. Therefore, the status that is specified for forced refreshing is not maintained in the user program.

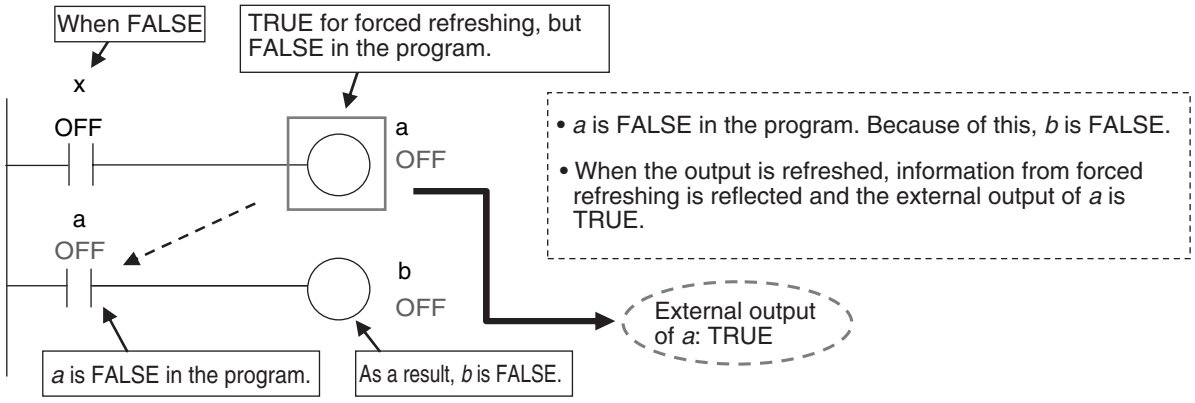
However, refreshing to external devices uses the values that were specified for forced refreshing, and not the status of the variables in the user program.

If forced refreshing is used in a program, the values of variables in the program may be different from the status of the external outputs.

Example: When *a* Is Refreshed to TRUE with Forced Refreshing



When There Is Another Input That Is Controlled by the Forced Input



Precautions for Safe Use

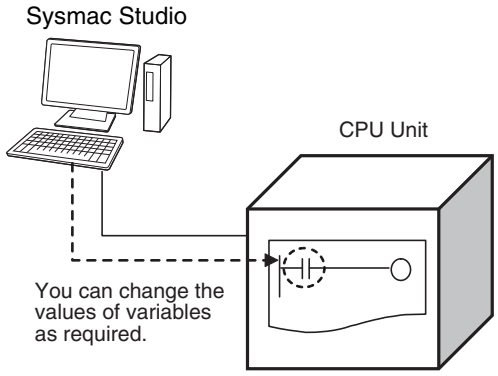
- Confirm that no adverse effect will occur in the system before you use forced refreshing.
- Forced refreshing ignores the results of user program execution and refreshes I/O with the specified values.
  If forced refreshing is used for inputs for which I/O refreshing is not supported, the inputs will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

Depending on the difference in the forced status, the control system may operate unexpectedly.

8-7-2 Changing Present Values

Introduction

You can change the present values of variables that are used in the user program and settings and you can change program inputs and outputs to TRUE or FALSE. This allows you to check the operation of the user program and settings.



Precautions for Safe Use

Always confirm the safety of the system before you change the present value of a variable.

Application

● Changing Program Inputs and Outputs to TRUE or FALSE

You can change the value of any BOOL variable to TRUE or FALSE. The specified value is then overwritten by the execution results of the user program. If the operating mode is changed or the power supply is cycled, the initial value is restored.

You can control BOOL variables in the Ladder Editor, Watch Tab Page, or I/O Map.

● Changing the Values of Other Variables

You can change the present values of user-defined variables, system-defined variables, and device variables as required. You can do this on a Watch Tab Page.

Precautions for Safe Use

Always confirm the safety of the system before you change the present value of a variable.

Operating Procedure

Operations can be performed from the following panes to change the present values. Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on the operating procedures on the panes.

- Program panes (ladder diagrams and ST)
- I/O Map
- Watch Tab Page

Precautions on Changing the Status of Outputs Assigned to External Devices by Changing Present Values

Observe the following precautions when you change the status of an output that is assigned to an I/O port of a CJ-series Basic Output Unit or EtherCAT output slave by changing a present value.

● Changing Present Values in the I/O Map in RUN Mode

Any value of an I/O port that is changed in the I/O Map is then overwritten by the execution results of the user program.

The value that was specified by changing the present value is not output to the external device.

To change the value of an I/O port and output that value to an external device, use forced refreshing.

● Changing Present Values in a Watch Tab Page in PROGRAM Mode

The value that was specified in a Watch Tab Page by changing the present value of a device variable* that is defined as an external or local variable is not output to the external device.

To output a specified value to an external device, do one of the following:

- Use forced refreshing.
- Change the present value in a Watch Tab Page of a device variable* that is defined as a global variable.

* The devices variables must be assigned to an I/O port of a CJ-series Basic Output Unit or EtherCAT output slave. This also applies to a global variable with an AT specification to an output bit that is assigned to a CJ-series Basic Output Unit.

● Precaution When Directly Writing to I/O Memory Addresses Assigned to Output Bits for CJ-series Basic Output Units

Any value that is written to an I/O memory address that corresponds to an output bit that is assigned to a CJ-series Basic Output Unit through a tag data link will be overwritten by the execution results of the user program.

The value that is written directly to the I/O memory address from the tag data link will therefore not be output to the external device.

Note You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

8-7-3 Online Editing

This section introduces online editing. Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details.

Introduction

The online editing function is used to add to or change part of a program in the CPU Unit directly from the Sysmac Studio.

You can select any of the following to perform online editing.

- POUs (programs, functions, and function blocks)
  For a ladder diagram program, select a section.
- Global variables

Application

You can use online editing to change a user program without stopping the operation of the CPU Unit.

Sysmac Studio Operations

● Performing Online Editing

1. Select the item to edit online.
2. Select **Online Edit** - **Start** from the Project Menu.
3. Make the required changes.
4. Select **Online Edit** - **Transfer** from the Project Menu.
5. Check the results.
6. The user program will begin operation after online editing.

⚠ CAUTION

Execute online editing only after confirming that no adverse effects will occur if the I/O timing is disrupted. If you perform online editing, the task execution time may exceed the task period, I/O may not be refreshed with external devices, input signals may not be read, and output timing may be changed.

Precautions for Correct Use

- The differentiation status of differentiated instructions in a program that is edited online is initialized.
- When online editing changes are applied, the execution times of the tasks are extended. Set the task period appropriately so that you do not cause a Task Period Exceeded error due to online editing.
- If the power supply to the Controller is interrupted when online edits are being saved,* a major fault level Controller error (User Program/Controller Configurations and Setup Transfer Error, Incorrect User Program/Controller Configurations and Setup, or Non-volatile Memory Restored or Formatted) occurs. If one of these errors occurs, download the user program again.
- Do not execute the MC_SaveCamTable instruction while online edits are being saved.* Otherwise the online edits may not be saved correctly.

* Online edits are saved from when you click the **Yes** Button in the confirmation dialog box until the Online Editing Pane closes. However, with a CPU Unit with a unit version of 1.04 or later and Sysmac Studio version 1.05 or higher, saving continues until the dialog box that indicates saving data to built-in non-volatile memory (which is displayed after the confirmation dialog box) closes.

8-7-4 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.

Precautions for Correct Use

If you use data tracing to sample following variables, correct data may not be sampled.
- Structure members whose data size is 16 bits or more, except for system-defined variables for motion control
- Array elements whose data size is 16 bits or more

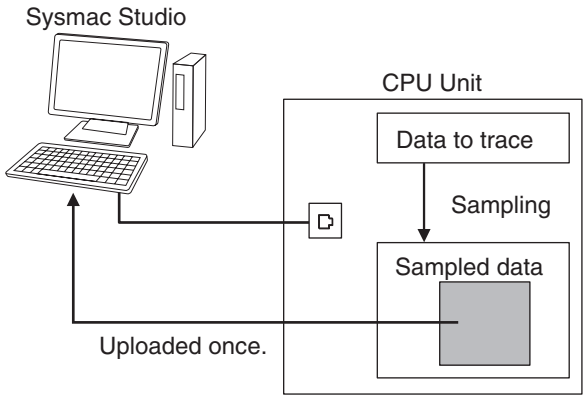
If you sample the data with the above variables, perform either of the followings.
- Copy the above variables to the internal variables with a basic data type other than a data size of 64 bits, and trace data for the copied variables.
- Use the settings for exclusive control of variables in tasks, and the task for which you use data tracing to sample is set as a refreshing task.

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 on-line, you can trace data when trigger conditions are met and then upload the data after placing the Sysmac Studio online.

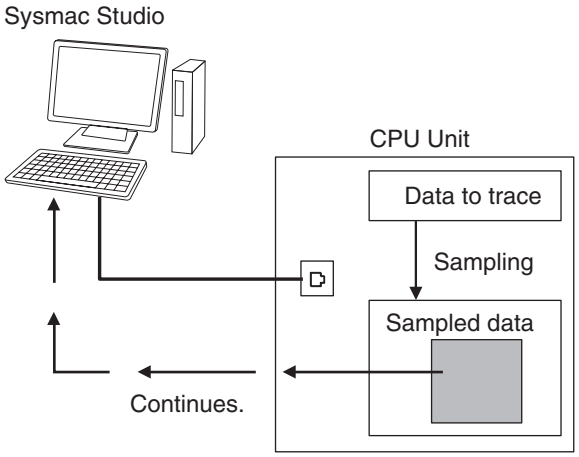
- You can check the flow of the program based on the status of changes in the present values of variables.
- You can use the data to investigate the cause of unexpected changes in the values of variables.



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 a 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	**Triggered tracing**	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. *1
	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*2	**Maximum number of targets**	NX701-□□□□: 192 variables NX502-□□□□: 192 variables NX102-□□□□: 48 variables NX1P2-□□□□: 48 variables NJ501-□□□□: 192 variables NJ301-□□□□: 48 variables NJ101-□□□□: 48 variables
	Data types	Basic data types except for text strings Arrays (specify the element) Enumerations Members of structures and unions
Maximum number of records		10,000 samples per variable
Setting trigger positions		The trigger position is set in respect to the overall trace time or quantity.

Item		Description
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*3	Tracing is started when one of the following conditions is met. BOOL: when it changes to TRUE or FALSE Non- BOOL: Equals (=), Greater than (>), Greater than or equal (≥), Less than (<), Less than or equal (≤), Not equal (≠)*4
	Command 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	**Command 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).
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 *stop 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

Item		Description
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*5		NX701-□□□□: 4 traces NX502-□□□□: 4 traces NX102-□□□□: 2 traces NX1P2-□□□□: 2 traces NJ501-□□□□: 4 traces NJ301-□□□□: 2 traces NJ101-□□□□: 2 traces
Importing trace data		You can display CSV format trace results on top of the current graph.
Saving		You can save the trace results in the project along with the trace settings.
Printing		You can print graphs. The Sysmac Studio's printing functionality is used.

*1. For an NJ-series CPU Unit and NX701 CPU Unit, if you perform an internal variable with a data size of 64 bits or more to the sampled data, specify the task that assigns the program in which the target internal variable is defined. If the different task is specified, the concurrency of variable values may not be ensured.

*2. 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.

*3. Data tracing will not start when data trace is started even if the trigger condition is met.

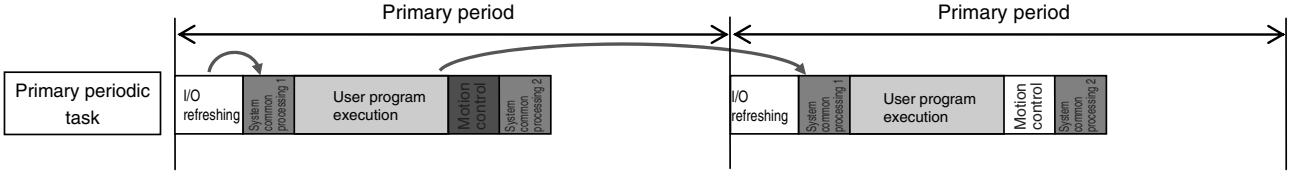
*4. Combinations of multiple condition expressions are not permitted. Also, the valid range for comparison constants is determined by the valid range of the literal expressions for the variable type on the left side of the condition expression.

*5. Trace numbers 0 to 3 are set for the NX701, NX502, and NJ501. Trace numbers 0 and 1 are set for the NX102, NX1P2, NJ301, and NJ101. These numbers are used to execute instructions and to access system-defined variables.

Data Trace Operation

Processing for data traces (sampling and trigger checking) 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.



Additional Information

I/O refreshing, user program execution, and motion control processing are all executed in the same task period. For data tracing, user program execution and motion control processing for the current task period and I/O refreshing for the next task period are displayed at the same time. The timing charts in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* are based on the task periods, so the display are not the same as those for data tracing.

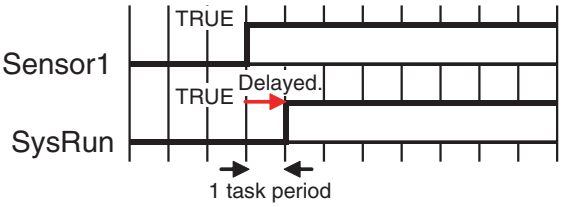
Display examples for data trace operations and execution results is given below for sampling in a specified task period.

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*.

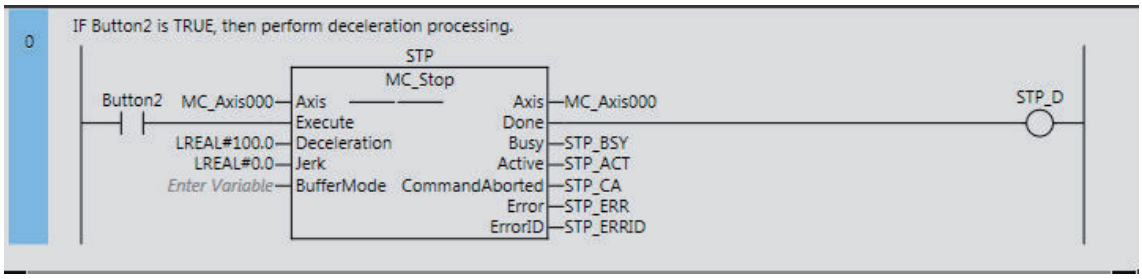
Therefore, in the data trace display, *SysRun* is shown as TRUE one task period after *Sensor1*.
Data Trace Display



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.

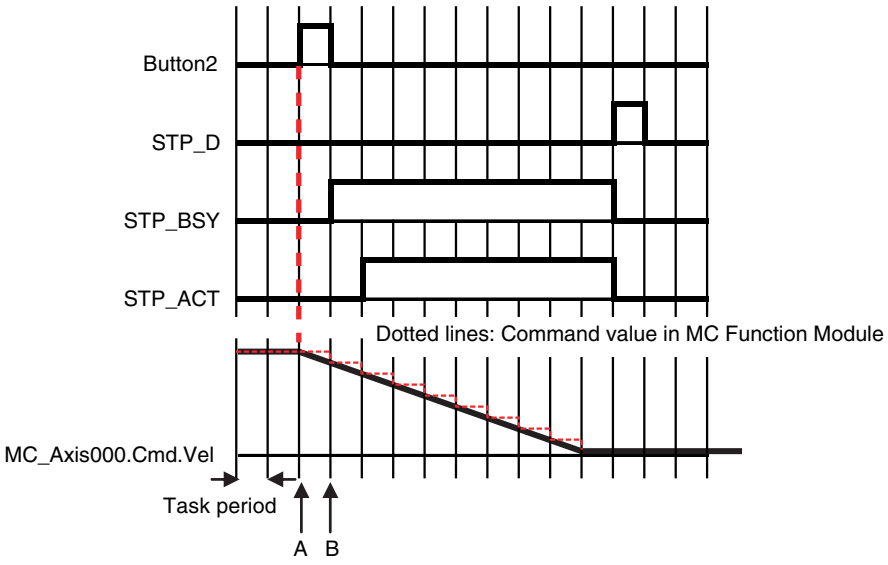
Example 2:
When the *Button2* variable (assigned to an input signal from a pushbutton) changes to TRUE during velocity control, the user program in this example decelerates axis 0 (*MC_Axis000*) to a stop.



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 variables, and output parameters and output variables, change in the same task period.

Related System-defined Variables

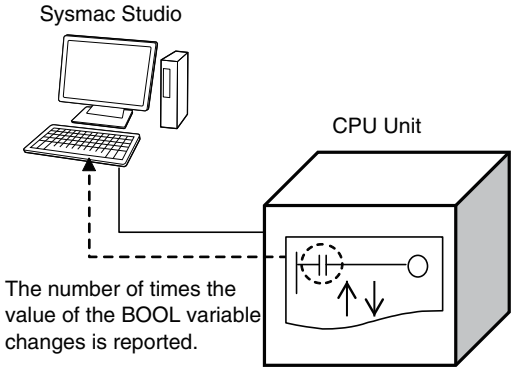
Variable name		Meaning	Function	Data type	R/W
	Member name				
_PLC_TraceSta[0..3]*1				_sTRACE_STA	R
	.IsStart	Trace Busy Flag	TRUE when a trace starts.	BOOL	R
	.IsComplete	Trace Completed Flag	TRUE when a trace is completed. Changes to FALSE when the next trace starts.	BOOL	R
	.IsTrigger	Trace Trigger 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

*1. 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.

8-7-5 Differential Monitoring

Differential monitoring reports the number of times the value of the specified BOOL variable matches the specified condition. The specified condition is evaluated for a match in every task period of the primary periodic task (called the primary period). Differential monitoring provides a running total of the number of times the condition is matched.



Version Information

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

Application

You can use differential monitoring to check or count the number of times an external input signal turns ON or OFF or an 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
Operation 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 cases.

- 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 disconnected

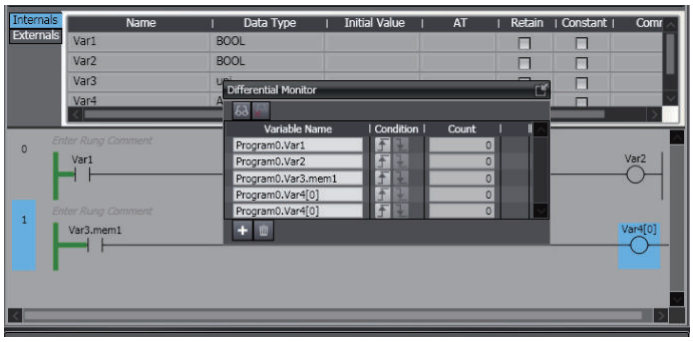
Operating Procedure

Use the following procedures to control differential monitoring. Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details.

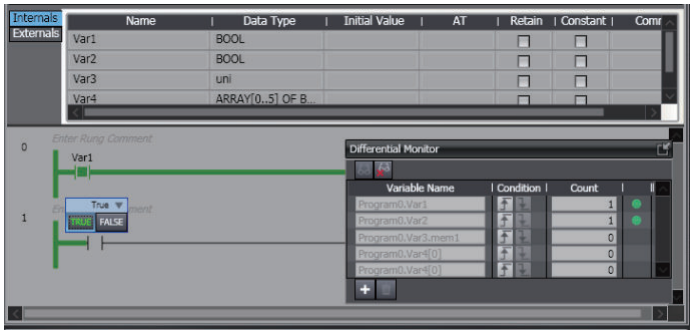
1 Select **Differential Monitor** from the View Menu on the Sysmac Studio.

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

3 Set the differential monitoring condition expression for each variable in the Differential Monitor Window.



4 Execute the user program.
The number of times that the condition is met for each variable is displayed in the Differential Monitor Window.



Precautions for Differential Monitoring

Observe the following precautions when you use differential monitoring.

● Loss of Communications with Sysmac Studio While Differential Monitoring Is in Progress

Let's assume that communications with the Sysmac Studio were cut off during differential monitoring because the communications cable was disconnected or because the Sysmac Studio ended due to an error. In such cases, the CPU Unit will continue execution of differential monitoring. To restart execution of differential monitoring, you must resume communications with the Sysmac Studio and stop differential monitoring.

● Simultaneous Execution of Differential Monitoring

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

● Specifying Global Variables and External Variables

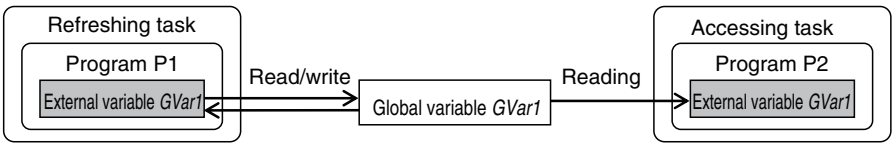
You can specify global variables or external variables (which specify global variables in POUs) for differential monitoring. Keep in mind that the values of global variables and external variables are updated at different times.

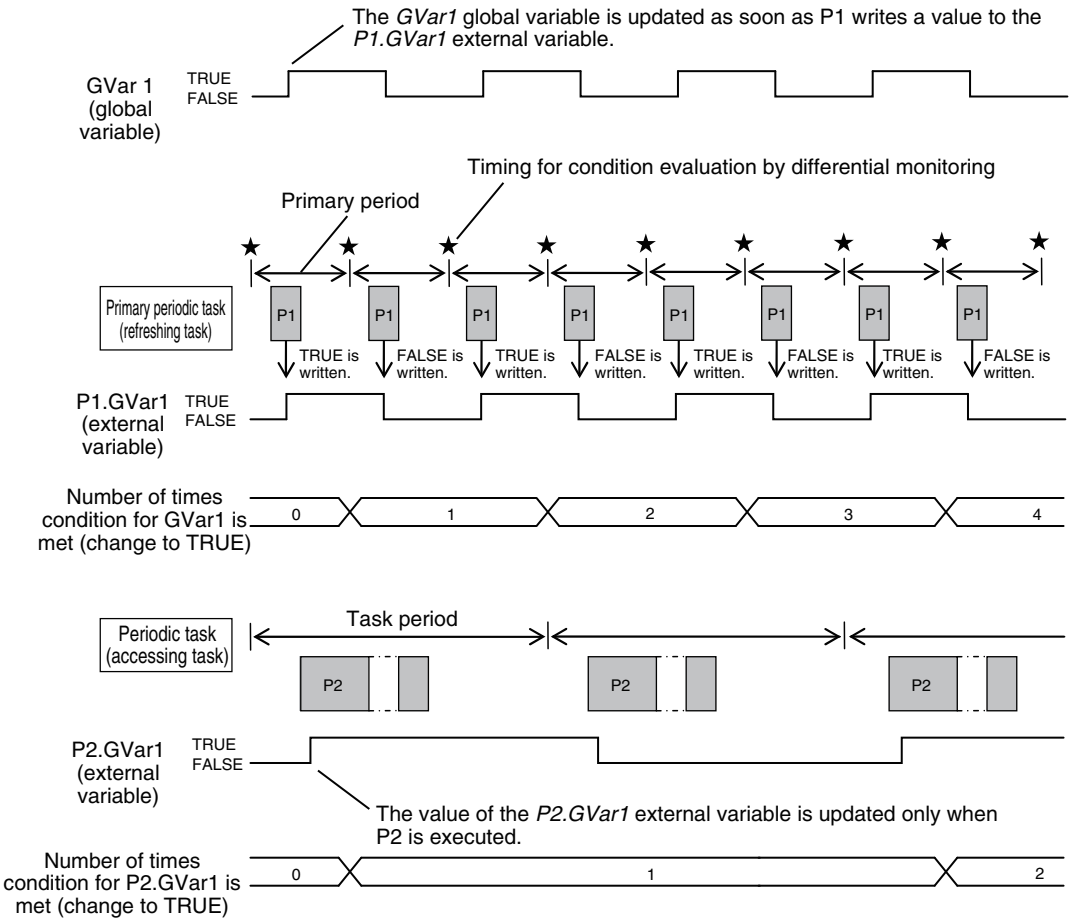
A global variable is updated as soon as the value is written. An external variable, however, is updated only when the CPU Unit executes the POU in which that external variable is declared.

The following figure shows this. In this example, the following two variables are monitored.

Variable name	Type of variable	POU that executes the read/write
GVar1	Global variable	The P1 program that is assigned to the primary periodic task
P2.GVar1	This is an external variable that is declared in the P2 program and points to GVar1.	The P2 program that is assigned to the periodic task

The *GVar1* global variable is read and written by the P1 program that is assigned to the primary periodic task. Therefore, it will be updated in the primary period as long as the program writes to it every period. The *P2.GVar1* external variable, however, is updated only when the CPU Unit executes the P2 program that is assigned to the periodic task. This means the external variable is updated only in the task period of the periodic task. Because the task period of the periodic task is longer than the primary periodic, the count for *P2.GVar1* is updated fewer times than the count for *GVar1*.





8-7-6 Automation Playback

Automation playback is a function that supports all three activities of system maintenance, recording, reproduction, and analysis, in an integrated manner.

You can use the automation playback function to replay the program operations on Sysmac Studio monitor using pre-recorded data of variables in the program.You can also analyze problems and identify causes by playing back videos captured in conjunction with program operations.

Refer to the *NX-series CPU Unit Automation Playback User's Manual (Cat. No. W639)* for details on the automation playback function.

Version Information

An NX502 CPU Unit with unit version 1.63 or later and Sysmac Studio version 1.55 or higher are required to use automation playback function.

8-8 Event Logs

This section describes the event logs.

8-8-1 Introduction

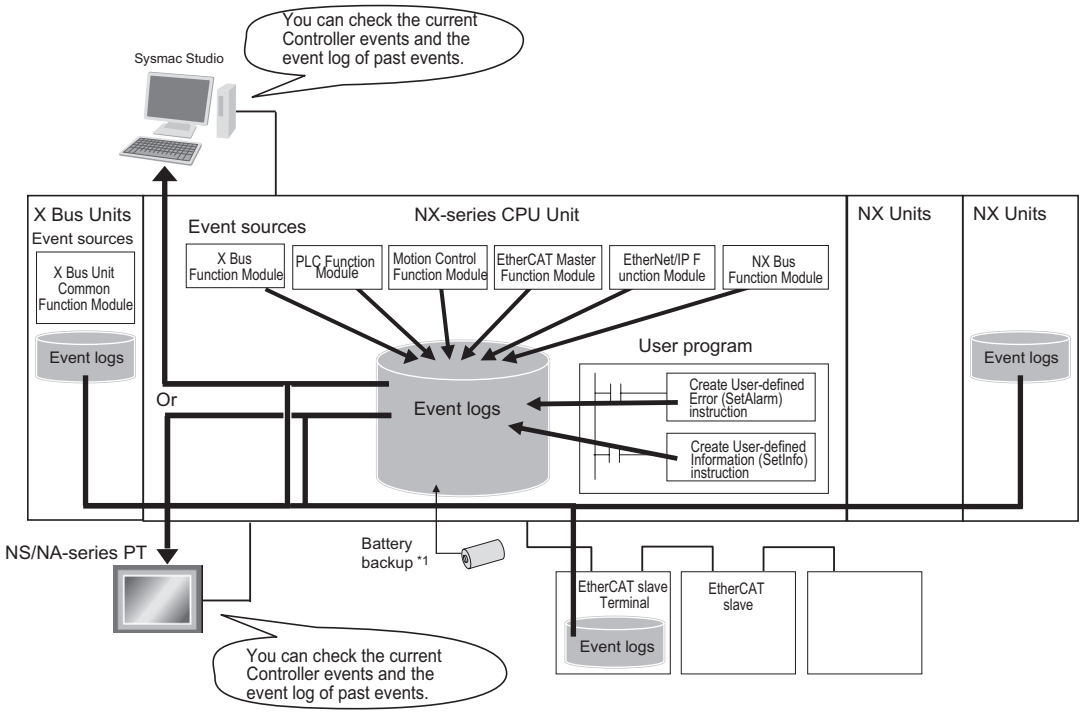
The event logs contain records of events,* such as errors, status changes, and user-defined events, that occurred in the NJ/NX-series Controller.

* Here, events are unscheduled events that occur on the Controller, such as errors. “Event” refers to an error or to information that does not indicate an error but for which the user must be notified by the Controller or for a user definition.

There are two types and four classifications of events.

- Controller events
  Controller errors
  Controller information
- User-defined events
  User-defined errors
  User-defined information

● NX-series CPU Units

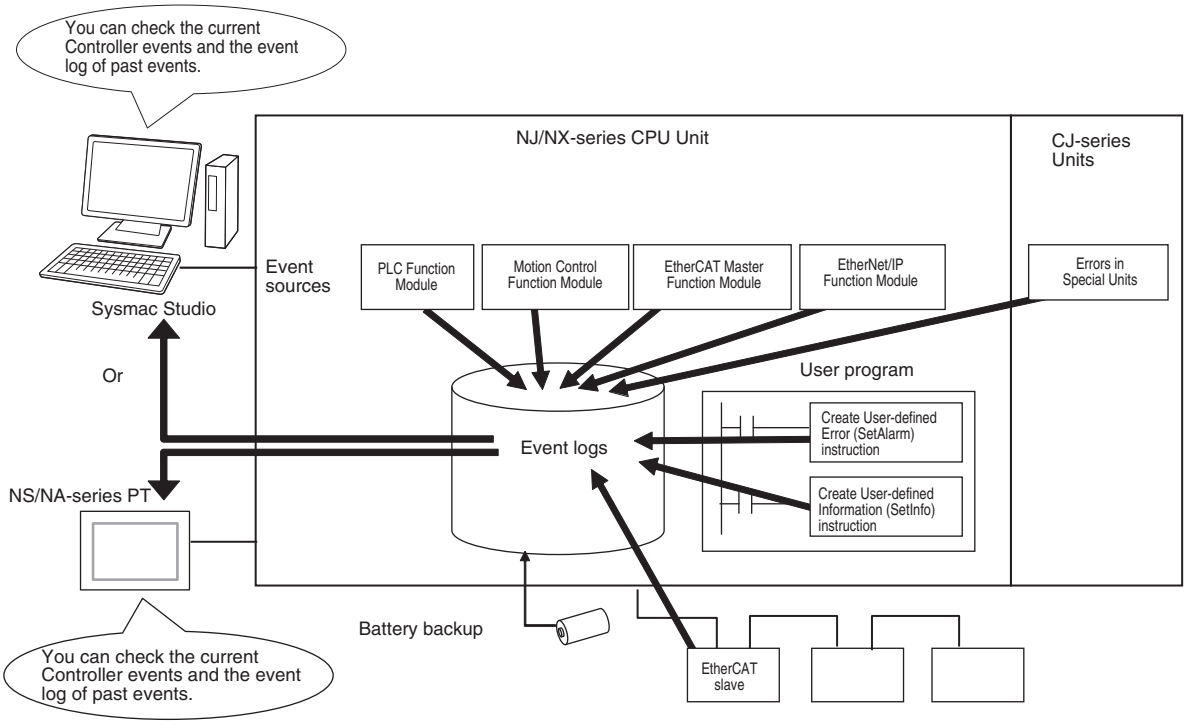


*1. The event logs are saved in battery-backup memory in the NX701 CPU Unit.
The event logs are saved in the non-volatile memory in the NX502 CPU Unit, NX102 CPU Unit and NX1P2 CPU Unit.

Precautions for Correct Use

- Only the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units have the NX Bus Function Module.
- The only CPU Units on which NX Units can be mounted are the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
- Only the NX502 CPU Units have the X Bus Function Module.
- The only CPU Units on which X Bus Units can be mounted are the NX502 CPU Units.
- Refer to the appendices of the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for the applicable range of the HMI Troubleshooter.

● NJ-series CPU Units



To use an NS-series PT to check events, connect the PT to the built-in EtherNet/IP port on the CPU Unit.

Precautions for Correct Use

- You can use CJ-series Units only with NJ-series CPU Units.
- Refer to the appendices of the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for the applicable range of the HMI Troubleshooter.

Features

Event logs have the following features.

- In addition to error records, various records are recorded for events such as the time the power supply is turned ON or OFF, and the time when operation is started.
- You can check these records based on the time. You can therefore use them to isolate the causes of errors when problems occur.

Types of Events

Events are classified as shown below.

● System-defined Events (Controller Events)

The Controller automatically detects these events. Controller events include events for the function modules in the CPU Unit, NX Units, NX-series Slave Terminals, EtherCAT slaves, and CJ-series Units.

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.

8-8-2 Detailed Information on Event Logs

Event Sources

This information identifies where an event occurred in the Controller.
The event sources are given below for Controller events and user-defined events.

● Sources of Controller Events

Controller events occur in the function modules in the CPU Unit.
For some function modules, there is more detailed information about the event source. This information is called the "detailed event source".
The following are Controller events.

Event Sources	Detailed event source
PLC Function Module	Instruction, power supply, built-in I/O, Option Board, I/O bus master or CJ-series Unit*1
NX Bus Function Module*2	Master or NX Unit
X Bus Function Module*3	Master
X Bus Unit Common Function Module *4	X Bus Unit
Motion Control Function Module	Common, Axis, Axes group
EtherCAT Master Function Module	Communications port, EtherCAT master, or EtherCAT slave
EtherNet/IP Function Module	Communications port/communications port 1/communications port 2, CIP/CIP1/CIP2, FTP, NTP, or SNMP

*1. The source details information does not show information from the error histories from within CJ-series CPU Special Units or EtherCAT slaves. Read the error histories from the appropriate Support Software.

*2. Only the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units have the NX Bus Function Module.

*3. Only the NX502 CPU Units have the X Bus Function Module.
*4. This function module is commonly installed in the X Bus Units. The X Bus Unit has a function module specific to each Unit as an event source. Refer to the user's manual for each X Bus Unit for details on the specific function modules for the X Bus Unit.

● Sources of User-defined Events

User-defined events occur in the PLC Function Module.

Category

This information displays the category of event log.
It is used to access error logs from the Sysmac Studio or an HMI.

Event type	Event log category	Description
Controller events	System log	The Controller automatically detects and records these events. CJ-series Unit errors are also included.
	Access log	This is a record of events that have affected Controller operation due to user actions.
User-defined events	User-defined event log	This is a log of events that are defined by the user.

Number of Records

Each event log can contain the following number of records.
If the number of events exceeds the number of records permitted, the CPU Unit overwrites the oldest events.

Event type	Event log category	Maximum number of records						
		NX701	NX502	NX102	NX1P2	NJ501	NJ301	NJ101
Controller events	System log	2,048	2,560*1	768*2	576*3	1,024	512	512
	Access log	1,024	1,152*4	576*5	528*6	1,024	512	512
User-defined events	User-defined event log	1,024	1,024	512	512	1,024	512	512

*1. Up to 2,048 system logs generated on CPU Units and up to 512 system logs generated on NX Units can be recorded.
*2. Up to 512 system logs generated on CPU Units and up to 256 system logs generated on NX Units can be recorded.
*3. Up to 512 system logs generated on CPU Units and up to 64 system logs generated on NX Units can be recorded.
*4. Up to 1,024 access logs generated on CPU Units and up to 128 access logs generated on NX Units can be recorded.
*5. Up to 512 access logs generated on CPU Units and up to 64 access logs generated on NX Units can be recorded.
*6. Up to 512 access logs generated on CPU Units and up to 16 access logs generated on NX Units can be recorded.

Retaining Events during Power Interruptions

When the power is interrupted, the NJ-series CPU Unit and NX701 CPU Unit use a Battery to retain the event logs, and the NX502, NX102, and NX1P2 CPU Units use non-volatile memory to retain the event logs.

Precautions for Correct Use

For the CPU Unit that uses a Battery to retain the event logs, the event logs are not retained when there is no Battery.
Periodically export event logs as required.

Event Codes

Event codes are assigned to Controller events by the system in advance according to the type of event. Event codes are assigned to user-defined events by the user. Controller event codes are 8-digit hexadecimal values.
You can use the Get Error Status instruction to read the error codes of current errors.
You can assign a decimal number from 1 to 60,000 as the event code for a user-defined event.

Event Levels

Each event has an "event level" that indicates its level.
The event level depends on the type of event. Levels are defined separately for Controller events and user-defined events.

● Controller Events

Controller events are classified into five levels according to the degree of the effect that the events have on control, as shown in the following table.

No.	Level		Classification
1	High	Controller errors	Major fault level
2	↕		Partial fault level
3			Minor fault level
4			Observation level
5	Low	Controller information	Information level

Errors with a higher level have a greater impact on the functions that the Controller provides, and are more difficult to recover from.
When an event in one of these levels occurs, the Sysmac Studio or an HMI will display the error.

● User-defined Events

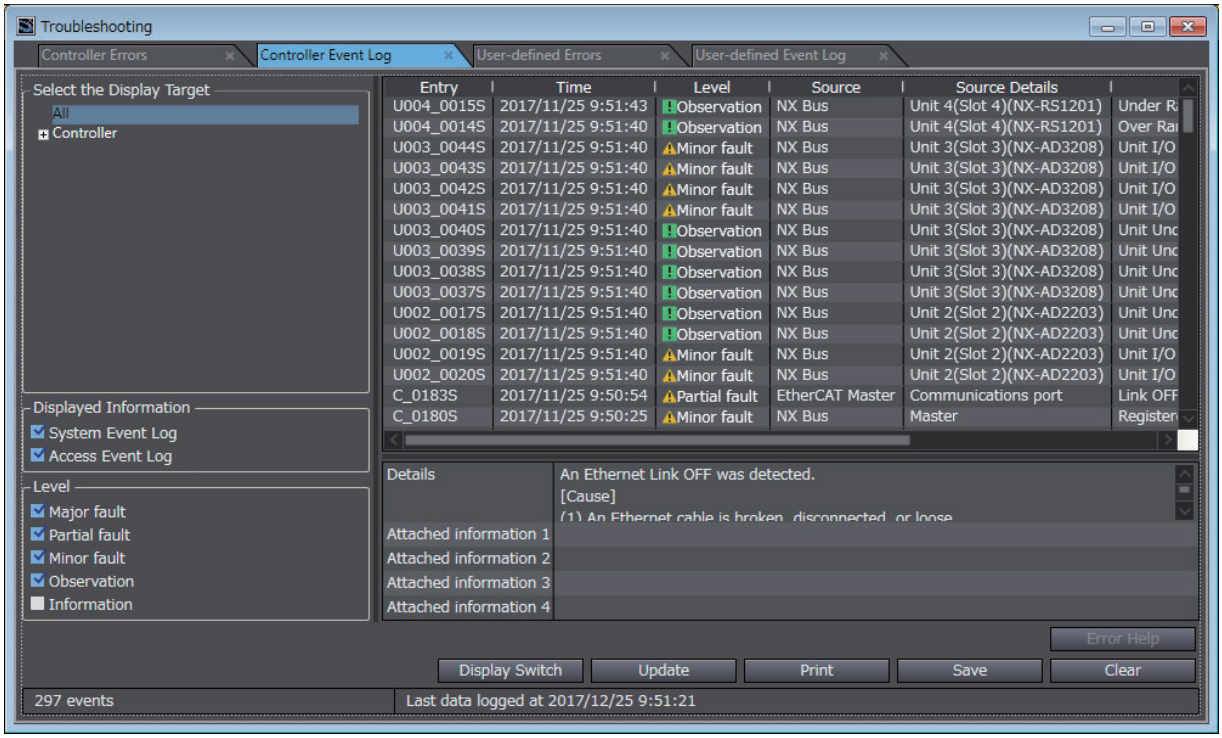
User-defined events are classified into the following levels. These levels are defined by the NJ/NX-series System.
The event levels are defined for user-defined events.

No.	Level	Type	Meaning
1	High	User fault Level1	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 Level2	
3		User fault Level3	
4		User fault Level4	
5		User fault Level5	
6		User fault Level6	
7		User fault Level7	
8		User fault Level8	
9	Low	User Information	These event levels indicate user-defined information in an application. The user executes the SetInfo (Create User-defined Information) instruction to create the event.

Displaying Event Logs

The Sysmac Studio or an HMI displays two event logs: the Controller event log and the user-defined event log. The Controller logs include both the access log and the system log.
The Sysmac Studio can also display the error logs that are recorded in the CJ-series Units and EtherCAT slaves.

The events in these logs are displayed in tables on the Sysmac Studio. Select an event from the table to display detailed information.



Additional Information

If an event occurs in the Controller that is not supported by the version of the Sysmac Studio or an HMI, the source is displayed as *Unknown* and the event name is displayed as *Unknown Event*. The event code and attached information are displayed correctly.

Clearing Event Logs

● Clearing Event Logs from the Sysmac Studio or an HMI

You can clear the event logs from the Sysmac Studio or from an HMI. You can clear the Controller event log and user-defined event log separately.

Precautions for Correct Use

- If you need to delete event log in the CPU Unit from the Sysmac Studio or an HMI, make sure you do not need any of the event information before you delete the event log. You may have overlooked some important information and observation level Controller events or user-defined events. Always check for these before you delete an event log.
- Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for restrictions on clearing an event log from the PT.

● Clearing Event Logs with the Clear All Memory Operation

When you perform the Clear All Memory operation for an NJ/NX-series CPU Unit from the Sysmac Studio, you can select whether to clear the event logs.

Exporting Event Logs

You can use the Sysmac Studio or an HMI to export the displayed event log to a CSV file.

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

Introduction

Controller errors and information are defined by the NJ/NX-series System.
These events occur when the NJ/NX-series System detects an error or information factor.

● Controller Errors

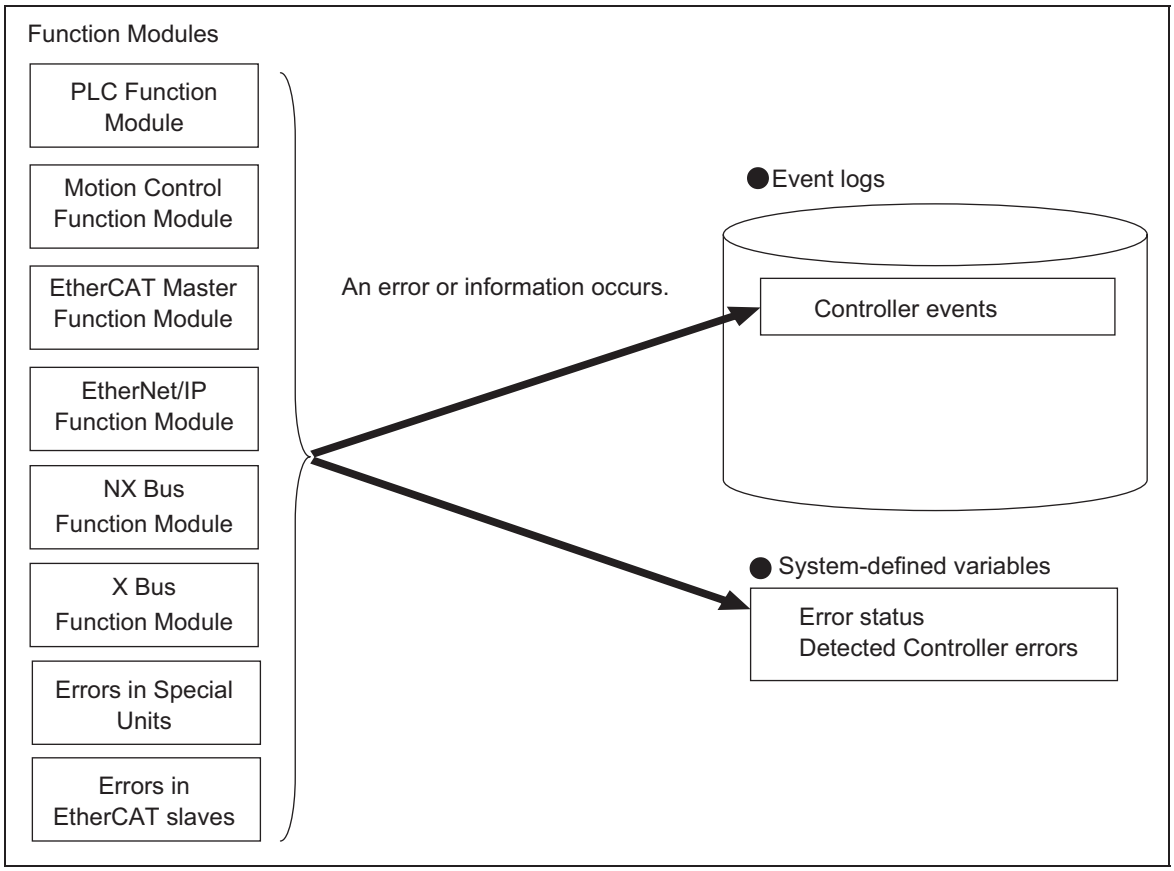
These are system-defined errors.
"Controller error" is a collective term for *major fault level*, *partial fault level*, *minor fault level*, and *observation level* Controller events.
Errors in the function modules of the CPU Unit, NX Units, NX-series Slave Terminals, X Bus Units, EtherCAT slaves, and CJ-series Units are detected. When one of these events occurs, a Controller error is recorded in the event log.
To check the status of a Controller error on the user program, you execute the Get Error Status instruction to access the status of the *Error Status variable*, which is a system-defined variable.

Note You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
Note You can use X Bus Units only on the NX502 CPU Units.
Note You can use CJ-series Units only on the NJ-series CPU Units.

Controller errors are not reset when the operating mode changes.
Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for details on Controller errors.

● Controller Information

Controller information is system-defined notification information. This information does not indicate errors. It represents information level Controller events.
Examples include events other than errors, such as turning the power ON and OFF, starting and stopping operation, connecting the Sysmac Studio online, and downloading user programs.



8-8-4 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.

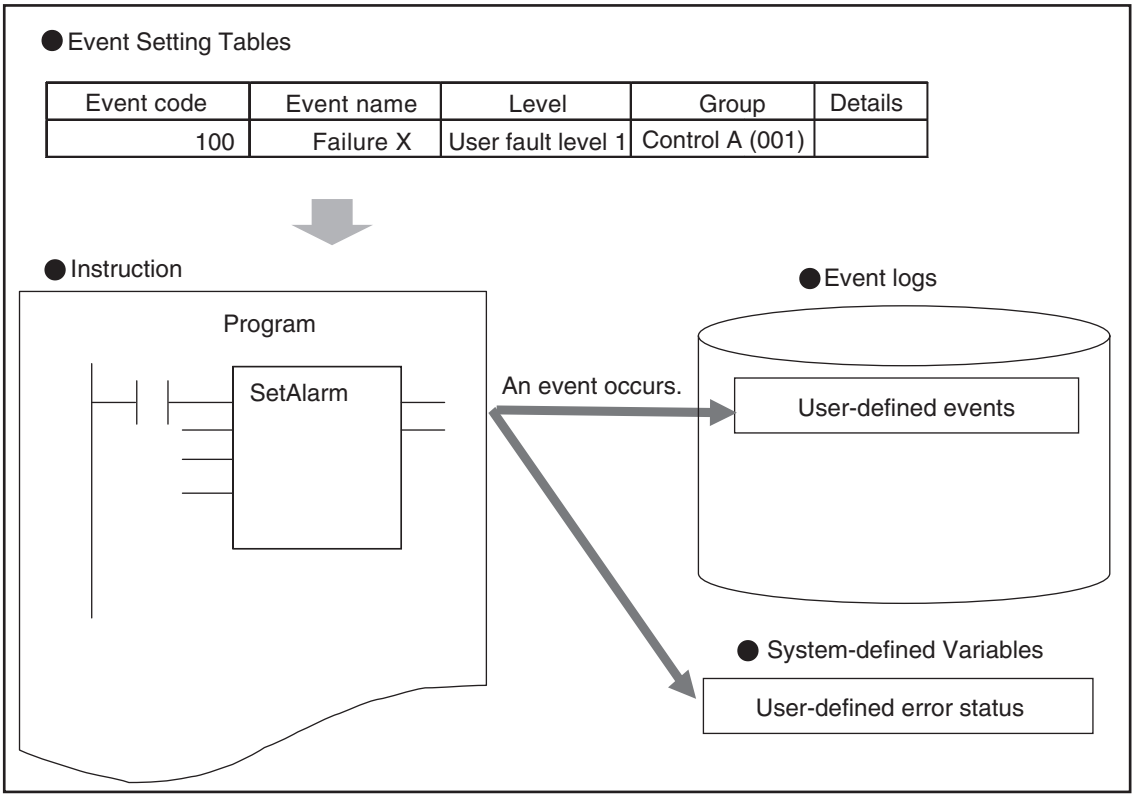
● User-defined Errors

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.

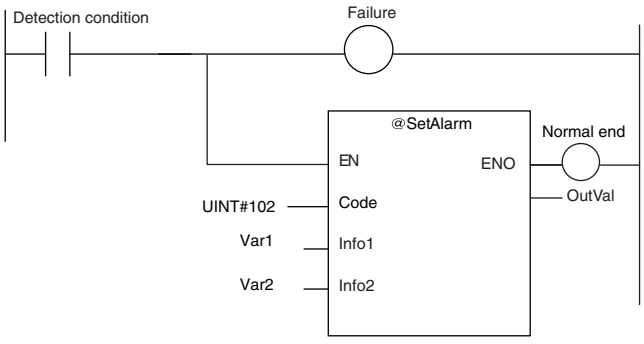
Event Setting Table

Event Setting Table

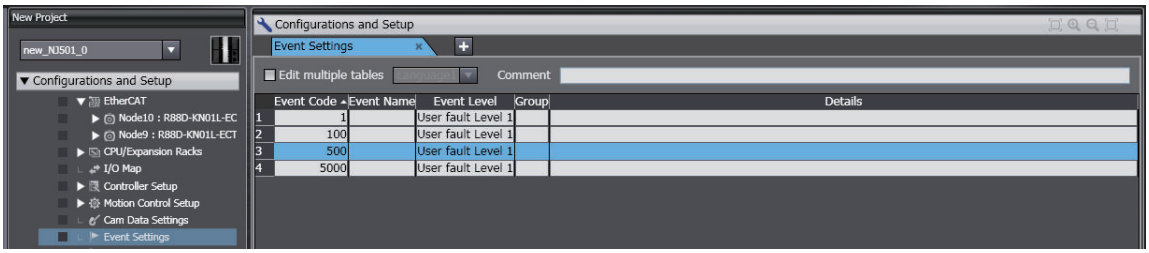
Event code	Event name	Level	Group	Details
10001	Failure X	User fault Level3	Control A (001)	
:	:	:	:	:

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 Displaying User Messages on an HMI When a Major Fault Level Controller Error Occurs page 8-91 that is given below in the additional information for more details.	128 characters max. There are no restrictions on the characters that can be used. Case sensitive. Reserved words: None
Comment	The comment is attached for each set of table entries.	

Additional Information

You can set up to nine different languages for the same event code for different regions and users. On the Sysmac Studio, you can import an Event Setting Table from a Microsoft Excel file via the clipboard.

Additional Information

Displaying User Messages on an HMI When a Major Fault Level Controller Error Occurs:
When a major fault level Controller error occurs, the user program execution stops.
The NJ/NX-series Controllers can display user messages on an HMI when a major fault level Controller error occurs. You can set the display messages under the list of user-defined events in the Event Setting Table on the Sysmac Studio.

● Event Levels and Event Codes

Event classification	Level	Event level category*	Range of corresponding event code	Description
User-defined errors	High	User fault Level1	1 to 5000	Select from eight levels.
		User fault Level2	5001 to 10000	
		User fault Level3	10001 to 15000	
		User fault Level4	15001 to 20000	
		User fault Level5	20001 to 25000	
		User fault Level6	25001 to 30000	
		User fault Level7	30001 to 35000	
	Low	User fault Level8	35001 to 40000	
User-defined Information	Lowest	User Information	40001 to 60000	The event type is user-defined information.

* User-defined error levels are separate from Controller error levels.

Precautions for Correct Use

If you update the Event Setting Table and transfer it to the CPU Unit, the event logs for user-defined events still contain old information. This can result in inconsistencies with the new Event Setting Table. Program operations with caution.

Related Instructions

There are instructions that you can use to create and check user-defined errors and to clear existing user-defined errors.

● Creating and Clearing User-defined Errors

Use the following instructions to create and reset user-defined errors and to create user-defined information.

Up to 32 events per level can occur simultaneously, for a total of 256 possible simultaneous events.

Instruction name	Instruction	Description
Create User-defined Error	SetAlarm	The SetAlarm instruction creates a user-defined error.
Reset User-defined Error	ResetAlarm	The ResetAlarm instruction resets a user-defined error.
Create User-defined Information	SetInfo	The SetInfo instruction records the specified user-defined information in the event log.

● Checking for User-defined Errors

You can use the Get User-defined Error Status (GetAlarm) instruction to obtain the status of the current user-defined errors and the highest priority event level and code of the current user-defined errors.

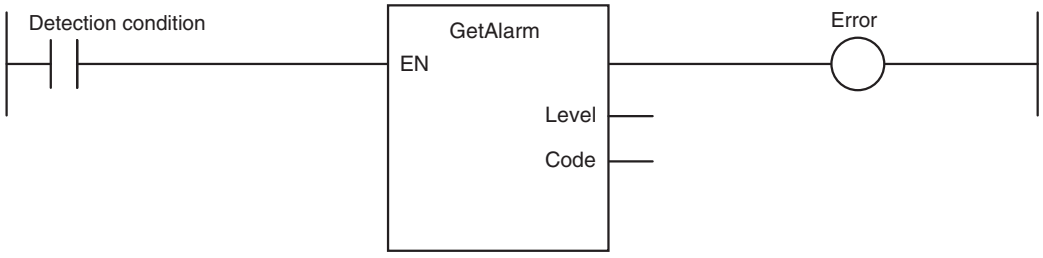
Example:

Event Setting Table

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

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

Actual Programming



Additional Information

You can use user-defined errors to add a message on possible corrections or other information when a Controller error occurs.
Use instructions such as the GetPLCError instruction to obtain information about the error status or event code when a Controller error occurs. You can then use the information to trigger a user-defined error.

Example 1
When a Low Battery Voltage error occurs, the event code (16#000B0000) is obtained and the following message is displayed.

Battery is dead. Apply power for at least five minutes before changing the Battery. Install a new Battery within five minutes of turning OFF the power supply.

Example 2
When a partial fault level Controller error occurs, the event error level is obtained (highest level status: 2) and the following message is displayed.

A device failed. Call the following number for support. Repair Contact Hours: 8:00 AM to 9:00 PM TEL: xxx-xxxx-xxxx

System-defined Variables Related to User-defined Errors

Variable name	Meaning	Function	Data type	R/W
_AlarmFlag	User-defined Error Status	The bit corresponding to the event level is TRUE while there is a user-defined error. Bits 00 to 07 correspond to user fault levels 1 to 8.	WORD	R

Records in Event Log

An event is recorded in the event log when you create user-defined information or a user-defined error, or when you use the ResetAlarm instruction to reset an error. When this happens, the time, event code, event level, and attached information 1 and 2 are recorded in the user-defined event log in the event logs.

Reset User-defined Errors

User-defined errors are cleared when the power supply to the NJ/NX-series Controller is turned ON. You can also clear errors with the Sysmac Studio, the Reset User-defined Error instruction (ResetAlarm) and an HMI.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details.

8-9 Changing Event Levels

Errors, status changes, and user-defined events that occur in the NJ/NX-series Controller are all called events. You can tell what type of event has occurred by viewing the display in Sysmac Studio, or by checking the indicators on the front panel of the CPU Unit.
There are two types of events: Controller events that are defined in the system and user-defined events. The Controller events are further classified into five event levels. Refer to *Event Levels* on page 8-85 for details on event levels.
You can change the event levels that are assigned to some of the Controller events.

Version Information

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

8-9-1 Applications of Changing Event Levels

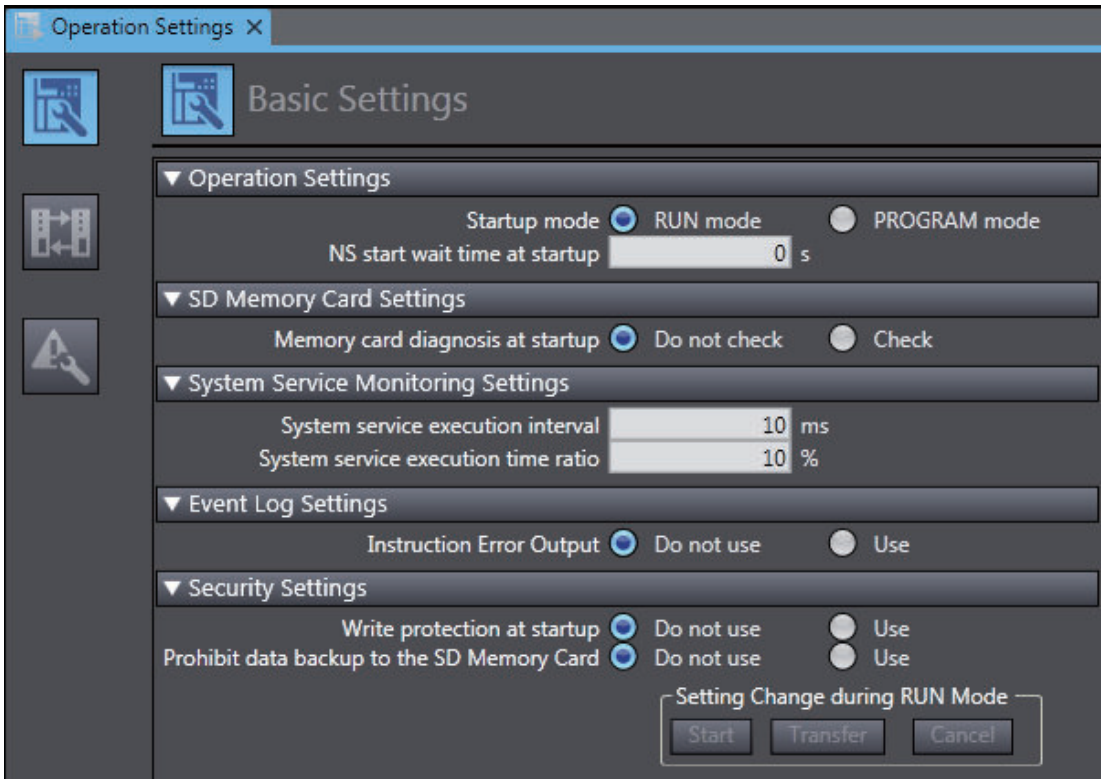
The lighting pattern for the indicators on the front panel of the CPU Unit is predefined according to the event level that is assigned to each Controller event. You can change the event level for some events to change how the Controller operates when that event occurs.
For example, the ERROR indicator flashes for minor fault level events and stays unlit for observation level events. You can change the lighting pattern of the ERROR indicator so that it goes out or flashes for a given event.
Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for details on how the Controller operates for different event levels.

8-9-2 Events for Which the Event Level Can Be Changed

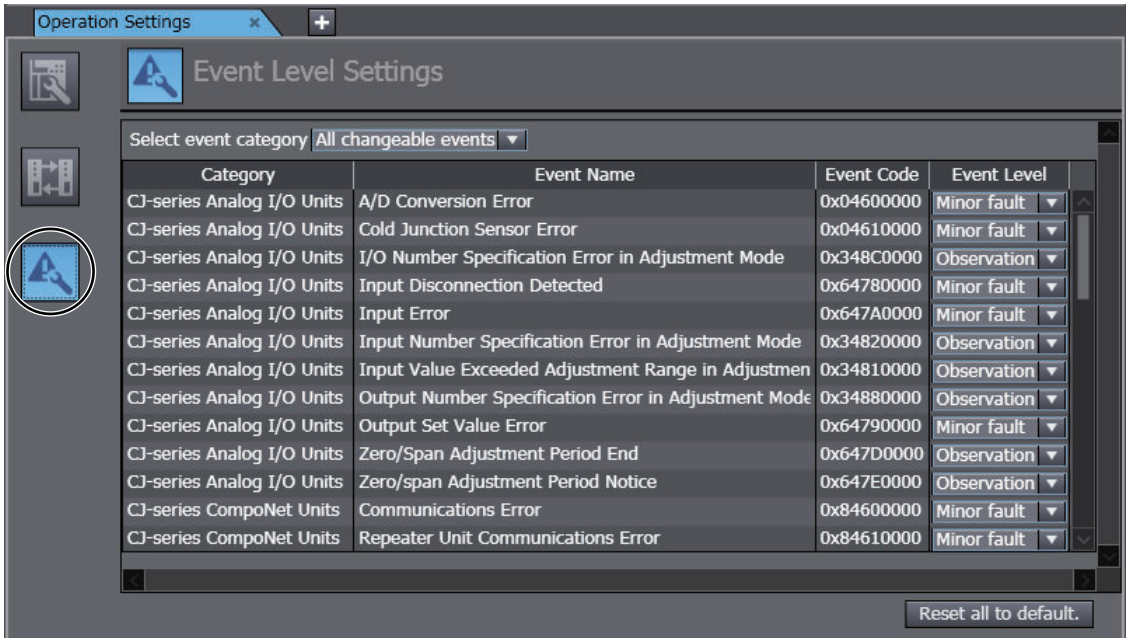
Whether an event level can be changed depends on the specific event.
Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for details on the types and levels of the Controller events, and whether the event levels can be changed.

8-9-3 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.

8-10 Displaying Firmware Update Log

You can use the Sysmac Studio to display the firmware update log.
Contact your OMRON representative for more information on the firmware update.
The Sysmac Studio displays the firmware update log only when the Controller is placed online.
You can display the firmware update log by selecting **Controller** - **Firmware Update Log** on the Sysmac Studio. The Firmware Update Log Dialog Box is displayed.
The Sysmac Studio version 1.53 or higher and an NJ-series CPU Unit, NX1P2 CPU Unit, or NX102 CPU Unit with unit version 1.60 or later, or an NX701 CPU Unit with unit version 1.32 or later are required to use the function to display the firmware update log.
The items in the firmware update log are shown in the table below.

Item	Display content	Description
Index	The record index of a firmware update execution log.	**1** represents the most recent execution log and the logs are output in ascending order. When the maximum number of log entries is reached, the oldest log will be overwritten thereafter. *1
Update Time (UTC)	yyyy/MM/dd hh:mm:ss	Displays the firmware update execution time in UTC.
Unit Version	Example: 1.60.00	Displays the unit version of the firmware.
Identifier	The identifier of the executed Firmware Updater.	Displays system information stored in the Firmware Updater. *2
Update Mode	Cleanup/ Development/ Manufacturing	Displays the mode of firmware update.
Result	Successfully completed/ Failed	Displays the execution result of the firmware update.

*1. The maximum number of log entries varies depending on the CPU Unit as follows.
- For NX701 CPU Units and NX502 CPU Units, 127 entries.
- For NX102 CPU Units and NX1P2 CPU Units, 63 entries.
- For NJ-series CPU Units, 127 entries. However, when the hardware revisions for the Units are in blank for the NJ101 and NJ301 CPU Units, 100 entries.

*2. If the unit version displayed in the firmware update log is one of the following unit versions, the identifier column is in blank.
- NX701 CPU Unit with unit version 1.29 or earlier
- NJ-series CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit with unit version 1.50 or earlier

9

Backup Functions

This section describes the backup functions for the settings in an NJ/NX-series Controller. Various types of backup functions are available. The target data and storage location are different for each type. It starts with the overall description of the backup functions and then introduces each backup function.

9-1 The Backup Functions 9-3
- 9-1-1 Applications of Backup Functions 9-4
- 9-1-2 Examples of Operating Procedures for the Backup Functions 9-4
- 9-1-3 Data that Is Backed Up 9-6
- 9-1-4 Types of Backup Functions 9-8
- 9-1-5 Relation between the Different Types of Backup Functions and Data Groups 9-11
- 9-1-6 Applicable Range of the Backup Functions 9-12

9-2 SD Memory Card Backups 9-15
- 9-2-1 Backup (Controller to SD Memory Card) 9-16
- 9-2-2 Restore (SD Memory Card to Controller) 9-21
- 9-2-3 Verify (between Controller and SD Memory Card) 9-29

9-3 Disabling Backups to SD Memory Cards 9-35

9-4 Automatic Transfers from SD Memory Cards 9-37

9-5 Program Transfer from SD Memory Card 9-39

9-6 Sysmac Studio Controller Backups 9-46
- 9-6-1 Backup (Controller to Computer) 9-47
- 9-6-2 Restore (Computer to Controller) 9-48
- 9-6-3 Verify (between Controller and Computer) 9-49

9-7 Importing and Exporting Sysmac Studio Backup File Data 9-51

9-8 Sysmac Studio Variable and Memory Backup Functions 9-53
- 9-8-1 Applicable Data for Sysmac Studio Variable and Memory Backup Functions 9-53
- 9-8-2 Using Sysmac Studio Variable and Memory Backup Functions 9-53
- 9-8-3 Compatibility between CPU Unit Models 9-54

9-9 Backup Functions When EtherCAT Slaves Are Connected 9-57
- 9-9-1 Backed Up EtherCAT Slave Data 9-57
- 9-9-2 Backup Support Depending on the Controller Status 9-57
- 9-9-3 Conditions for Restoring EtherCAT Slave Data 9-58
- 9-9-4 EtherCAT Slaves for Which You Can Back Up Data 9-59

9-10 Backup Functions When EtherCAT Slave Terminals Are Connected 9-62

9-10-1 Backing Up Data in an EtherCAT Slave Terminal 9-62
9-10-2 Backup Support Depending on the EtherCAT Slave Terminal Status 9-63
9-10-3 Conditions for Restoring EtherCAT Slave Terminal Data 9-63

9-11 Backup Functions When NX Units Are Connected 9-65
9-11-1 Backing Up Data in NX Units on the CPU Unit 9-65
9-11-2 Backup Support Depending on the Controller Status 9-65
9-11-3 Conditions for Restoring NX Unit Data on the CPU Unit 9-66

9-12 Backup Functions When X Bus Units Are Connected 9-67
9-12-1 Backing Up Data in X Bus Units on the CPU Unit 9-67
9-12-2 Backup Support Depending on the Controller Status 9-67
9-12-3 Conditions for Restoring X Bus Unit Data on the CPU Unit 9-67

9-13 Backup Functions When CJ-series Units Are Connected 9-68
9-13-1 Backed Up CJ-series Unit Data 9-68
9-13-2 Backup Support Depending on the Controller Status 9-68
9-13-3 Conditions for Restoring CJ-series Unit Data 9-68

9-14 Backup-related Files 9-70
9-14-1 Types of Backup-related Files 9-70
9-14-2 Specifications of a Backup File 9-71
9-14-3 Specifications of a Restore Command File 9-72
9-14-4 Specifications of an Automatic Transfer Command File 9-74
9-14-5 Specifications of a Controller Verification Results File 9-76
9-14-6 Specifications of an EtherCAT Verification Results File 9-77
9-14-7 Specifications of an EtherCAT Slave Terminal Verification Results File 9-78
9-14-8 Specifications of an NX Unit Verification Results File 9-79
9-14-9 Specifications of an X Bus Unit Verification Results File 9-80
9-14-10 Specifications of a CJ-series Unit Verification Results File 9-82

9-15 Compatibility between Backup-related Files 9-84
9-15-1 Compatibility between Backup Functions 9-84
9-15-2 Compatibility between CPU Unit Models 9-85
9-15-3 Compatibility between Unit Versions of CPU Units 9-86

9-16 Functions that cannot be Executed during Backup Functions 9-88

9-1 The Backup Functions

The following three functions are supported for data backup for an NJ/NX-series Controller.

Function	Description
Backing up data	You can back up all of the data in the Controller to an SD Memory Card or to a computer. A file to save is called a backup file.
Restoring data	You can transfer the contents of a backup file on the SD Memory Card or computer to the Controller. All of the data in the Controller are replaced with the data when it is backed up.
Verifying data	You can compare the contents of a backup file on the SD Memory Card or computer with the data in the Controller to see if they are the same.

The following items are described for the backup functions.

Item	Description
Applications of backup functions	Effective usage of the backup functions is described.
Examples of operating procedures for the backup functions	The backup functions are executed with simple procedures. Examples are provided.
Data that is backed up	The data that can be saved with the backup functions from the connected Units and slaves is described.
Types of backup functions	There are different types of backup functions that differ in where the data is saved. The types of backup functions and the difference between them are described.
Relation between the different types of backup functions and data groups	Different types of backup functions handle different data groups. The relation between the different types of backup functions and data groups is described.
Applicable range of the backup functions	The connected Units and slaves for which you can save data with the backup functions are described.

Precautions for Safe Use

- The performance may be different if the hardware revisions are different. Before you transfer the user program, data, and parameter settings to the CPU Units with the different hardware revisions, check them for proper execution and then use them for actual operation.
- For NX-series CPU Unit, we recommend that you back up the present values of variables while the retained variables are not refreshed.
  If you back up the following variables while the values of retained variables are refreshed, the data may not be saved correctly.
  - Structure members whose data size is 16 bits or more.
  - Array elements whose data size is 16 bits or more.

Precautions for Correct Use

- The client certificate and client private key that are related to the secure socket communications are information that are attached to the individual CPU Unit, therefore, the information are out of the target of backup and restoration.
  Transfer the client certificate and client private key to the CPU Unit when you replace the hardware of the CPU Unit.
  Similarly, the secure socket setting is also out of the target of backup and restoration. Use the Secure Socket Configuration commands to make the setting to the CPU Unit.
- Network security issues such as the server data be illegally obtained or tampered, or communications to the server be disabled may occur due to theft, information leaks and tampering of the client certificate, private key and secure socket setting by third parties.
  Take necessary measures for the management of the client certificate, private key and secure socket setting and for the prevention of theft, information leaks and tampering of those.
  Especially, use an encrypted safe communications path, etc. when obtaining the private key to avoid information leaks.
  Furthermore, store the private key in a location where possibilities of leaks are eliminated as much as possible.

Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for information on the secure socket communications.

9-1-1 Applications of Backup Functions

You can use the backup functions in the following instances.

Item	Application
Program and setting changes	When you change the user program and settings for equipment that is currently in operation.
Hardware replacements	When you replace the hardware for the CPU Unit, other Units, or slaves.
Troubleshooting equipment failures	When you want to save data in the Controller to analyze the cause of an error that occurs in the equipment.
Equipment backup and recovery	When an error occurs in the equipment, and when you want to restore the equipment with data from 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 Examples of Operating Procedures for the Backup Functions

You can use the backup functions to easily back up, restore, and verify Controller data.
This section describes the procedure for performing a backup, restore or compare operation of the SD Memory Card using CPU Unit front-panel switch.

Precautions for Correct Use

For the NX701 CPU Units, eight pins, pins 1 to 8, are provided on the DIP switch. Before you use the backup functions, set all of pins 5 to 8 to OFF.

Backup Procedure

● Preparations

1 Insert the SD Memory Card into the CPU Unit.

2 Set pins 1 to 4 on the DIP switch on the CPU Unit as follows: 1: OFF, 2: OFF, 3: ON, and 4: OFF.

● Executing the Backup

1 Press the SD Memory Card power supply switch for 3 seconds.
The backup is started. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. When the backup operation is completed, the SD PWR indicator will stop flashing and remain lit.

● Ending the Backup Procedure

1 Set all of pins 1 to 4 on the DIP switch on the CPU Unit to OFF.

2 Press the SD Memory Card power supply switch to turn OFF the SD PWR indicator.

3 Remove the SD Memory Card.

Restoration Procedure

● Preparations

1 Turn OFF the power supply to the NJ/NX-series Controller and to the EtherCAT slaves.

2 Insert the SD Memory Card that contains the backup file into the CPU Unit.

3 Set pins 1 to 4 on the DIP switch on the CPU Unit as follows: 1: OFF, 2: OFF, 3: ON, and 4: ON.

● Restoring Data

1 Turn ON the power supply to the NJ/NX-series Controller and to the EtherCAT slaves.
The restoration operation is started. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. When the restoration operation is completed, the SD PWR indicator will stop flashing and remain lit.

● Ending the Restoration Procedure

1 Press the SD Memory Card power supply switch to turn OFF the SD PWR indicator.

2 Turn OFF the power supply to the NJ/NX-series Controller and to the EtherCAT slaves.

● Starting Normal Operation

1 Remove the SD Memory Card.

2 Set all of pins 1 to 4 on the DIP switch on the CPU Unit to OFF.

3 Turn ON the power supply to the NJ-series Controller and to the EtherCAT slaves.

Precautions for Correct Use

Restoring Data when EtherCAT Slaves Are Connected

- Always cycle the power supply to the NJ/NX-series Controller and the EtherCAT slaves after you restore data when EtherCAT slaves are connected. If you start operation without cycling the power supply, the Controller may perform unexpected operation.
- To verify the data after you restore data with EtherCAT slaves connected, first turn OFF the power supply to the NJ/NX-series Controller and EtherCAT slaves, and then start in Safe Mode before you perform the verification procedure. If you cycle the power supply normally, the Controller will start operation before you can perform the verification procedure. That means that operation could be started with data that is not correct. For information on Safe Mode, refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)*.

Verification Procedure

● Preparations

1 Insert the SD Memory Card that contains the backup file into the CPU Unit.

2 Set all of pins 1 to 4 on the DIP switch on the CPU Unit to OFF.

● Verifying the Data

1 Press the SD Memory Card power supply switch for 3 seconds.
Data comparison is started. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds.
If the verification operation is completed and the data is the same, the SD PWR indicator will stop flashing and remain lit.
If the verification operation is completed and differences were found in the data, the SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds.

● Ending the Verification Procedure

1 Press the SD Memory Card power supply switch to turn OFF the SD PWR indicator.

2 Remove the SD Memory Card.

9-1-3 Data that Is Backed Up

The following data is backed up.
This section describes the backup functions based on the following *data groups* for the backup data.

Data group	Data items
User program and settings	EtherCAT configuration (EtherCAT slave configuration and EtherCAT master settings) Unit Configuration and Unit Setup[*1*2] I/O Map Controller Setup (Operation Settings, Built-in EtherNet/IP Port Settings, Built-in I/O Settings, and Option Board Settings)[*3] Motion Control Setup Cam Data Settings Event Settings Task Settings Data Trace Settings Tag Data Link Tables Controller name Operation authority verification User authentication[*4] User program execution ID in user program Built-in clock (time zone setting) POU Data (data types and global variables) Memory Settings for CJ-series Units[*5]
IP address of the built-in EtherNet/IP port[*6]	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[*7]
Present values of memory used for CJ-series Units	Values of the Holding, DM, and EM Areas in the memory for CJ-series Units[*5]
Units and slaves settings	Backup parameters for EtherCAT slaves[*8] Parameters in the CJ-series Units[*5*9] NX Unit Settings
Absolute encoder home offsets	The set value to restore the actual position of a Servo Drive with an absolute encoder
X Bus Unit Settings[*2]	Settings in X Bus Unit
IP address of an EtherNet/IP port on an X Bus Unit[*2]	Of the TCP/IP Settings in the NX-series EtherNet/IP Unit Settings, setting type, IP address, subnet mask, and default gateway

*1. For the NX Units on the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, data of configuration information, Unit operation settings and Unit application data is backed up.
*2. You can use X Bus Units only on the NX502 CPU Units.
*3. Data of Built-in I/O Settings and Option Board Settings is backed up only for the NX1P2 CPU Units.
*4. User authentication is backed up for the following CPU Units of the stated unit versions.
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

*5. You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units. You can use parameters in the CJ-series Units only with NJ-series CPU Units.
*6. With a combination of the CPU Unit 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 for other combinations.
*7. Of the system-defined variables with a Retain attribute, some variables are not applicable for the data backup function. Refer to *A-8 Specifications for Individual System-defined Variables* on page A-163 for details on the specifications for individual system-defined variables.

*8. 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 *9-9-4 EtherCAT Slaves for Which You Can Back Up Data* on page 9-59.

*9. Refer to the *CJ-series CJ2 CPU Unit Hardware User's Manual (Cat. No. W472)* for details on the data that is backed up.

Precautions for Safe Use

Precautions on the Absolute Encoder Home Offset

The *absolute encoder home offsets* are retained in the CPU Unit as absolute encoder information. If any of the following conditions is met, clear the absolute encoder home offsets from the list of data items to restore, and then restore the data. Then, define the absolute encoder home again. If you do not define home, unintended operation of the controlled system may occur.

- The Servomotor or Servo Drive was changed since the data was backed up.
- The absolute encoder was set up after the data was backed up.
- The absolute data for the absolute encoder was lost.

9-1-4 Types of Backup Functions

There are backup functions for the NJ/NX-series Controllers that save data to SD Memory Cards and others that save data to a computer. Also, there are three methods used to execute the backup functions: the CPU Unit front-panel DIP switches, system-defined variables, and the Sysmac Studio.

Functions That Save Data to SD Memory Cards

The SD Memory Card backup functions are used to back up, restore, and compare data on SD Memory Cards. Related functions include disabling backups to SD Memory Cards, automatic transfers from SD Memory Cards, and program transfer from SD Memory Card.

Function name		Description	Operation method			Reference
			CPU Unit front-panel DIP switch	System-defined variables	Sysmac Studio	
SD Memory Card Backups	Backing up data	The Controller data is saved in a backup file on the SD Memory Card.	○	○	○	*9-2-1 Backup (Controller to SD Memory Card)* on page 9-16
	Restoring data	The data in a backup file on the SD Memory Card is transferred to the Controller.	○	○		*9-2-2 Restore (SD Memory Card to Controller)* on page 9-21
	Verifying data	The Controller data and the data in a backup file on the SD Memory Card are compared.	○	○	○	*9-2-3 Verify (between Controller and SD Memory Card)* on page 9-29
Disabling backups to SD Memory Cards		You can disable backing up data to SD Memory Cards.			○	*9-3 Disabling Backups to SD Memory Cards* on page 9-35

Function name	Description	Operation method			Reference
		CPU Unit front-panel DIP switch	System-defined variables	Sysmac Studio	
Automatic transfers from SD Memory Card	When the power supply is turned ON, the data in a backup file on the SD Memory Card is automatically transferred to the Controller. After the data transfer, the operating mode of the CPU Unit will change to the mode that is specified in **Startup Mode** setting.	○			*9-4 Automatic Transfers from SD Memory Cards* on page 9-37
Program transfer from SD Memory Card	With a system-defined variable, you can transfer a program that is stored in the SD Memory Card to the Controller. After the transfer, the operating mode of the CPU Unit will change to the mode that is specified in **Startup Mode** setting.		○		*9-5 Program Transfer from SD Memory Card* on page 9-39
Safety unit restore*1	The data in a Safety Unit restore file on the SD Memory Card is transferred to the NX-SL5□□□ that is mounted to the NX502 CPU Unit or NX102 CPU Unit.	*2			*NX-series Safety Control Unit User's Manual (Cat. No. Z930-E1-12 or later)*

*1. An NX502 CPU Unit with unit version 1.60 or later and Sysmac Studio version 1.54 or higher are required to use the function. Also, an NX102 CPU Unit with unit version 1.31 or later and Sysmac Studio version 1.24 or higher are required to use this function.
The safety unit restore function restores only the data in the NX-SL5□□□. If you restore in the entire system, you must execute the safety unit restore with a combination of the SD Memory Card backups.

*2. Set the front-panel DIP switch on the NX-SL5□□□ Safety CPU Unit. Refer to the *NX-series Safety Control Unit User's Manual (Cat. No. Z930-E1-12 or later)* for details.

Functions That Save Data to the Computer

The Sysmac Studio Controller backup functions are used to back up, restore, and compare data on the computer.
Importing and exporting Sysmac Studio backup file data are used to save and read different types of data between the Sysmac Studio projects and backup files on the computer without using a Controller.
The Sysmac Studio variable and memory backup functions are used to back up battery-backup present values to the computer and restore them from the computer.

Function name		Description	Operating method			Reference
			CPU Unit front-panel DIP switch	System-defined variables	Sysmac Studio	
Sysmac Studio Controller backups	Backing up data	The Controller data is saved in a backup file on the computer.			○	*9-6-1 Backup (Controller to Computer)* on page 9-47
	Restoring data	The data in a backup file on the computer is transferred to the Controller.			○	*9-6-2 Restore (Computer to Controller)* on page 9-48
	Verifying data	The Controller data and the data in a backup file on the computer are compared.			○	*9-6-3 Verify (between Controller and Computer)* on page 9-49
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.			○	*9-7 Importing and Exporting Sysmac Studio Backup File Data* on page 9-51
	Importing data	The data in the backup file is imported into the Sysmac Studio project without using a Controller.			○	
Sysmac Studio variable and memory backup functions	Backing up data	You can back up the present values of data that is backed up by a battery to an XML file on the computer.			○	*9-8 Sysmac Studio Variable and Memory Backup Functions* on page 9-53
	Restoring data	You can restore the present values of data that is backed up by a battery from the computer to the CPU Unit.			○	

Version Information

- A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to use the following backup functions: SD Memory Card backups, automatic transfers from SD Memory Cards, Sysmac Studio Controller backups, and importing and exporting Sysmac Studio backup file data.
- A CPU Unit with unit version 1.11 or later and Sysmac Studio version 1.15 or higher are required to transfer programs from the SD Memory Card.
- A CPU Unit with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required to use the restore of SD Memory Card backups by the system-defined variable.

Additional Information

The backup functions are executed as a system service. This means that if you perform a backup or verification operation in RUN mode with an NJ-series CPU Unit, it may take time for the operation to be completed. If you perform a backup or verification operation in RUN mode, make sure that the sufficient execution time is allocated for the system service. You can reduce the processing time by performing the system service in PROGRAM mode.

9-1-5 Relation between the Different Types of Backup Functions and Data Groups

Different types of backup functions handle different data groups. The relation between the different types of backup functions and data groups is given in the following table.

(○: Applicable, x: Not applicable)

Type of backup function		Data group					
		User program and settings	IP address of built-in EtherNet/IP port*2	Present values of variables	Present values of memory used for CJ-series Units*1	Units and slaves settings	Absolute encoder home offsets
SD Memory Card backups	Backing up data	○	○	○*3	○*4	○	○
	Restoring data	○	○	○*3	○*4	○	○
	Verifying data	○*5	○	×	×	○	×
Automatic transfers from SD Memory Cards*6		○	○	○*3	○*4	×	×
Program transfer from SD Memory Card*7		○	○	○*3	○*4	×	×
Sysmac Studio Controller backups	Backing up data	○	○	○*3	○*4	○*8	○
	Restoring data	○	○	○*3	○*4	○*8	○
	Verifying data	○*5	○	×	×	○*8	×
Importing and exporting Sysmac Studio backup file data	Exporting backup file data	○*9	○	×	×	×	×
	Importing backup file data	○*10*11	○	×	×	○	×
Sysmac Studio variable and memory backup functions	Backing up and restoring data	×	×	○*3	○*4	×	○

*1. You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

*2. With a combination of the CPU Unit 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.

*3. The backup data is processed only for the present values of variables that are specified for retention with the Retain attribute.

*4. The backup data is processed only for the present values of addresses that are specified for retention with the Retain attribute in the memory for CJ-series Units.

*5. Of the user program and setting data groups, the Data Trace Settings are not compared.

*6. For all of the data groups except for the user program and setting group, only the items that are specified to be transferred in the automatic transfer command file are transferred.
*7. For all of the data groups, only the data that is specified as the transfer target by the system-defined variable is transferred.
*8. If the CJ-series Units are specified for backup, the parameters in the CJ-series Units are backed up. If the EtherCAT slaves are specified for the backup, parameters for the EtherCAT slaves are backed up.
*9. The following data is not processed.
- The built-in EtherNet/IP port name in the Controller name
- The built-in EtherNet/IP tag data link settings in the Controller Setup
- Words allocated to CPU Bus Units in the Unit Configuration and Unit Settings
- Operation authority verification
- User authentication
- Data Trace Settings

*10. The following data is not processed.
- The built-in EtherNet/IP port name in the Controller name
- Words allocated to CPU Bus Units in the Unit Configuration and Unit Settings
- Operation authority verification
- User authentication
- Data Trace Settings

*11. The following data is not processed.
When Sysmac Studio lower than version 1.16 is used
- The built-in EtherNet/IP tag data link settings in the Controller Setup

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 *9-8 Sysmac Studio Variable and Memory Backup Functions* on page 9-53 for details on the Sysmac Studio variable and memory backup functions.

9-1-6 Applicable Range of the Backup Functions

Different types of backup functions handle data for different Units or slaves. The applicable Units and slaves for each backup function are given in the following table.

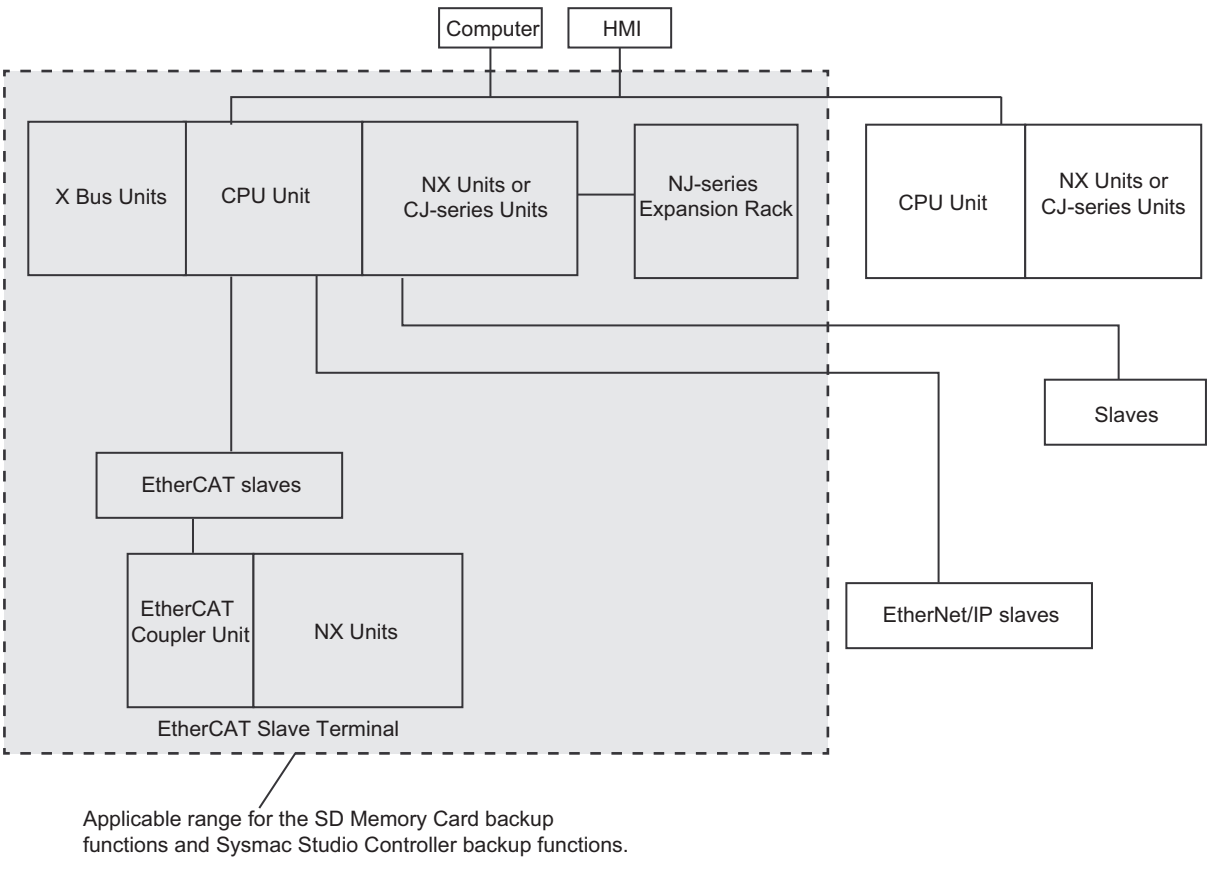
(○: Applicable, ×: Not applicable)

Type of backup function	Units/slaves							
	NJ/NX-series CPU Unit	EtherCAT slaves*1	NX Units on the CPU Unit*2*3	X Bus Units*4	CJ-series Units*5		Ether-Net/IP slaves	Computer and HMIs
					Units and Master Units	Slaves		
SD Memory Card backups	○	○*3	○	○	○	×	×	×
Automatic transfers from SD Memory Card	○	×	×	○	×	×	×	×
Program transfer from SD Memory Card	○	×	×	○	×	×	×	×
Sysmac Studio Controller backups	○	○*3	○	○	○	×	×	×

Type of backup function	Units/slaves							
	NJ/NX-series CPU Unit	EtherCAT slaves*1	NX Units on the CPU Unit*2*3	X Bus Units*4	CJ-series Units*5		EtherNet/IP slaves	Computer and HMIs
					Units and Master Units	Slaves		
Importing and exporting backup file data on Sysmac Studio	○	○*6	○	○	×	×	×	×
Variable and memory backup functions of Sysmac Studio	○	×	×	×	×	×	×	×

*1. EtherCAT Slave Terminals are included. If EtherCAT Slave Terminals are set for backup, the backup function applies to both the EtherCAT Coupler Unit and the NX Units.
*2. You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*3. This does not apply to Safety Control Units. Refer to the *NX-series Safety Control Unit User's Manual (Cat. No. Z930-E1-12 or later)* for information on importing and exporting settings and safety unit restore settings for a Safety Control Unit.
*4. You can use X Bus Units only on the NX502 CPU Units.
*5. You can use CJ-series Units only on the NJ-series CPU Units.
*6. Only importing data is possible. Exporting is not possible.

The Units and slaves that are shown in the following figure are covered by the SD Memory Card backup functions and Sysmac Studio Controller backup functions.



Note You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

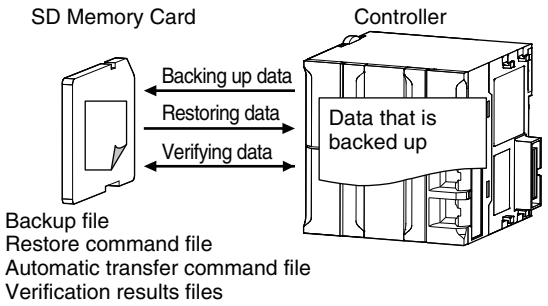
Note You can use X Bus Units only on the NX502 CPU Units.
Note You can use CJ-series Units and NJ-series Expansion Racks only with the NJ-series CPU Unit.

Version Information

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

9-2 SD Memory Card Backups

You can use SD Memory Cards to back up, restore, and verify Controller data.



When you back up data, the *Backup file*, *Restore command file*, and *Automatic transfer command file* are created in the specified directory on the SD Memory Card. When you verify data, the *Verification results files* are created in the specified directory.
All of these files are collectively referred to as *Backup-related files*.
The functions of the backup-related files are given in the following table.

File	Function			
	Description	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 restore when restoring data.You can edit this file with a text editor on a computer to specify the data groups to restore.	Created.	Accessed.	Accessed.
Automatic transfer command file	This file specifies the data groups to transfer when automatically transferring data from an SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer.	Created.	Nothing is done.	Nothing is done.
Verification results files	These files contain the verification results after data is verified.	Nothing is done.	Nothing is done.	Created.

The execution method for the functions, applicable directory, and applicable operating modes are given in the following table.

Procedure	Directory*1	Applicable operating modes		
		Backing up data	Restoring data	Verifying data
CPU Unit front-panel DIP switch*2	The root directory	RUN mode and PROGRAM mode	When power is turned ON	RUN mode and PROGRAM mode

Procedure	Directory*1	Applicable operating modes		
		Backing up data	Restoring data	Verifying data
System-defined variables*3*4	The directory that you specified in the system-defined variable	RUN mode and PROGRAM mode	RUN mode and PROGRAM mode*5	RUN mode and PROGRAM mode
SD Memory Card Window in Sysmac Studio	The directory that you specified in the tab page	RUN mode and PROGRAM mode	Cannot be executed	RUN mode and PROGRAM mode

*1. You can specify a directory only on the SD Memory Card.
*2. Before you restore or verify data, save the backup file and restore command file in the root directory.
*3. This method is used to control the backup functions from an HMI. You can access the system-defined variables only for the restore from the user program.
*4. Make arrangements to prevent backup or verification operations from being performed on HMIs while a backup, restore, or verification operation is in progress. Otherwise, the intended operation may not occur.
*5. CPU Units with unit version 1.13 or earlier do not support restoring with the system-defined variables, and restoration cannot be executed.

9-2-1 Backup (Controller to SD Memory Card)

This operation is used to save data in the Controller to the SD Memory Card in the CPU Unit.

Processing Contents

- This backup operation processes all data groups.
- When you back up data, the backup file, restore command file, and automatic transfer command file are created in the specified directory on the SD Memory Card.
- If the backup-related files are already in the specified directory, they are overwritten.
- If an error occurs while writing the backup-related files to the SD Memory Card, the previous back-up-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 power is continued to supply even if the SD Memory Card power supply switch is pressed when a backup is in progress.
- The SD Memory Card will remain mounted after completion of the backup.

Procedure

● Backing Up Data with the CPU Unit Front-panel DIP Switch

Processing stage	Procedure
Start command	The backup starts when the SD Memory Card power supply switch is pressed for 3 seconds with the DIP switch pins set as follows: 1: OFF, 2: OFF, 3: ON, and 4: OFF.*1

Processing stage	Procedure
Executing	Immediately after Starting Backup*2 The SD PWR indicator will light, go out for 0.5 seconds, and then light again. While Backing Up Data The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	Normal End: The SD PWR indicator will light. Error End: The SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. Press the SD Memory Card power supply switch so that the indicator will light.

*1. For the NX701 CPU Unit, set all of pins 5 to 8 on the DIP switch to OFF.

*2. If an SD Memory Card is not inserted, the SD PWR indicator will not light.

● Backing Up Data with the *_Card1BkupCmd* (SD Memory Card Backup Command) System-defined Variable

Processing stage	Procedure
Start command	The name of the directory where the files are saved is stored in the *_Card1BkupCmd.DirName* (Directory Name) system-defined variable. Example: “dirA/dirB” specifies the dirB directory inside the dirA directory. The backup operation starts when you change the *_Card1BkupCmd.ExecBkup* (Execute Backup Flag) system-defined variable to TRUE.
Cancel command	You can cancel the backup operation. The backup operation ends in an error if you change the *_Card1BkupCmd.CancelBkup* (Cancel Backup Flag) system-defined variable to TRUE.
Executing	The *_Card1BkupSta.Active* (Active Flag) system-defined variable changes to TRUE. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	Normal End: The *_Card1BkupSta.Done* (Done Flag) system-defined variable changes to TRUE. Error End: The *_Card1BkupSta.Err* (Error Flag) system-defined variable changes to TRUE.

Note Do not use this system-defined variable from the user program.

● Backing Up Data from the SD Memory Card Window on the Sysmac Studio

Processing stage	Procedure
Start command	Click the **SD Memory Card Backup** Button on the SD Memory Card Window in the Sysmac Studio, specify the directory to save the backup file in, and execute the backup.
Executing	The progress of the backup is displayed in the dialog box. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	A message will appear when the backup is completed. You will then be asked to confirm whether to verify the backup data.

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

● Backing Up Data with Special Instruction

Processing stage	Procedure
Start command	Execute the BackupToMemoryCard instruction in the user program.
Executing	The value of the *Busy* output variable from the BackupToMemoryCard instruction will change to TRUE. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	Normal End: The value of the *Done* output variable from the BackupToMemoryCard instruction changes to TRUE. Error End: The value of the *Error* output variable from the BackupToMemoryCard instruction changes to TRUE. The error code is stored in the *ErrorID* output variable from the BackupToMemoryCard instruction.

Version Information

A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the BackupToMemoryCard instruction.

Related System-defined Variables

The system-defined variables that are related to the operation when system-defined variables are used to back up data are shown below. Refer to *A-8 Specifications for Individual System-defined Variables* on page A-163 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
ExecBkup*1	Execute Backup Flag	Change this variable to TRUE to back up Controller data to an SD Memory Card.	BOOL	RW
CancelBkup*1	Cancel Backup Flag	Change this variable to TRUE to cancel backing up data to an SD Memory Card.	BOOL	RW
DirName*1	Directory Name	Use this variable to specify the directory name in the SD Memory Card for which to back up data.	STRING[64]	RW
_Card1BkupSta*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. Do not use this system-defined variable from the user program.

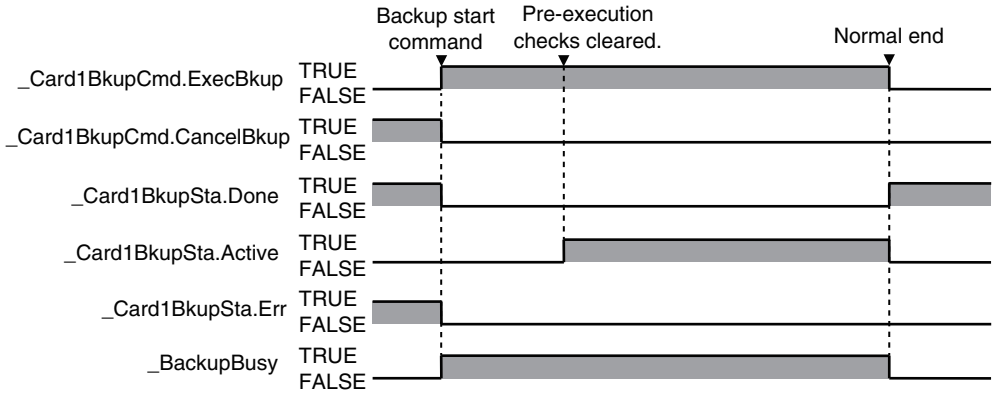
Additional Information

- Refer to the *NA-series Programmable Terminal Software User's Manual (Cat. No. V118)* for information on mapping variables when you connect an NA-series PT to the NJ/NX-series Controller.
- Refer to *A-12 Registering a Symbol Table on the CX-Designer* on page A-265 for the procedure to register these system-defined variables in the variable table of the CX-Designer when you connect an NS-series PT to the NJ/NX-series Controller.

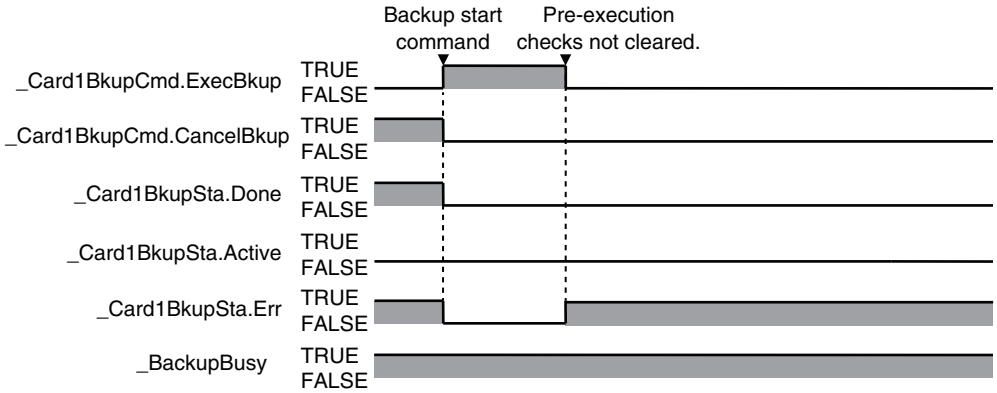
Timing Charts

The operation of the system-defined variables when they are used to backup data is shown below. In the charts, "pre-execution checks" indicates processing to check whether there is an SD Memory Card in the CPU Unit and other items before the backup starts. The value of *_Card1BkupSta.Active* (Active Flag) changes to TRUE only after all of the pre-execution checks are cleared and the actual backup is started.

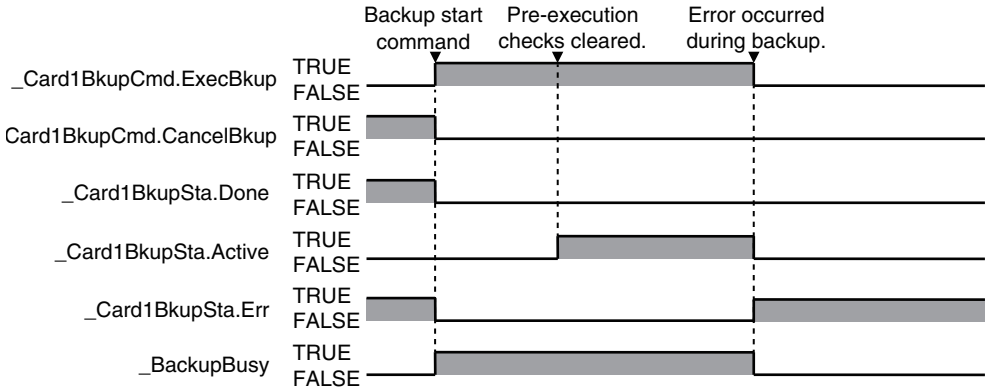
● Normal Operation



● Operation When the Backup Cannot Start Because Another Backup Function Is in Progress

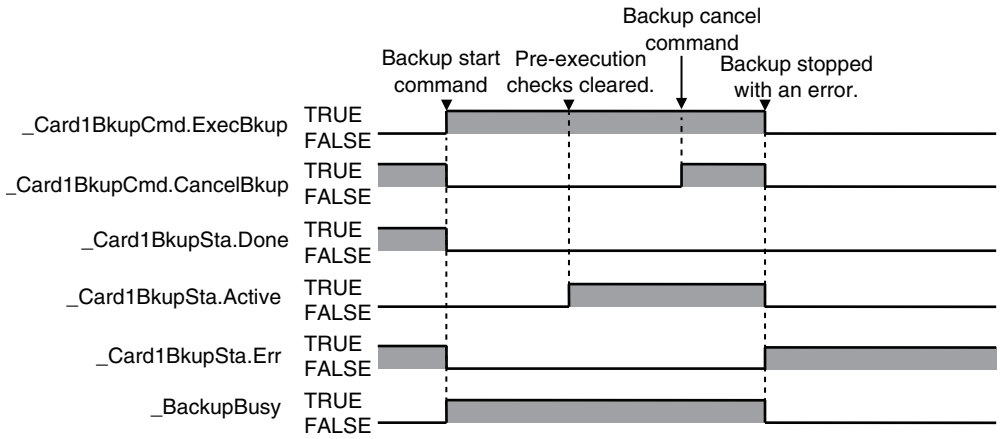


● Operation When the Backup Fails After a Normal Start



● Operation When the Backup Is Canceled While the Backup Is in Progress

The time required to stop the backup operation after it is canceled depends on the progress of the backup operation.



Processing Time and Backup File Size

The time that is required to back up the data depends on factors such as the CPU Unit, operating mode, Unit configuration, and user program. The size of the backup file depends on factors such as the Unit configuration and user program. Some guidelines for the backup time and backup file size are given in the following table.

CPU Unit	Operating mode	Connected EtherCAT slaves	Connected NX Units or CJ-series Units	Connected Option Boards	Number of user-defined POUs	User program memory size (MB)	Backup time (s)	Backup file size (MB)
NX701-□□□□	PROGRAM mode	*1	---	---	113	7.13	Approx. 30	29.45
NX502-□□□□		*7	---	---	69	5.7	Approx. 120	17.7
NX102-□□□□		*2	*3	---	15	0.37	Approx. 30	13.8
NX1P2-□□□□		*2	*3	*4	15	0.35	Approx. 30	2.96
NJ501-□□□□		*1	*5	---	53	2.36	Approx. 50	20
NJ301-□□□□		*6		---	20	0.53	Approx. 30	9.85
NJ101-□□□□		*2		---	15	0.38	Approx. 30	9.94

*1. Thirty-two each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.
*2. Two each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.
*3. One NX-PF0630 Additional I/O Power Supply Unit, three NX-ID5342 Digital Input Units, two NX-OD3153 Digital Output Units, one NX-AD4608 Analog Input Unit, and one NX-DA3605 Analog Output Unit.
*4. Two NX1W-CIF01 Serial Communications Option Boards.
*5. Four CJ1W-SCU22 Serial Communications Units and one CJ1W-EIP21 EtherNet/IP Unit.
*6. Eight each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.
*7. Thirty-two R88D-KNA-ECT AC Servo Drives and five NX-ECC201 EtherCAT Coupler Units.

9-2-2 Restore (SD Memory Card to Controller)

You can transfer the data in a backup file on the SD Memory Card in the CPU Unit to the Controller. You can use the front-panel DIP switch on the CPU Unit or system-defined variables to perform this operation.

The transfer starts when the power supply is turned ON with the CPU Unit front-panel DIP switch. You can specify data to restore by the restore command file. You cannot specify the source directory for backup-related files. The backup file to restore must be stored in the root directory on the SD Memory Card.

With system-defined variables, you can specify the data to restore and source directory for backup-related files, and give a command to start the transfer by a system-defined variable. You can specify whether to use this function or not and set a password in the Controller Setup. You can use this function to operate the CPU Unit with the data in a backup file on the SD Memory Card, by operating an HMI.

Processing Contents

The following describes the processing contents for restoring data with the CPU Unit front-panel DIP switch and with system-defined variables.

Restoring Data with the CPU Unit Front-panel DIP Switch

- The data in a backup file in the root directory on the SD Memory Card is transferred to the Controller.
- The data groups that are processed by the *restoration operation* in the RestoreCommand.ini file (restore command file) that is stored in the root directory. Refer to *9-14-3 Specifications of a Restore Command File* on page 9-72 for details on the restore command file.
- If there is not a restore command file in the root directory of the SD Memory Card, all of the data from the backup files in the root directory that can be transferred to the Controller will be transferred.
- After the operation is completed, the operating mode will change to PROGRAM mode. You cannot start operation in this state. To start operation, turn OFF all DIP switch pins and then cycle the power supply to the Controller or reset the Controller.
- Cycle the power supply to all of the EtherCAT slaves after you restore data.
- While the data is being restored, the CPU Unit will be in startup state.
- If an error occurs in the checks that are performed before starting to restore the data, the previous data will be retained in the Controller.
- If the power supply to the Controller is interrupted while the data is being restored, a *User Program/ Controller Configurations and Setup Transfer Error (a major fault level Controller error)* will occur. If that occurs, the data in the Controller is not dependable. Use one of the following methods to clear the error.
  - Perform the restore operation again.
  - Clear all of memory and then download the project from the Sysmac Studio.
- If the configuration for Units and slaves in the backup file does not match the actual configuration where data is restored, a Restore Execution Error will occur when you restore the data.
- The restore operation is possible even if the Option Board configuration in the backup file do not match the actual configuration where data is restored. However, the Option Board does not operate. Refer to the *NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual* (Cat. No. W579) for details.
- 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.
- For the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, memory for CJ-series Units is generated by the settings in the Memory Settings for CJ-series CPU Units in the backup file.
- If the present values of memory for CJ-series Units are not set to be restored in the NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit, the previous present values in the DM, EM and Holding Areas will be retained. However, when the DM, EM and Holding Areas are newly generated or the area is expanded, the values in those areas will be the initial values.
- The SD Memory Card will remain mounted after completion of the restore operation.
- The write protection for the CPU Unitthat is set in the *Write Protection at Startup* setting is used after completion of the restore operation.

Restoring Data with System-defined Variables

Precautions for Correct Use

- The data to be restored is the group of data specified with the system-defined variable. A specification of the restore command file (RestoreCommand.ini) is not affected. Refer to *Related System-defined Variables* on page 9-26 for details on the related system-defined variables.
- To prevent an unexpected restoration, set to enter the password every time before the restore operation.
- Executing this function automatically resets the Controller. The outputs during the Controller reset behave according to the slave and Unit specifications. Also, during the Controller reset, variables in the Controller cannot be accessed from the outside.
- If a variable on the Controller that was accessed before the transfer is deleted by the restoration, the system-defined variables may not be accessed because the deleted variable cannot be recovered by an HMI. For example with an NS-series PT, if the tag verification result finds any inconsistency, the list of tag verification result is displayed and the screen cannot be changed to others. Confirm, in advance, no variables that are used on the HMI are deleted.
- If the power is interrupted while this function is in progress, a User Program/Controller Configurations and Setup Transfer Error (event code 10200000 hex) or other errors may occur.
- You cannot execute the restore by the system-defined variable after you transfer a backup file for which **Restore by system-defined variable** is set to **Do not use** in the Controller Setup. If you intend to continue restoring, transfer a backup file for which the above setting is set to **Use**.
- You cannot execute other backup function while a restore operation is in progress.

- With the *_Card1RestoreCmd* (SD Memory Card Restore Command) system-defined variable, you can transfer the data saved in the SD Memory Card that is mounted on the CPU Unit to the Controller.
- The backup file to be restored is the file stored in the directory specified with the system-defined variable. The target backup file must be stored in a directory on the SD Memory Card in advance.
- If the password set on thePasswordof theSD Memory Card Restore Settingdiffers from the password set in the *_Card1RestoreCmd.Password* system-defined variable, the Restore Operation Failed to Start error occurs.
- When the restore is started, the password set in the *_Card1RestoreCmd.Password* system-defined variable is initialized.
- The Controller is automatically reset during the restore operation.
- After the Controller reset, the CPU Unit will be in startup state. After the restore operation is completed, the operating mode will change to PROGRAM mode. You cannot start operation in this state. To start operation, cycle the power supply to the Controller or reset the Controller.
- Cycle the power supply to all of the EtherCAT slaves after you restore data.
- If an error occurs in the checks that are performed before starting the restore operation or in the pre-execution checks, 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.
- All data items that are not specified for the restore will retain their present values.

- If the present values of variables that are set to be retained (with the Retain attribute) are not set to be transferred, the previous present values of those variables will be retained. However, the values of any variables that do not meet the retain conditions are initialized. These are the retain conditions for the variable:
  - The variable name, data type name, and data type size must be the same before and after transferring the data.
- If the present values of memory for CJ-series Units are not set to be restored in the NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit, the previous present values in the DM, EM and Holding Areas will be retained. However, when the DM, EM and Holding Areas are newly generated or the area is expanded, the values in those areas will be the initial values.
- The power is continued to supply even if the SD Memory Card power supply switch is pressed when a restore operation is in progress.
- The SD Memory Card will remain mounted after completion of the restore operation.
- The write protection for the CPU Unit that is set in the Write Protection at Startup setting is used after completion of the restore operation.

Procedure

● Restoring Data with the CPU Unit Front-panel DIP Switch

Processing stage	Procedure
Start command	Turn ON the power supply to the Controller with the DIP switch set as follows: 1: OFF, 2: OFF, 3: ON, and 4: ON. *1
Executing	While Restoring Data The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The RUN indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly.
Execution results	Normal End: The SD PWR indicator will light. Error End: The SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The indicator stop flashing and stay lit when the SD Memory Card power supply switch is pressed. *2

*1. For the NX701 CPU Unit, set all of pins 5 to 8 on the DIP switch to OFF.
*2. If an SD Memory Card is not inserted, the SD PWR indicator will not light.

● Restoring Data with the *_Card1RestoreCmd* (SD Memory Card Restore Command) System-defined Variable

Processing stage	Procedure
Pre-start preparation	To use the restore by the system-defined variable, set to use the **Restore by system-defined variable** in the Controller Setup. *1

Processing stage	Procedure
Start command	Specify the name of the directory where the backup files are saved in the *_Card1RestoreCmd.DirName* (Directory Name) system-defined variable. Example: “dirA/dirB” specifies the dirB directory inside the dirA directory. Specify a password in the *_Card1RestoreCmd.Password* (Password) system-defined variable. 2 Change the *_Card1RestoreCmdTargetIPAdr* (IP Address Transfer Flag) system-defined variable to TRUE to specify the IP address of the built-in EtherNet/IP port as the restore target. 3 Change the *_Card1RestoreCmdTargetVariable* (Present Values of Variables with the Retain Attribute Transfer Flag) system-defined variable to TRUE to specify the present values of variables with the Retain attribute as the restore target. Change the *_Card1RestoreCmdTargetMemory* (Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag) system-defined variable to TRUE to specify the present values of the memory used for CJ-series Units with the Retain attribute as the restore target. Change the *_Card1RestoreCmdTargetUnitConfig* (Unit and Slave Parameters Transfer Flag) system-defined variable to TRUE to specify Units and slaves settings as the restore target. Change the *_Card1RestoreCmdTargetAbsEncoder* (Absolute Encoder Home Offset Transfer Flag) system-defined variable to TRUE to specify the absolute encoder home offsets as the restore target. Change the _Card1RestoreCmdTargetOpcuaSPF (OPCUA Security Profile Transfer Flag) system-defined variable to TRUE to specify the OPC UA Security Profile as the restore target. The restore operation starts when you change the _Card1RestoreCmd.Exec (Execute Restore Flag) system-defined variable to TRUE.
Executing	The *_BackupBusy* (Backup Function Busy Flag) system-defined variable changes to TRUE. The *_Card1RestoreSta.Active* (Active Flag) system-defined variable changes to TRUE. While Restoring Data The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly. The RUN indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The Controller is automatically reset during the restore operation.
Execution results	Normal End: The SD PWR indicator will light. The RUN indicator goes out. The *_Card1RestoreSta.Done* (Done Flag) system-defined variable changes to TRUE. Error End at Checks Performed Before Restore Start: The SD PWR indicator will light.*4 RUN and ERR indicators are in the state before the restore starts. The *_Card1RestoreSta.Err* (Error Flag) system-defined variable changes to TRUE. The Controller is not reset for an error end at checks performed before restore start. Error End at Pre-execution Check or during Execution: The SD PWR indicator will light. *4 The RUN indicator goes out. The *_Card1RestoreSta.Err* (Error Flag) system-defined variable changes to TRUE.

*1. You cannot execute the restore by the system-defined variable after you transfer a backup file for which **Restore by system-defined variable** is set to **Do not use** in the Controller Setup. If you intend to continue restoring, transfer a backup file for which the above setting is set to **Use**.

*2. The password is initialized when the restore by the system-defined variable is started. Specify a password every time you start a restore.
If a password is not set on the Password of the SD Memory Card Restore Setting on the Controller Setup, the restore is started when the value of the *_Card1RestoreCmd.Password* system-defined variable is the initial value. The restore is not started if the value is not the initial value.
*3. The IP address means setting type, IP address, subnet mask, and default gateway.
*4. If an SD Memory Card is not inserted, the SD PWR indicator will not light.

Related System-defined Variables

The following table lists the related system-defined variables. Refer to *A-8 Specifications for Individual System-defined Variables* on page A-163 for details on system-defined variables.

Variable name / Member name	Meaning	Function	Data type	R/W
_Card1RestoreCmd	SD Memory Card Restore Command		_sRESTORE_CMD	RW
Exec	Execute Restore Flag	Change this variable to TRUE to restore the data in a backup file on the SD Memory Card to the Controller.	BOOL	RW
DirName	Directory Name	Use this variable to specify the directory name in the SD Memory Card in which the backup file to be restored by the system-defined variable is stored.	STRING[64]	RW
Password	Password	Use this variable to specify the password that is used for verification when you start the restore by the system-defined variable. The password is initialized every time when the restore by the system-defined variable is started.	STRING[33]	RW
_Card1RestoreSta	SD Memory Card Restore Status		_sRESTORE_STA	R
Done	Done Flag	TRUE when a restore operation is completed.	BOOL	R
Active	Active Flag	TRUE when a restore operation is in progress.	BOOL	R
Err	Error Flag	TRUE when a restore operation ended in an error.	BOOL	R

Variable name	Meaning	Function	Data type	R/W
_Card1RestoreCmdTargetUserProgram	User Program and Settings Transfer Flag	Change this variable to TRUE to set a user program or setting for the restore by the system-defined variable as the transfer target. Always set this variable to TRUE for the restore by the system-defined variable.	BOOL	RW

Variable name	Meaning	Function	Data type	R/W
_Card1Restor-eCmdTargetIPAdr	IP Address Transfer Flag	Change this variable to TRUE to include the IP address of the built-in EtherNet/IP port for the restore by the system-defined variable as the transfer target. The IP address means setting type, IP address, subnet mask, and default gateway.	BOOL	RW
_Card1Restor-eCmdTargetVariable	Present Values of Variables with the Retain Attribute Transfer Flag	Change this variable to TRUE to set the present values of variables with the Retain attribute for the restore by the system-defined variable as the transfer target.	BOOL	RW
_Card1Restor-eCmdTargetMemory	Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag	Change this variable to TRUE to set the present values of the memory used for CJ-series Units with the Retain attribute for the restore by the system-defined variable as the transfer target.	BOOL	RW
_Card1Restor-eCmdTargetUnit-Config	Unit and Slave Parameters Transfer Flag	Change this variable to TRUE to set the Unit and slave settings for the restore by the system-defined variable as the transfer target.	BOOL	RW
_Card1Restor-eCmdTargetAbsEncoder	Absolute Encoder Home Offset Transfer Flag	Change this variable to TRUE to set the absolute encoder home offset for the restore by the system-defined variable as the transfer target.	BOOL	RW

Additional Information

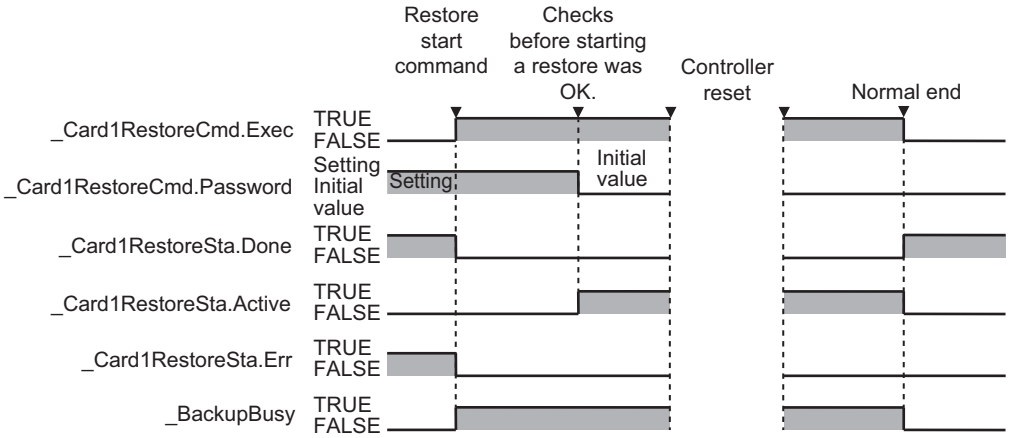
- Refer to the *NA-series Programmable Terminal Software User's Manual (Cat. No. V118)* for information on mapping variables when you connect an NA-series PT to the NJ/NX-series Controller.
- Refer to *A-12 Registering a Symbol Table on the CX-Designer* on page A-265 for the procedure to register these system-defined variables in the variable table of the CX-Designer when you connect an NS-series PT to the NJ/NX-series Controller.

Timing Charts

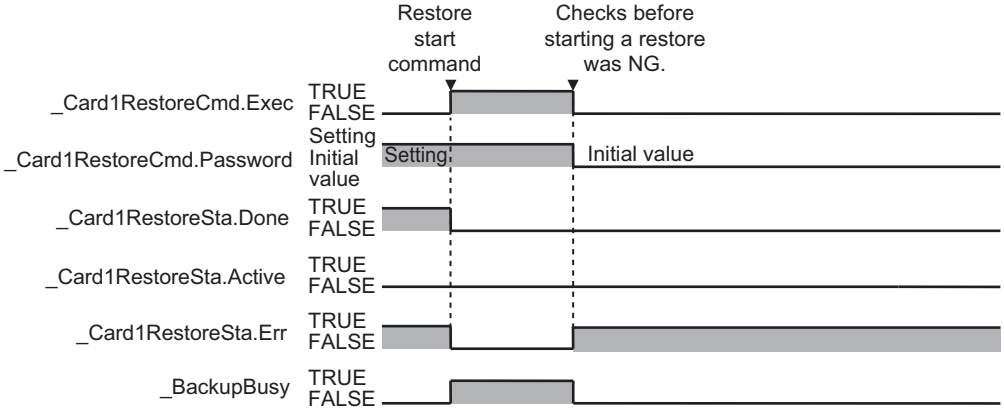
The operation of the system-defined variables when they are used for the restart of SD Memory Card backups is shown below.

In the charts, "checks performed before starting" indicates the processing performed before the restore operation to check whether the password matches. The value of *_Card1PrgRestoreSta.Active* (Active Flag) changes to TRUE only after the checks performed before starting result in OK.

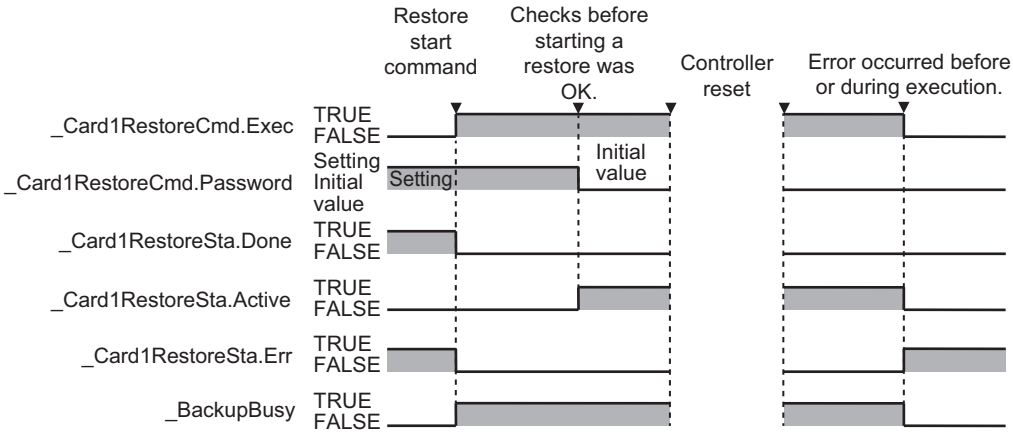
● Normal Operation



● Operation When the Restoration Cannot Start Because Another Backup Function Is in Progress



● Operation When the Restoration Fails After a Normal Start



Processing Time

The time that is required to restore the data depends on factors such as the CPU Unit, Unit configuration, and user program. Guidelines for the restoration time are given in the following table.

CPU Unit	Connected EtherCAT slaves	Connected NX Units or CJ-series Units	Connected Option Boards	Number of user-defined POUs	User program memory size (MB)	Restoration time (s)
NX701-□□□□	*1	*3	---	113	7.13	Approx. 50
NX502-□□□□	*7	---	---	69	5.7	Approx. 160
NX102-□□□□	*2	*3	---	15	0.37	Approx. 70
NX1P2-□□□□	*2	*3	*4	15	0.35	Approx. 60
NJ501-□□□□	*1	*5	---	53	2.36	Approx. 100
NJ301-□□□□	*6		---	20	0.53	Approx. 70
NJ101-□□□□	*2		---	15	0.38	Approx. 70

*1. Thirty-two each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.
*2. Two each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1600 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.
*3. One NX-PF0630 Additional I/O Power Supply Unit, three NX-ID5342 Digital Input Units, two NX-OD3153 Digital Output Units, one NX-AD4608 Analog Input Unit, and one NX-DA3605 Analog Output Unit.
*4. Two NX1W-CIF01 Serial Communications Option Boards.
*5. Four CJ1W-SCU22 Serial Communications Units and one CJ1W-EIP21 EtherNet/IP Unit.
*6. Eight each of the following: R88D-KNA-ECT AC Servo Drives, GX-ID1611 Digital I/O Terminals, and GX-OD1611 Digital I/O Terminals.
*7. Thirty-two R88D-KNA-ECT AC Servo Drives and five NX-ECC201 EtherCAT Coupler Units.

9-2-3 Verify (between Controller and SD Memory Card)

You can compare the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit.

Processing Contents

- The Controller data and the data in a backup file that is saved in the specified directory of the SD Memory Card are compared.
- The data groups that are processed by the *restoration operation* are specified in the RestoreCommand.ini file (restore command file).
- The present values of variables, the present values in memory used for the CJ-series Units, and the absolute encoder home offsets are not compared because these values may change while the verification is in process.
- When you verify the data, the *verification results file (VerifyResult.log)* is created in the specified directory. The verification results are stored in this file. If a verification results file already exists in the specified directory, it will be overwritten. However, if the SD Memory Card is write-protected, the verification results files will not be created.
- If there is not a restore command file in the specified directory of the SD Memory Card, all of the data from the backup files in the specified directory that can be compared will be compared.
- If the Unit and slave configuration in the backup file is not the same as the actual configuration of the Controller, a Verification Error will occur.
- The SD Memory Card will remain mounted after completion of the verification operation.

Procedure

● Verifying Data with the CPU Unit Front-panel DIP Switch

Processing stage	Procedure
Start command	The verification operation starts when the SD Memory Card power supply switch is pressed for 3 seconds with the DIP switch pins set as follows: 1: OFF, 2: OFF, 3: OFF, and 4: OFF.*1
Executing	Immediately after Starting Verification*2 The SD PWR indicator will light, go out for 0.5 seconds, and then light again. While Verifying Data The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE.
Execution results	Normal End with No Differences Found: The SD PWR indicator will light. Normal End with Differences Found: The SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The indicator stop flashing and stay lit when the SD Memory Card power supply switch is pressed. Error End: The SD PWR indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. Press the SD Memory Card power supply switch so that the indicator will light.

*1. For the NX701 CPU Unit, set all of pins 5 to 8 on the DIP switch to OFF.
*2. If an SD Memory Card is not inserted, the SD PWR indicator will not light.

● Verifying 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 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.

Processing stage	Procedure
Execution results	Normal End with No Differences Found: The *_Card1BkupSta.Done* (Done Flag) system-defined variable and the *_Card1BkupSta.VefyRslt* (Verify Result Flag) system-defined variable change to TRUE. Normal End with Differences Found: The *_Card1BkupSta.Done* (Done Flag) system-defined variable changes to TRUE and the *_Card1BkupSta.VefyRslt* (Verify Result Flag) system-defined variable changes to FALSE. Error End: The *_Card1BkupSta.Err* (Error Flag) system-defined variable changes to TRUE.

Note Do not use this system-defined variable from the user program.

● Verifying Data from the SD Memory Card Window on the Sysmac Studio

Processing stage	Procedure
Start command	Click the **Compare SD Memory Card Backup** Button on the SD Memory Card Window in Sysmac Studio, specify the directory that contains the file to compare, and execute the verification.
Executing	The progress of the verification is displayed in the dialog box. The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will change to TRUE. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly.
Execution results	The results of the verification are displayed in the dialog box.

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

Related System-defined Variables

The system-defined variables that are related to the operation when system-defined variables are used to verify data are shown below. Refer to *A-8 Specifications for Individual System-defined Variables* on page A-163 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 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 SD Memory Card.	BOOL	RW
DirName*1	Directory Name	Use this variable to specify the directory name in the SD Memory Card for which to back up data.	STRING[64]	RW
_Card1VefySta*1	SD Memory Card Verify Status		_sVEFY_STA	R

Variable name		Meaning	Function	Data type	R/W
	Member name				
	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. Do not use this system-defined variable from the user program.

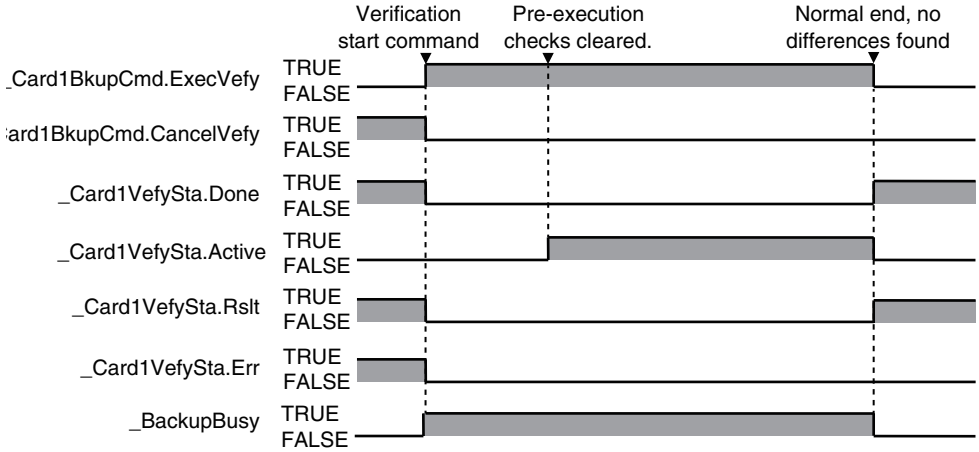
Additional Information

- Refer to the NA-series Programmable Terminal Software User's Manual (Cat. No. V118) for information on mapping variables when you connect an NA-series PT to the NJ/NX-series Controller.
- Refer to *A-12 Registering a Symbol Table on the CX-Designer* on page A-265 for the procedure to register these system-defined variables in the variable table of the CX-Designer when you connect an NS-series PT to the NJ/NX-series Controller.

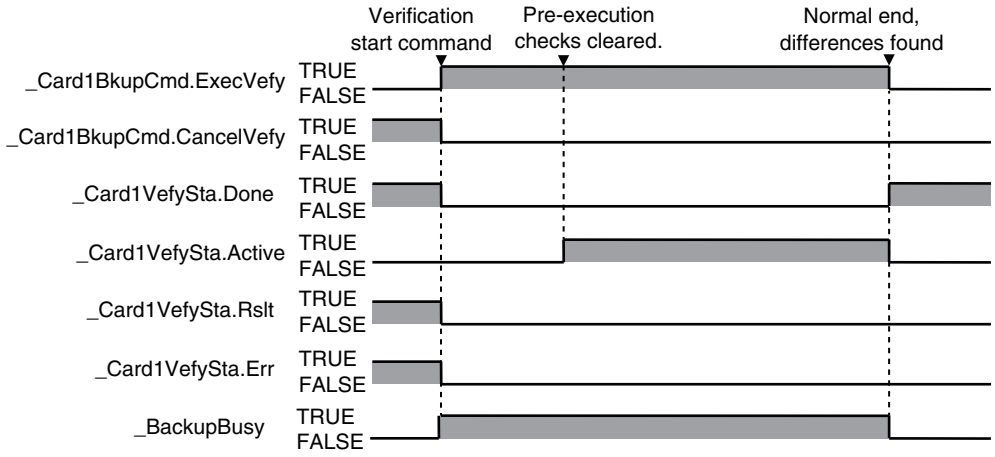
Timing Charts

The operation of the system-defined variables when they are used to verify data is shown below.
In the charts, "pre-execution checks" indicates processing to check whether there is an SD Memory Card in the CPU Unit and other items. The value of *_Card1VefySta.Active* (Active Flag) changes to TRUE only after all of the pre-execution checks are cleared and the actual verification is started.

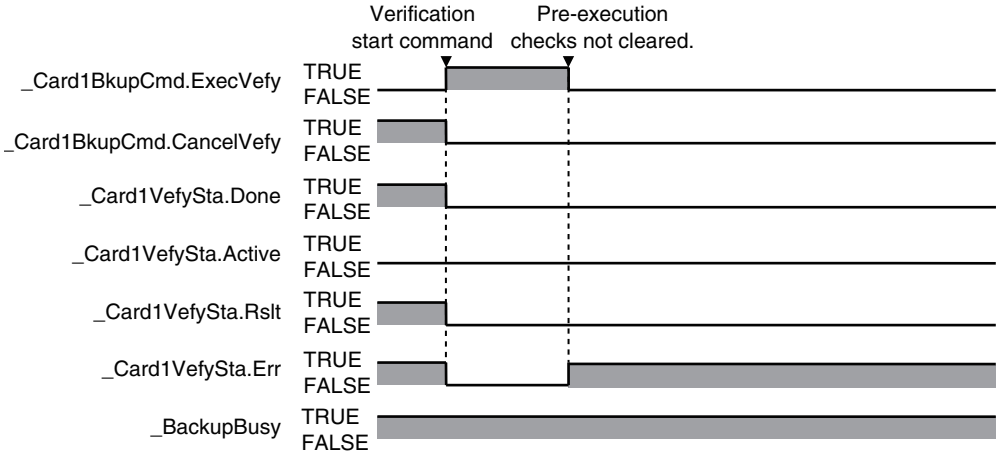
● Normal End with No Differences Found



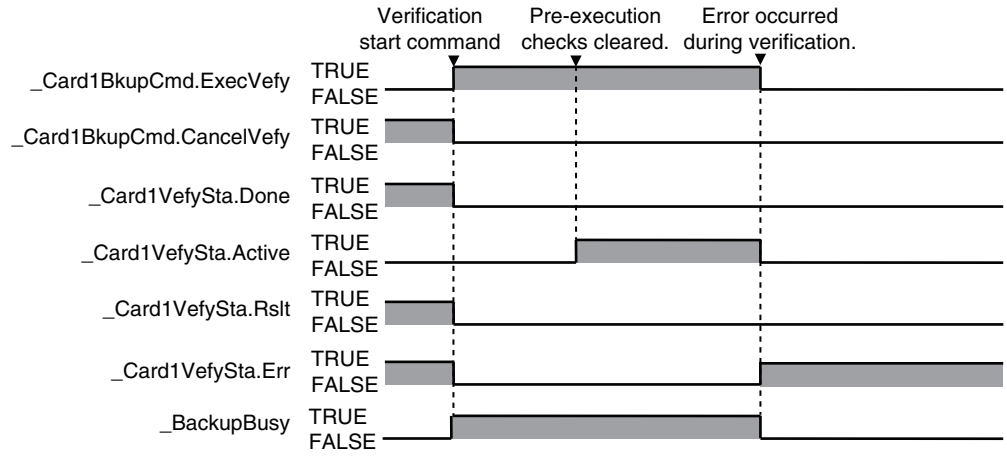
● Normal End with Differences Found



● Operation When the Verification Cannot Start Because Another Backup Function Is in Progress

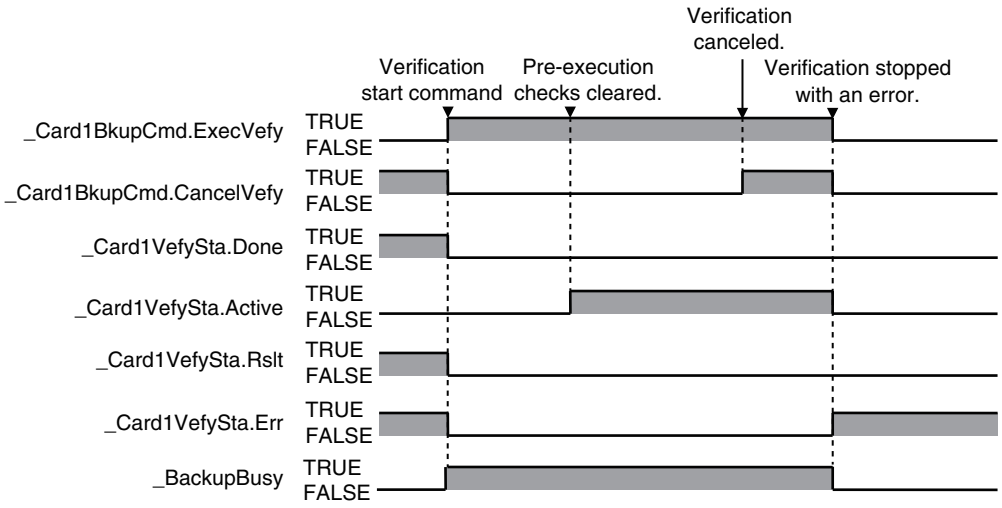


● Operation When the Verification Fails After a Normal Start



● Operation When the Operation Is Canceled While Verification Is in Progress

The time required to stop the verification operation after it is canceled depends on the progress of the verification operation.



9-3 Disabling Backups to SD Memory Cards

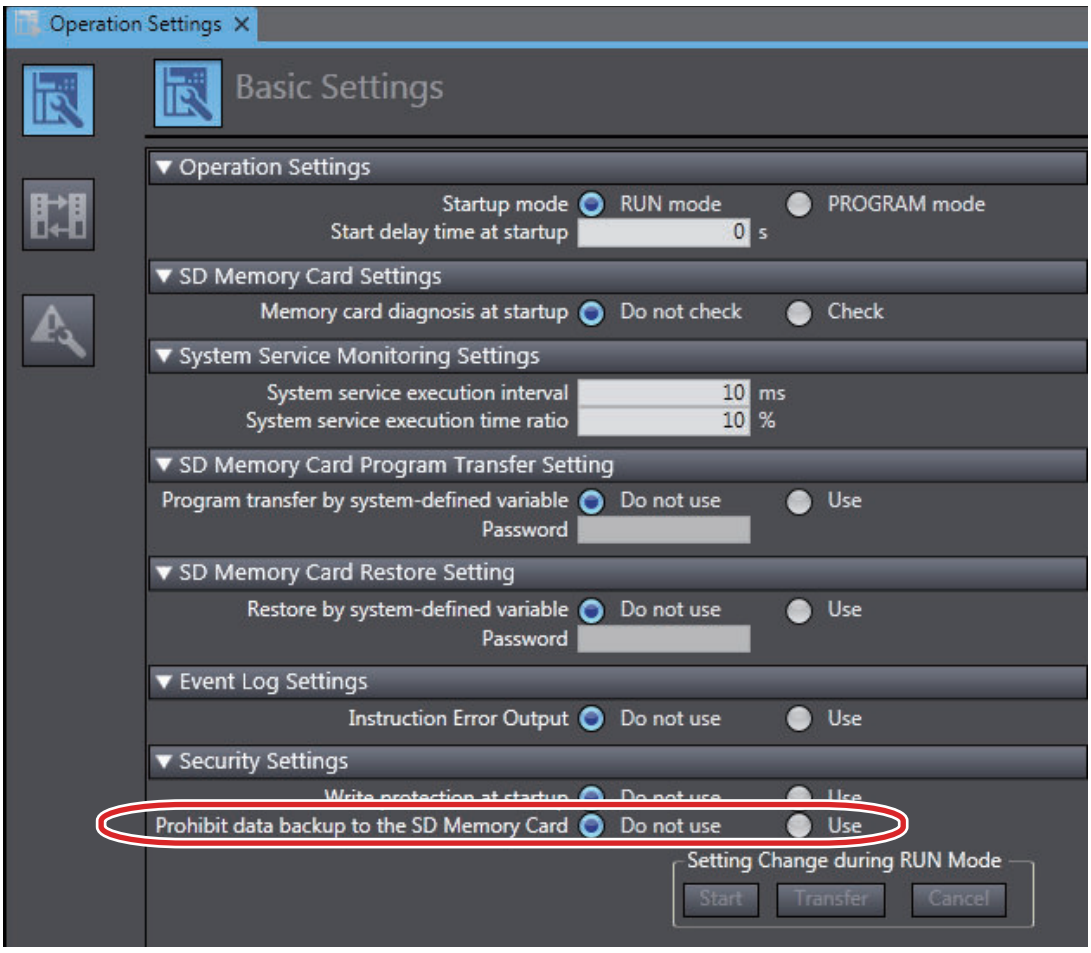
You can disable the backup function from writing data to the SD Memory Card to protect your programming assets.

The following three functions are applicable for disabling backup to SD Memory Card.

- Backups using the CPU Unit front-panel DIP switch
- Backups using system-defined variables
- Backups from the SD Memory Card Window on the Sysmac Studio

Backup function using the BackupToMemoryCard instruction is not applicable. This means that you can backup data using the BackupToMemoryCard instruction even if the Prohibit data backup to the SD Memory Card setting is set to be used.

Use the following procedure to set the *Prohibit data backup to the SD Memory Card* setting.
Select the **Use** Option for the Prohibit data backup to the SD Memory Card setting in the Basic Settings Display of the **Operation Settings** Tab Page under **Configurations and Setup − Controller Setup** on the Sysmac Studio.

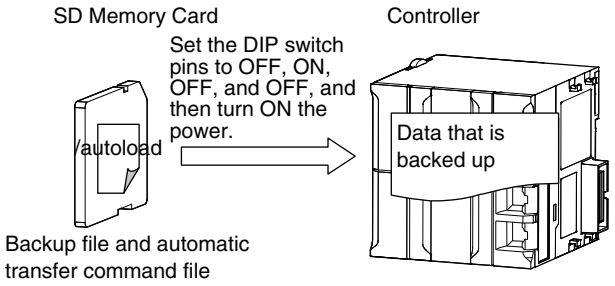


Version Information

A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the BackupToMemoryCard instruction.

9-4 Automatic Transfers from SD Memory Cards

This function automatically transfers the data in a backup file to the Controller when the power supply is turned ON. The backup file must be stored in the */autoload* directory on the SD Memory Card in the CPU Unit.
You can use this to operate the CPU Unit with the data in a backup file on the SD Memory Card.
The only way to perform this operation is to use the front-panel DIP switch on the CPU Unit.



The automatic transfer uses a backup file that is created with the backup function and an automatic transfer command file. Save both files in the */autoload* directory in advance.

File	Function
Backup file	This file contains the Controller data that is handled by the functions that are related to data backup.
Automatic transfer command file	This file specifies the data groups to transfer when transferring data from an SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer.

The following tables gives the procedure, the applicable directory, and the timing at which the transfer is executed.

Procedure	Directory	Execution timing
CPU Unit front-panel DIP switch	*/autoload* directory on the SD Memory Card	At startup

Processing Contents

- When the power is turned ON, the data in the backup file in the */autoload* directory on the SD Memory Card is automatically transferred to the Controller.
- The automatic transfer function transfers the data in the data groups that are specified in the AutloadCommand.ini file in the */autoload* directory. Refer to *9-14-4 Specifications of an Automatic Transfer Command File* on page 9-74 for details on the automatic transfer command file.
- If an AutoloadCommand.ini file is not in the */autoload* directory on the SD Memory Card, all of the data from the backup file in the */autoload* directory that can be transferred will be transferred.
- The operating mode that is set in the Startup Mode setting in the Controller Setup is used after completion of the automatic transfer.
- While the data is being automatically transferred, the CPU Unit will be in startup state.

- If an error occurs in the checks that are performed before starting the automatic transfer, the previous data will be retained in the Controller.
- If the power supply to the Controller is interrupted while the data is being automatically transferred, a User Program/Controller Configurations and Setup Transfer Error (a major fault level Controller error) will occur. If that occurs, the data in the Controller is not dependable. Use one of the following methods to clear the error.
  - Perform the automatic transfer again.
  - Clear all of memory and then download the project from the Sysmac Studio.
- All data items that are not specified for the automatic transfer will retain their present values.
- If the present values of variables that are set to be retained (with the Retain attribute) are not set to be transferred, the previous present values of those variables will be retained. However, the values of any variables that do not meet the retain conditions are initialized. These are the retain conditions for the variable:
  - The variable name, data type name, and data type size must be the same before and after transferring the data.
- For the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, memory for CJ-series Units is generated by the settings in the Memory Settings for CJ-series CPU Units in the backup file.
- If the present values of memory for CJ-series Units are not set to be restored in the NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit, the previous present values in the DM, EM and Holding Areas will be retained. However, the values in the DM, EM and Holding Areas, which are newly generated or the area is expanded, will be the initial values.
- The SD Memory Card will remain mounted after completion of the automatic transfer operation.
- The write protection for the CPU Unit that is set in the Write Protection at Startup setting is used after completion of the automatic transfer operation.

Procedure

● Transferring Data with the CPU Unit Front-panel DIP Switch

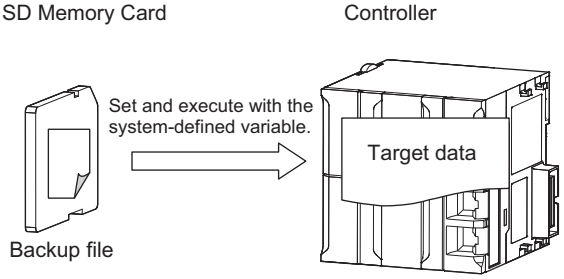
Processing stage	Procedure
Start command	Turn ON the power supply to the Controller with the DIP switch set as follows: 1: OFF, 2: ON, 3: OFF, and 4: OFF.*1
Executing	The transfer is in progress. The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The RUN indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly.
Execution results	Normal End: The SD PWR indicator will light. The operating mode that is set in the Startup Mode setting in the Controller Setup is used after completion of the transfer. Error End: The RUN indicator goes out, the ERR indicator lights, and a major fault level Controller error occurs. The SD PWR indicator will light. *2

*1. For the NX701 CPU Unit, set all of pins 5 to 8 on the DIP switch to OFF.

*2. If an SD Memory Card is not inserted, the SD PWR indicator will not light.

9-5 Program Transfer from SD Memory Card

With the *_Card1PrgTransferCmd* (SD Memory Card Program Transfer Command) system-defined variable, you can transfer a program stored in the SD Memory Card that is mounted on the CPU Unit to the Controller.
You can specify whether to use this function or not and set a password in the Controller Setup.
You can use this function to operate the CPU Unit with the program in a backup file on the SD Memory Card, by operating an HMI.



The transfer uses a backup file that is created with the backup function. Save the backup file in a directory on the SD Memory Card in advance.
Use the system-defined variable to specify the directory that contains the backup file.

File	Function
Backup file	This file contains the Controller data that is handled by the functions that are related to data backup.

The execution method for the functions, applicable directory, and applicable operating modes are given in the following table.

Operating method	Directory*1	Applicable operating modes
System-defined variable	The directory that you specified in the system-defined variable	RUN mode and PROGRAM mode

*1. You can specify a directory only on the SD Memory Card.

Precautions for Correct Use

- When you use this function to transfer a program whose Startup Mode setting is set to RUN mode, the operating mode changes to RUN after the transfer is completed regardless of the status and setting before the transfer. Use this function after you confirm that system startup does not cause any problem.
- To prevent an unexpected transfer of a program, set to enter the password every time before a transfer.
- Executing this function automatically resets the Controller. The outputs during the Controller reset behave according to the slave and Unit specifications. Also, during the Controller reset, variables in the Controller cannot be accessed from the outside.
- When an EtherCAT slave is used or a motion control is executed, an error may occur in the EtherCAT Master Function Module or Motion Control Function Module after the program transfer is completed. If an error occurs, reset the error after the program transfer is completed. Refer to sample programming for the ResetMCError instruction in the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for details on resetting errors in the user program.
- If a variable on the Controller that was accessed before the transfer is deleted by the program transfer, the system-defined variables may not be accessed because the deleted variable cannot be recovered by an HMI. For example with an NS-series PT, if the tag verification result finds any inconsistency, the list of tag verification result is displayed and the screen cannot be changed to others. Confirm, in advance, no variables that are used on the HMI are deleted.
- If the power is interrupted while this function is in progress, a User Program/Controller Configurations and Setup Transfer Error (event code 10200000 hex) or other errors may occur.
- You cannot transfer programs after you transfer a backup file for which **Program transfer by system-defined variable** is set to **Do not use** in the Controller Setup. If you intend to continue transferring programs, transfer a backup file for which the above setting is set to **Use**.
- You cannot execute other backup function while a program transfer is in progress.

Processing Contents

- With the _Card1PrgTransferCmd (SD Memory Card Program Transfer Command) system-defined variable, you can transfer a program saved in the SD Memory Card that is mounted on the CPU Unit to the Controller.
- The backup file of the programs to be transferred is the file stored in the directory specified with the system-defined variable. The target backup file must be stored in a directory on the SD Memory Card in advance.
- The data of the programs to be transferred is the group of data specified with the system-defined variable. Refer to *Related System-defined Variables* on page 9-43 for details on the related system-defined variables.
- If the password set on the **Password** of the **SD Memory Card Program Transfer Setting** differs from the password set in the _Card1PrgTransferCmd.Password system-defined variable, the *SD Memory Card Program Transfer Failed to Start* error occurs.
- When the program transfer is started, the password set in _Card1PrgTransferCmd.Password system-defined variable is initialized.
- The Controller is automatically reset during the program transfer. The outputs during the Controller reset behave according to the slave and Unit specifications.
- After the Controller reset, the CPU Unit will be in *startup state*. Then when the program transfer is completed, the CPU Unit will be in *normal operation state* and operate in the operating mode set in the **Startup Mode** setting in the **Controller Setup**, which was transferred by the program transfer function.

However, the CPU Unit will operate in PROGRAM mode when the CPU Unit front-panel DIP switch is set to the Safe Mode.

- If an error occurs in the checks that are performed before starting the program transfer or in the pre-execution checks, the previous data will be retained in the Controller.
- If the power supply to the Controller is interrupted while the user program transfer is in progress, 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.
  - Transfer the programs from the SD Memory Card again.
  - Use the automatic transfer from SD Memory Card function, or restore with SD Memory Card backup function.
  - Clear all of memory and then download the project from the Sysmac Studio.
- All data items that are not specified for the program transfer will retain their present values.
- If the present values of variables that are set to be retained (with the Retain attribute) are not set to be transferred, the previous present values of those variables will be retained. However, the values of any variables that do not meet the retain conditions are initialized. These are the retain conditions for the variable:
  - The variable name, data type name, and data type size must be the same before and after transferring the data.
- For the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, memory for CJ-series Units is generated by the settings in the Memory Settings for CJ-series CPU Units in the backup file.
- If the present values of memory for CJ-series Units are not set to be restored in the NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit, the previous present values in the DM, EM and Holding Areas will be retained. However, when the DM, EM and Holding Areas are newly generated or the area is expanded, the values in those areas will be the initial values.
- The power is continued to supply even if the SD Memory Card power supply switch is pressed while the program transfer is in progress.
- The SD Memory Card will remain mounted after completion of the program transfer operation.
- The write protection for the CPU Unit that is set in the *Write Protection at Startup* setting is used after completion of the program transfer operation.

Procedure

● *_Card1PrgTransferCmd* (SD Memory Card Program Transfer Command) System-defined Variable

Processing stage	Procedure
Pre-start preparation	To use the program transfer from the SD Memory Card, set to use the **Program transfer by system-defined variable** in the Controller Setup. *1

Processing stage	Procedure
Start command	Specify the name of the directory where the backup files are saved in the *_Card1PrgTransferCmd.DirName* (Directory Name) system-defined variable. Example: “dirA/dirB” specifies the dirB directory inside the dirA directory. Specify a password in the *_Card1PrgTransferCmd.Password* (Password) system-defined variable. *2 Change the *_Card1PrgTransferCmd.TargetIPAdr* (IP Address Transfer Flag) system-defined variable to TRUE to specify the built-in EtherNet/IP port settings as the transfer target. *3 Change the *_Card1PrgTransferCmd.TargetVariable* (Present Values of Variables with the Retain Attribute Transfer Flag) system-defined variable to TRUE to specify the present values of variables with the Retain attribute as the transfer target. Change the *_Card1PrgTransferCmd.TargetMemory* (Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag) system-defined variable to TRUE to specify the present values of the memory used for CJ-series Units with the Retain attribute as the transfer target. Change the *_Card1PrgTransferCmd.Exec* (Execute Program Transfer Flag) system-defined variable to TRUE to start the transfer operation.
Executing	The *_BackupBusy* (Backup Function Busy Flag) system-defined variable changes to TRUE. The *_Card1PrgTransferSta.Active* (Active Flag) system-defined variable changes to TRUE. During program transfer The SD PWR indicator will flash, lighting for 3 seconds and going out for 0.5 seconds. The SD BUSY indicator will flash irregularly. The RUN indicator will flash, lighting for 0.5 seconds and going out for 0.5 seconds. The Controller is automatically reset during the program transfer.
Execution results	Normal End: The SD PWR indicator will light. The operating mode that is set in the Startup Mode setting in the Controller Setup is used after completion of the transfer. The *_Card1PrgTransferSta.Done* (Done Flag) system-defined variable changes to TRUE. Error End at Checks Performed Before Transfer Start: The SD PWR indicator will light. *4 RUN and ERR indicators are in the state before the transfer starts. The *_Card1PrgTransferSta.Err* (Error Flag) system-defined variable changes to TRUE. The Controller is not reset for an error end at checks performed before transfer start. Error End at Pre-execution Check or during Execution: The SD PWR indicator will light. *4 The RUN indicator goes out, the ERR indicator lights, and a major fault level Controller error occurs. The *_Card1PrgTransferSta.Err* (Error Flag) system-defined variable changes to TRUE.

*1. You cannot transfer programs after you transfer a backup file for which **Program transfer by system-defined variable** is set to **Do not use** in the Controller Setup. If you intend to continue transferring programs, transfer a backup file for which the above setting is set to **Use**.

*2. The password is initialized when transferring programs from the SD Memory Card is started. Specify a password every time you start a transfer.
If a password is not set on the **Password** of the **SD Memory Card Program Transfer Setting** on the Controller Setup, the program transfer is started when the value of the *_Card1PrgTransferCmd.Password* system-defined variable is the initial value. The transfer is not started if the value is not the initial value.

*3. The IP address means setting type, IP address, subnet mask, and default gateway.

*4. If an SD Memory Card is not inserted, the SD PWR indicator will not light.

Related System-defined Variables

The following table lists the related system-defined variables. Refer to *A-8 Specifications for Individual System-defined Variables* on page A-163 for details on system-defined variables.

Variable name / Member name	Meaning	Function	Data type	R/W
_Card1PrgTransferCmd	SD Memory Card Program Transfer Command		_sPRGTRANSFER_CMD	RW
Exec	Execute Program Transfer Flag	Change this variable to TRUE to transfer the data in a backup file on the SD Memory Card to the Controller by using the function to transfer programs from the SD Memory Card.	BOOL	RW
DirName	Directory Name	Use this variable to specify the directory name in the SD Memory Card for which to back up data.	STRING[64]	RW
Password	Password	Use this variable to specify the password that is used for verification when you start transferring the programs. The password is initialized every time you start transferring programs from the SD Memory Card.	STRING[33]	RW
TargetUserProgram	User Program and Settings Transfer Flag*1	Change this variable to TRUE to set a user program or setting as the transfer target. Always set this variable to TRUE for transferring programs from SD Memory Card.	BOOL	RW
TargetIPAdr	IP Address Transfer Flag	Change this variable to TRUE to include the IP address of the built-in EtherNet/IP port as the transfer target. The IP address means setting type, IP address, subnet mask, and default gateway.	BOOL	RW
TargetVariable	Present Values of Variables with the Retain Attribute Transfer Flag	Change this variable to TRUE to set the present values of variables with the Retain attribute as the transfer target.	BOOL	RW
TargetMemory	Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag	Change this variable to TRUE to set the present values of the memory used for CJ-series Units with the Retain attribute as the transfer target.	BOOL	RW
_Card1PrgTransferSta	SD Memory Card Program Transfer Status		_sPRGTRANSFER_STA	R

Variable name Member name	Meaning	Function	Data type	R/W
Done	Done Flag	TRUE when a program transfer is completed.	BOOL	R
Active	Active Flag	TRUE when a program transfer is in progress.	BOOL	R
Err	Error Flag	TRUE when a program transfer ended in an error.	BOOL	R

*1. The data in the user program and setting data groups is the target of settings. However, the IP address for the built-in EtherNet/IP port is set with the IP Address Transfer Flag.

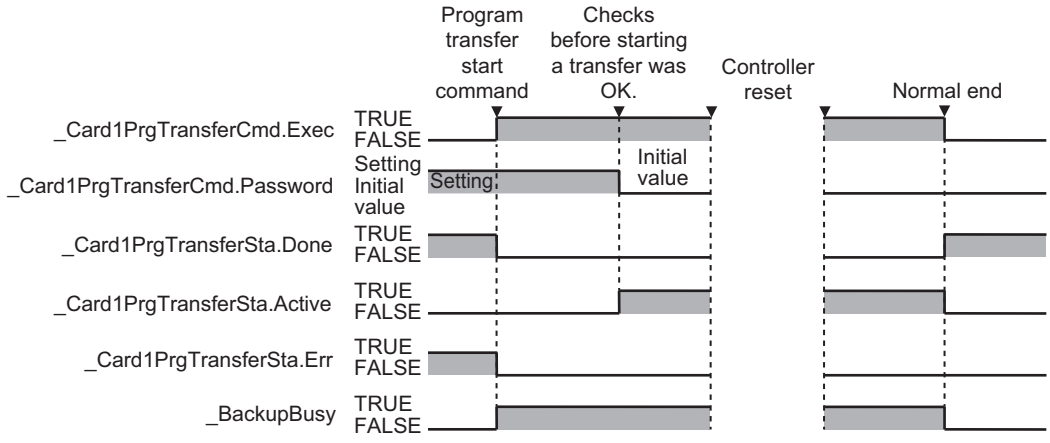
Additional Information

- Refer to the NA-series Programmable Terminal Software User's Manual (Cat. No. V118) for information on mapping variables when you connect an NA-series PT to the NJ/NX-series Controller.
- Refer to *A-12 Registering a Symbol Table on the CX-Designer* on page A-265 for the procedure to register these system-defined variables in the variable table of the CX-Designer when you connect an NS-series PT to the NJ/NX-series Controller.

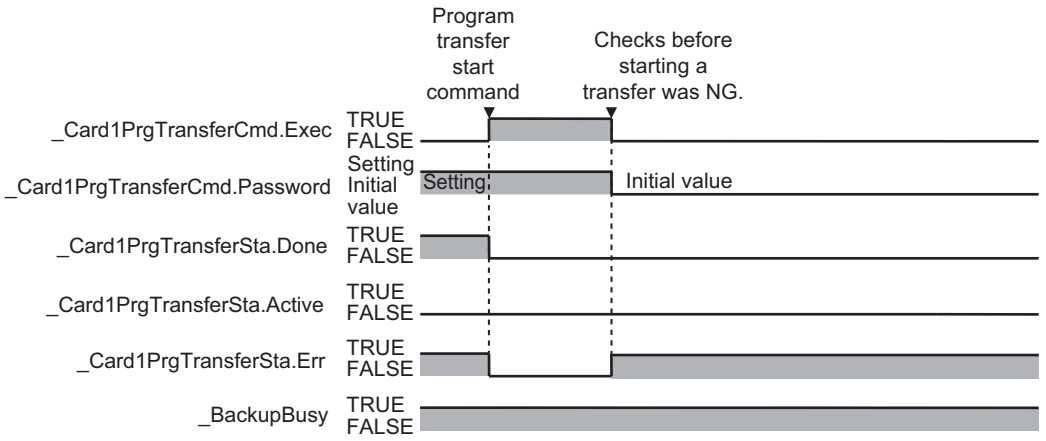
Timing Charts

The operation of the system-defined variables when they are used to transfer programs from the SD Memory Card is shown below.
In the charts, "checks performed before starting" indicates the processing performed before the program transfer to check whether the password matches. The value of *_Card1PrgTransferSta.Active* (Active Flag) changes to TRUE only after the checks performed before starting result in OK.

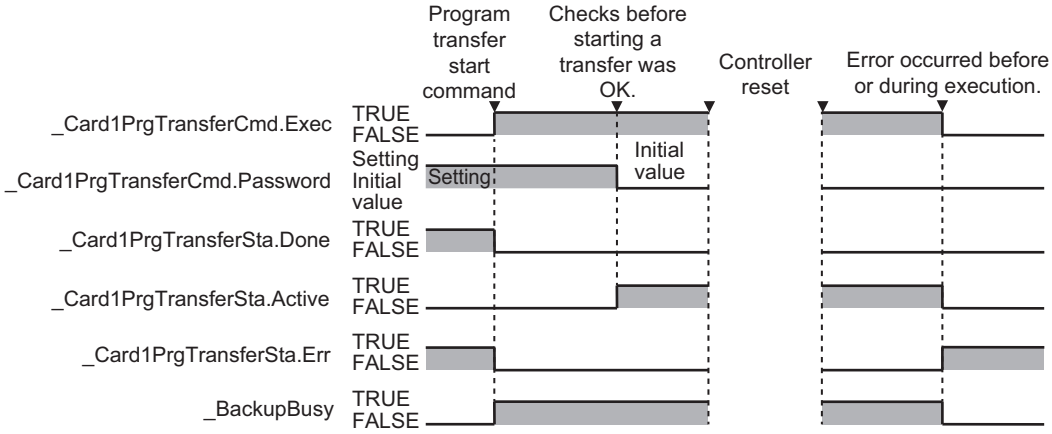
● Normal Operation



● Operation When the Program Transfer Cannot Be Started Because Another Backup Function Is in Progress

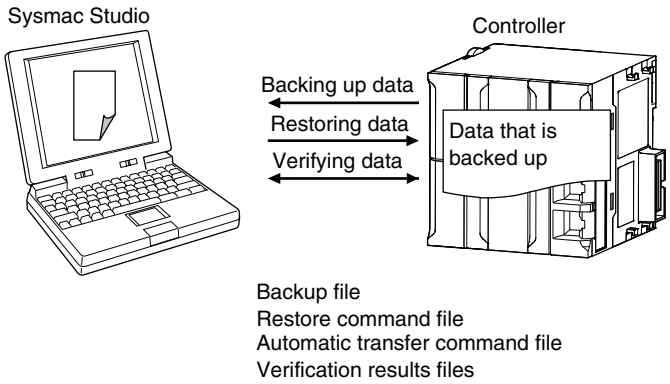


● Operation When the Program Transfer Fails After a Normal Start of Program Transfer



9-6 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 specifies the data groups to transfer when automatically transferring data from an SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer.	Created.	Nothing is done.	Nothing is done.

This function can be executed in the following operating modes:

Processing	Applicable operating modes
Backing up data	RUN mode and PROGRAM mode
Restoring data	PROGRAM mode
Verifying data	RUN mode and PROGRAM mode

Additional Information

You can change the operating mode of the CPU Unit while a backup or verification operation is in progress.
However, an error will occur if the backup or verification cannot be processed normally due to faulty memory in the CPU Unit, or some other failure.

9-6-1 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 the NX Units connected to the CPU Unit, all the X Bus Units that are connected, all the EtherCAT slaves that are connected, and all of the CJ-series Units that are connected.
- The backing up conditions for data groups are given in the following table.

Data group	Backing up condition
User program and settings	The CPU Unit must be selected.
IP address of the built-in EtherNet/IP port*1	The CPU Unit must be selected.
Present values of variables	The CPU Unit must be selected.
Present values of memory used for CJ-series Units	The CPU Unit must be selected.
Event logs	The CPU Unit must be selected.
Units and slaves settings	The **NX Units on the CPU Rack***2, *CJ-series Units**5, and *EtherCAT slaves* must be selected.
X Bus Unit Settings	The *X Bus Unit* must be selected.*3
IP address of an EtherNet/IP port on an X Bus Unit	The *IP address of an EtherNet/IP port on an X Bus Unit* must be selected.*4
Absolute encoder home offsets	The CPU Unit must be selected.

*1. A CPU Unit with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required.
*2. You can select NX Units on the CPU Unit only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*3. You can select the X Bus Unit only for the NX502 CPU Units.
*4. You can select IP address of an EtherNet/IP port on an X Bus Unit only for the NX502 CPU Units.
*5. You can select CJ-series Units only for NJ-series CPU Units.

- 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.

9-6-2 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.

Precautions for Correct Use

For CPU Units with unit version 1.40 or later, the Controller reset is required in any of the following cases. Transfer the project according to the message on the Sysmac Studio.

- When an attempt is made to transfer a project with project unit version 1.40 or later to a CPU Unit where a project with project unit version earlier than 1.40 is stored.
- When an attempt is made to transfer a project with project unit version earlier than 1.40 to a CPU Unit where a project with project unit version 1.40 or later is stored.
- After the Clear All Memory operation is performed on a CPU Unit where a project with project unit version 1.40 or later is stored, without executing the Controller reset, an attempt is made to transfer a project with project unit version earlier than 1.40.

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.
The 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 with Retain attribute* must be selected.
Present values of memory used for CJ-series Units	The *present values of memory for CJ-series Units with Retain attribute* must be selected.*2
Units and slaves settings	The *NX Units on the CPU Unit*[*3], *CJ-series Units*[*2], and *EtherCAT slaves* must be selected.
X Bus Unit Settings	The *X Bus Unit* must be selected.*4
IP address of an EtherNet/IP port on an X Bus Unit	The *IP address of an EtherNet/IP port on an X Bus Unit* must be selected.*5
Absolute encoder home offsets	The *absolute encoder home offsets* must be selected.

*1. A CPU Unit with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required.
*2. You can select the present values of variables with a Retain attribute in memory used for CJ-series Units only for the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*3. You can select NX Units on the CPU Unit only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*4. You can select the X Bus Unit only for the NX502 CPU Units.
*5. You can select IP address of an EtherNet/IP port on an X Bus Unit only for the NX502 CPU Units.

- If an error occurs in the checks that are performed before starting to restore the data, the previous data will be retained in the Controller.
- If the power supply to the Controller is interrupted while the data is being restored, a *User Program/ Controller Configurations and Setup Transfer Error (a major fault level Controller error)* will occur. If that occurs, the data in the Controller is not dependable. Use one of the following methods to clear the error.
  - Perform the restore operation again.
  - Clear all of memory and then download the project from the Sysmac Studio.
- If the present values of variables that are set to be retained (with the Retain attribute) are not set to be restored, the previous present values of those variables will be retained. However, the values of any variables that do not meet the retain conditions are initialized. These are the retain conditions for the variable:
  - The variable name, data type name, and data type size must be the same before and after restoring the data.
- The restore operation is possible even if the Option Board configuration in the backup file do not match the actual configuration where data is restored. However, the Option Board does not operate. Refer to the *NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual (Cat. No. W579)* for details.
- For the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, memory for CJ-series Units is generated by the settings in the Memory Settings for CJ-series CPU Units in the backup file.
- If the present values of memory for CJ-series Units are not set to be restored in the NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit, the previous present values in the DM, EM and Holding Areas will be retained. However, when the DM, EM and Holding Areas are newly generated or the area is expanded, the values in those areas will be the initial values.
- Cycle the power supply to all of the EtherCAT slaves after you restore data.
- If Sysmac Studio version 1.50 or later is used and user authentication function is enabled in the backup file to be used for restoration, the Controller will be automatically reset when restoration is completed.

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.

9-6-3 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 select them all at once, all of these data will be referenced.

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 NX Units on the CPU Unit*2, CJ-series Units*5, and EtherCAT slaves must be selected.
X Bus Unit Settings	The *X Bus Unit* must be selected.*3
IP address of an EtherNet/IP port on an X Bus Unit	The *IP address of an EtherNet/IP port on an X Bus Unit* must be selected.*4

*1. A CPU Unit with unit version 1.14 or later and Sysmac Studio version 1.18 or higher are required.
*2. You can select NX Units on the CPU Unit only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*3. You can select the X Bus Unit only for the NX502 CPU Units.
*4. You can select IP address of an EtherNet/IP port on an X Bus Unit only for the NX502 CPU Units.
*5. You can select CJ-series Units only for NJ-series CPU Units.

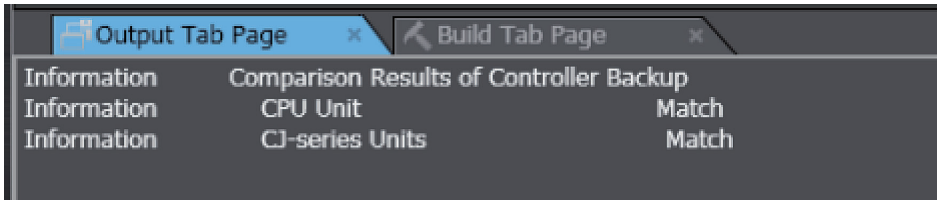
- The results of the verification are displayed in the dialog box on the Sysmac Studio.
- The value of the *_BackupBusy* (Backup Function Busy Flag) system-defined variable will be TRUE during the backup operation.

Procedure

1 Select **Backup** − **Compare with Backup File** from the **Tools** Menu on the Sysmac Studio.

2 Specify the folder that contains the backup file.

3 Click the **Execute** Button on the Comparison Confirmation Dialog Box.
The data is compared and the verification results files are created in the folder that contains the backup file. The comparison results are displayed in the Output Tab Page.



9-7 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:

(○: Applicable, x: Not applicable)

Function		Data group						
		User program and settings	IP address of built-in EtherNet/IP port*2	Present values of variables	Present values of memory used for CJ-series Units*1	Units and slaves settings	Absolute encoder home offsets	X Bus Unit Settings*6
Importing and exporting Sysmac Studio backup file data	Exporting backup file data	○*3	○	×	×	×	×	×
	Importing backup file data	○*4*5	○	×	×	○	×	○

*1. The target data for present values of memory used for CJ-series Units is processed only for the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

*2. With a combination of the CPU Unit 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.

*3. The following data is not processed.
- The built-in EtherNet/IP port name in the Controller name
- The built-in EtherNet/IP tag data link settings in the Controller Setup
- Words allocated to CPU Bus Units in the Unit Configuration and Unit Settings
- Operation authority verification
- User authentication
- Data trace settings

*4. The following data is not processed.
- The built-in EtherNet/IP port name in the Controller name
- Words allocated to CPU Bus Units in the Unit Configuration and Unit Settings
- Operation authority verification
- User authentication
- Data trace settings

*5. The following data is not processed.
When Sysmac Studio lower than version 1.15 is used
- The built-in EtherNet/IP tag data link settings in the Controller Setup

*6. The target data for X Bus Unit Settings is processed only for the NX502 CPU Units.

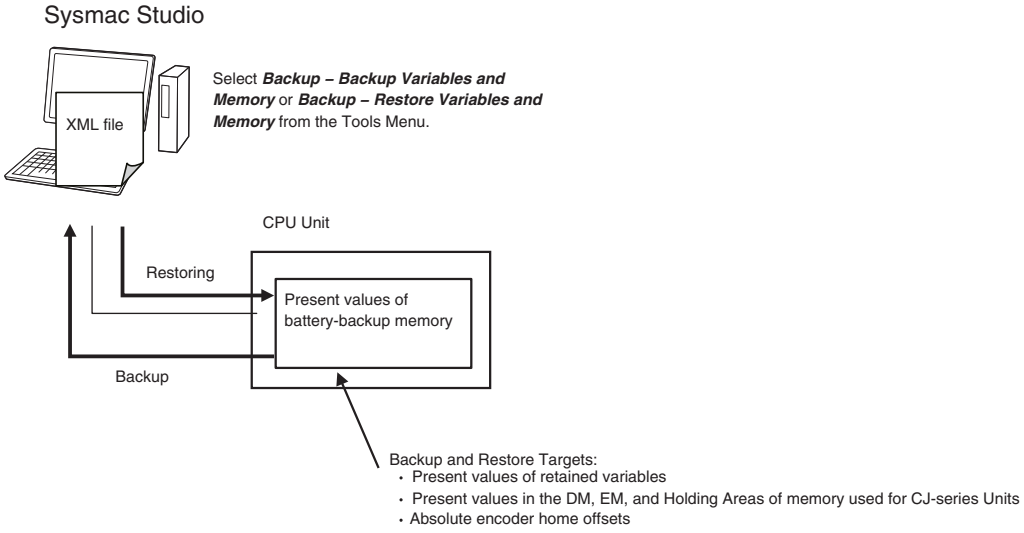
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for more information on these functions.

Precautions for Correct Use

The data related to OPC UA server function is included in the *user program and setting* data groups. However, the data related to OPC UA server function is stored in the CPU Unit, so the data is not subject to the importing and exporting Sysmac Studio backup file data.

9-8 Sysmac Studio Variable and Memory Backup Functions

You can back up the present values of the battery-backup memory in the CPU Unit to an XML file on your computer or restore the battery-backup memory from a previously saved backup file.
This section describes the applicable data, operating procedures, and CPU Unit model compatibility for the Sysmac Studio variable and memory backup functions.



9-8-1 Applicable Data for Sysmac Studio Variable and Memory Backup Functions

The Sysmac Studio variable and memory backup functions cover the following data.

- Present values of variables with a Retain attribute
- Present values in the DM, EM, and Holding Areas of memory used for CJ-series Units
- Absolute encoder home offsets

Note You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

Version Information

With a CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher, you can select specific variables to back up or restore the present values of variables with a Retain attribute.

9-8-2 Using Sysmac Studio Variable and Memory Backup Functions

The Sysmac Studio procedure is as follows:
Place the Sysmac Studio online with the CPU Unit, and select either **Backup** − **Backup Variables and Memory** or **Backup** − **Restore Variables and Memory** from the Tools Menu.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details.

9-8-3 Compatibility between CPU Unit Models

With the Sysmac Studio variable and memory backup functions, you can restore the data even if the models of the CPU Units for backing up and restoring data are different.

Additional Information

Database Connection CPU Units, SECS/GEM CPU Units, NJ Robotics CPU Units, Robot Integrated CPU Units, and NC Integrated Controller are not compatible.
Refer to the relevant manuals for specific Units for details on the compatibility of these CPU Units.

Version Information

The models of the CPU Units for backing up and restoring data can be different only when using a CPU Unit with version 1.04 or later and Sysmac Studio version 1.05 or higher. The compatibility for all other cases is given in the following table.

(○: Compatible, ×: Not compatible.)

CPU Unit model where data was backed up	CPU Unit model to restore to	
	NJ501-1500 NJ501-1400 NJ501-1300	NJ301-1200 NJ301-1100
NJ501-1500, NJ501-1400, or NJ501-1300	○	×
NJ301-1200 or NJ301-1100	×	○

The following precautions are required for the data that is being backed up and restored.

Present Values of Variables with a Retain Attribute

There are no precautions for the present values of variables with a Retain attribute. You can restore the data with no problems regardless of the models of the backup and restore CPU Units.

Present Values in the DM, EM, and Holding Areas of Memory Used for CJ-series Units

For memory used for CJ-series Units, if the area data saved in the backup file and area data in the actual CPU Unit to restore to are different, only the duplicated area data is restored.

The following precautions are necessary for the present values in the DM, EM, and Holding Areas of memory used for CJ-series Units if the models of the backup and restore CPU Units are different. An example of the NJ-series CPU Unit is given in the following table.

CPU Unit model where data was backed up	CPU Unit model to restore to	Precaution
NJ501-1300, NJ501-1400, or NJ501-1500	NJ301-1200, NJ301-1100, NJ101-1000, or NJ101-9000	For EM Area data, only data for banks E0 to E3 in the backup file is restored. The data for banks E4 to E18 in the backup file is ignored.

CPU Unit model where data was backed up	CPU Unit model to restore to	Precaution
NJ301-1200, NJ301-1100, NJ101-1000, or NJ101-9000	NJ501-1300, NJ501-1400, or NJ501-1500	For EM Area data, only data for banks E0 to E3 in the backup file is restored. Banks E4 to E18 for the CJ-series Units retain their previous values.

The following provides examples when the area data saved in the backup file and area data in the actual CPU Unit to restore to are different.

● If Area Data in the Backup File Is Larger Than Area Data in the Restore CPU Unit

Size of DM Area data when backed up = 8 words		Size of DM Area data in the restore CPU Unit = 4 words	
Address	Value	Address	Value
DM0	0x0001	DM0	0x0001
DM1	0x0002	DM1	0x0002
DM2	0x0003	DM2	0x0003
DM3	0x0004	DM3	0x0004
DM4	0x0005	---	---
DM5	0x0006	---	---
DM6	0x0007	---	---
DM7	0x0008	---	---

● If Area Data in the Backup File Is Less Than Area Data in the Restore CPU Unit

Size of DM Area data when backed up = 8 words		Size of DM Area data in the restore CPU Unit = 4 words	
Address	Value	Address	Value
DM0	0x0001	DM0	0x0001
DM1	0x0002	DM1	0x0002
DM2	0x0003	DM2	0x0003
DM3	0x0004	DM3	0x0004
---	---	DM4	No change
---	---	DM5	No change
---	---	DM6	No change
---	---	DM7	No change

Absolute Encoder Home Offsets

The following precautions are necessary for the absolute encoder home offsets if the models of the backup and restore CPU Units are different. An example of the NJ-series CPU Unit is given in the following table.

CPU Unit model where data was backed up	CPU Unit model to restore to	Precaution
NJ501-1300, NJ501-1400, or NJ501-1500	NJ301-1200, NJ301-1100, or NJ101-1000	Regardless of the number of enabled axes in the backup CPU Unit, the data for all axis in the backup file is restored in order for the number of enabled axes in the restore CPU Unit. Any remaining data in the backup file is ignored.
NJ301-1200, NJ301-1100, or NJ101-1000	NJ501-1300, NJ501-1400, or NJ501-1500	Regardless of the number of enabled axes in the backup CPU Unit, the data for all axis in the backup file is restored in order for the number of enabled axes in the restore CPU Unit. If the number of enabled axes in the restore CPU Unit exceeds the number of enabled axes for which there is data in the backup file, the remaining data in the restore CPU Unit retains the previous values.

9-9 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 *9-10 Backup Functions When EtherCAT Slave Terminals Are Connected* on page 9-62.

9-9-1 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 Start-up, 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 setting parameter settings.

9-9-2 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*5	Not possible.*2	Possible.*3
Network configuration mismatch with configuration information*6		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.*7	Possible.*7	Possible.*7
	No disabled slaves in actual configuration	Possible.	Possible.	Possible.

Controller status		Execution		
		Backing up data	Restoring data	Verifying data
Slave disconnected for “Disconnect” designation in network configuration	Disconnected slaves in actual configuration	Not possible.*1	Possible. Data for disconnected slaves is also restored.	Possible. Data for disconnected slaves is also restored.
	No disconnected slaves in actual configuration	Not possible.*1	Not possible.*2	Possible.*3
Slave State Transition Failed*8		Not possible.*1	Not possible.*2	Possible.*3

*1. EtherCAT Slave Backup Failed events are recorded in the event log.
*2. EtherCAT Slave Restore Operation Failed events are recorded in the event log.
*3. The verification results will show differences.
*4. For project unit version earlier than 1.40, 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”, Link OFF Error, and EtherCAT Frame Not Received.
For project unit version 1.40 or later, this refers to the following errors: Incorrect Wiring Detected, Duplicate Slave Node Address, Network Configuration Information Error, Network Configuration Verification Error (Unnecessary Slave Connected), Network Configuration Verification Error (Mismatched Slave), Network Configuration Verification Error (Slave Unconnected), Network Configuration Verification Error (Incorrect Ring Wiring), Link OFF Error, and EtherCAT Frame Not Received.
*5. The backup operation can be executed when the Network Configuration Information Error occurs.
*6. For project unit version earlier than 1.40, 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).
For project unit version 1.40 or later, 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), network configuration information mismatch with actual network configuration (incorrect connection ports for slaves on branched network are treated as a mismatch) and Ring Disconnection Detected.
*7. For a CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher, data for disabled slaves is also covered by the backup functions. Data for disabled slaves is not backed up for other versions.
*8. For project unit version earlier than 1.40, this is a status in which the Slave Initialization Error occurs.

9-9-3 Conditions for Restoring EtherCAT Slave Data

The following conditions must be met before you 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.

9-9-4 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-series EtherCAT Coupler Unit NX-ECC	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.
AC Servo Drives R88D-1SN□□□-ECT R88D-1SAN□□□-ECT	*1
AC Rotary Servo Drives R88D-KN□□□-ECT	*1*2
AC Linear Servo Drives R88D-KN□□□-ECT-L	*1*2
Inverters 3G3AX-MX2-ECT 3G3AX-RX-ECT	• When the unit version of the CPU Unit is 1.11 or later, the inverter parameters are not included in the restore target. An EtherCAT Slave Restore Operation Failed event (event code 10300000 hex) will not occur for Inverters. • Refer to *Procedure to Write Parameters for a 3G3AX-MX2-ECT or 3G3AX-RX-ECT Inverter* on page 9-61 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. • When the unit version of the CPU Unit is 1.10 or earlier, data is sometimes not restored due to Inverter restrictions. If an EtherCAT Slave Restore Operation Failed event (event code 10300000 hex) occurs when you try to restore the data, refer to *Procedure to Write Parameters for a 3G3AX-MX2-ECT or 3G3AX-RX-ECT Inverter* on page 9-61 to write the parameters from the Sysmac Studio to the Inverter. Note that even if the restore operation for the Inverter fails, all other data are restored including settings of Units and slaves, user program settings, and present values of variables.
Vision Sensors FH-3□□□□ FH-1□□□□	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.

EtherCAT slaves	Precautions
Vision Sensors FQ-M□□□□-ECT FQ-M□□□□-M-ECT	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 *FQ-M-series Specialized Vision Sensor for Positioning User's Manual* (Cat. No. Z314) for details.
Vision Sensors FZM1-□□□-ECT	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.
Digital I/O Terminals GX-□D16□□ GX-□D32□8 GX-OC1601	*1
Analog I/O Terminals GX-AD0471 GX-DA0271	*1
Encoder Input Terminals GX-EC0211 GX-EC0241	---
EtherCAT Junction Slaves GX-JC0□	There is no internal data that needs to be backed up.
Confocal Fiber Type Displacement Sensors ZW-CE1□	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.
Digital Sensors E3NW-ECT E3X-ECT	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 a 3G3AX-MX2-ECT or 3G3AX-RX-ECT Inverter

When the unit version of the CPU Unit is 1.11 or later, the inverter parameters are not included in the restore target.
When the unit version of the CPU Unit is 1.10 or earlier, an *EtherCAT Slave Restore Operation Failed event* (event code 10300000 hex) will occur if you restore data while a 3G3AX-MX2-ECT or 3G3AX-RX-ECT Inverter is connected.
Use the follow procedure from the Sysmac Studio to write the backup parameters to the Inverter. Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details.

1 Import the Inverter parameters from the backup file with the backup file import function of the Sysmac Studio.
Display the Inverter parameters on the Inverter Parameters Tab Page for the Controller Configurations and Setup of the Sysmac Studio.

2 Confirm that the model number of the Inverter in the parameters that you imported agrees with the model number of the Inverter that is actually connected.

3 Download the parameters to the Inverter using the “To Drive” menu on the Inverter Parameters Tab Page for the Controller Configurations and Setup of the Sysmac Studio.

Precautions for Correct Use

If you use the Inverter Mode Selection parameter (parameter number b171) in a 3G3AX-MX2-ECT Inverter, change the Inverter to the mode that was used when the backup data was created before you write the parameters. After you change the mode setting, you must initialize the Inverter to enable the change.

Additional Information

When the unit version of the CPU Unit is 1.10 or earlier, even if the restore operation for the Inverter fails, all other data are restored including settings of Units and slaves, user program settings, and present values of variables.

9-10 Backup Functions When EtherCAT Slave Terminals Are Connected

For EtherCAT Slave Terminals, you can use the SD Memory Card backup functions, the Sysmac Studio Controller backup functions, and Sysmac Studio backup import function.
This section provides information on the data that is backed up, backup support according to Controller status, and restore conditions when EtherCAT Slave Terminals are connected.

Precautions for Correct Use

You cannot back up, restore, or compare data for Safety Control Units on EtherCAT Slave Terminals. Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for information on importing and exporting settings for a Safety Control Unit.

Version Information

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

9-10-1 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.

(○: Applicable, x: Not applicable)

Unit	Data	Backup	Restore	Compare
EtherCAT Coupler Unit	Configuration information*1	○	○	○
	Unit operation settings	○	○	○
NX Units	Configuration information*1	○	○	○
	Unit operation settings	○	○	○
	Unit application data*2	○	○	○

*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.

9-10-2 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 information 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.

9-10-3 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 up 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.

9-11 Backup Functions When NX Units Are Connected

For NX Units on the NX502 CPU Unit, NX102 CPU Unit, and NX1P2 CPU Unit, you can use the SD Memory Card backup functions and the Sysmac Studio Controller backup functions.
This section provides information on the data that is backed up, backup support according to Controller status, and restore conditions when NX Units are connected to the NX1P2 CPU Unit.

Precautions for Correct Use

You can mount an NX-SL□□□□ Safety Control Unit on the NX502 CPU Unit and NX102 CPU Unit. However, you cannot back up, restore, or compare data for the Safety Control Unit. Refer to the *NX-series Safety Control Unit User's Manual (Cat. No. Z930-E1-12 or later)* for information on importing and exporting settings and safety unit restore settings for a Safety Control Unit.

9-11-1 Backing Up Data in NX Units on the CPU Unit

The data that is backed up for NX Units on the CPU Unit is given in the following table.

Unit	Data	Backup	Restore	Compare
NX Units	Configuration information*1	Possible	Possible	Possible
	Unit operation settings	Possible	Possible	Possible
	Unit application data*2	Possible	Possible	Possible

*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.

9-11-2 Backup Support Depending on the Controller Status

The following table shows when backup, restore, and compare operations can be performed for NX Units based on the Controller status.

Controller status	Execution		
	Backing up data	Restoring data	Verifying data
Automatic creation of the Unit configuration information	Possible*1	Possible*2	Possible
Watchdog time error in NX Unit	Not possible*3	Not possible*4	Possible*5
During NX Bus Controller Error	Not possible*3	Not possible*4	Possible*5
During Unit Configuration Verification Error	Not possible*3	Possible	Possible*5
The Unit configuration information 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. An NX Unit Backup Failed event is recorded in the event log.
*4. An NX Unit Restore Operation Failed event is recorded in the event log.
*5. The verification results will show differences.

9-11-3 Conditions for Restoring NX Unit Data on the CPU Unit

The following conditions must be met before you restore the backup data to the NX Units on the CPU Unit.

- The backup files must contain the data of the relevant CPU Unit and the data of the NX Units on the relevant CPU Unit.
- The original Unit configuration in the backup must match the actual Unit configuration where data is being restored.
- 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 **CPU Racks** in the backup file is set to *Setting* = *Actual device*.
- 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.

9-12 Backup Functions When X Bus Units Are Connected

For X Bus Units on the NX502 CPU Unit, you can use the SD Memory Card backup functions and the Sysmac Studio Controller backup functions.
This section provides information on data that is backed up, backup support according to Controller status, and restore conditions.

9-12-1 Backing Up Data in X Bus Units on the CPU Unit

The data that is backed up for X Bus Units on the NX502 CPU Unit is given in the following table.

(○: Possible, ×: Not possible)

Unit	Data	Backup	Restore	Compare
X Bus Units	X Bus Unit settings*1	○	○	○
	IP address of an EtherNet/IP port on an X Bus Unit	○	○	○

*1. Settings for the X Bus Unit itself. For example, for NX-series EtherNet/IP Units, the settings such as connection settings for tags or tag sets and EtherNet/IP settings are included.

9-12-2 Backup Support Depending on the Controller Status

The following table shows when backup, restore, and compare operations can be performed for X Bus Units based on the Controller status.

Controller status	Execution		
	Backing up data	Restoring data	Verifying data
The Unit configuration does not agree with the configuration information.	Not possible.*1	Possible.*2	Possible.*3
The saved backup data is incorrect.	Possible.	Not possible.*4	Possible. *5

*1. An *X Bus Unit Backup Failed* event is recorded in the event log.
*2. If there is a configuration mismatch with the configuration information in the backup file to be restored, an *X Bus Unit Restore Operation Failed* event is recorded in the event log.
*3. If there is a configuration mismatch with the configuration information in the backup file to be verified, the verification results will show differences.
*4. An *X Bus Unit Restore Operation Failed* event is recorded in the event log.
*5. The verification results will show differences.

9-12-3 Conditions for Restoring X Bus Unit Data on the CPU Unit

The following conditions must be met before you restore the backup data to the X Bus Units on the NX502 CPU Unit.

- The backup files must contain the data of the relevant CPU Unit and the data of the X Bus Units on the relevant CPU Unit.

9-13 Backup Functions When CJ-series Units Are Connected

Data in CJ-series Units is covered by the SD Memory Card backup functions and Sysmac Studio Controller backup functions.
This section provides precautions for connected CJ-series Units for the data that is backed up, backup support according to Controller status, and restore conditions.

Precautions for Correct Use

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

9-13-1 Backed Up CJ-series Unit Data

The present values in memory used for CJ-series Units and the parameters in the CJ-series Units are backed up. Some of this data is in the CJ-series Unit and some are in the CPU Unit.
You do not need to be aware of where the data is located because the backup, restoration, and verification operations will automatically process this data.

9-13-2 Backup Support Depending on the Controller Status

The following table shows when backup, restore, and verify operations can be performed for CJ-series Units based on the Controller status.

Controller status	Execution		
	Backing up data	Restoring data	Verifying data
I/O Bus Check Error End Cover Missing Incorrect Unit/Expansion Rack Connection Duplicate Unit Number Error Too Many I/O Points I/O Setting Check Error	Not possible.*1	Not possible.*2	Possible.*3
Restarting the CJ-series Unit	Not possible.*1	Possible.	Possible.*3

*1. CJ-series Unit Backup Failed events are recorded in the event log.
*2. CJ-series Unit Restore Operation Failed events are recorded in the event log.
*3. The verification results will show differences.

9-13-3 Conditions for Restoring CJ-series Unit Data

The following conditions must be met before you restore the backup data to the CJ-series Units.

- The backup files must contain the CJ-series Unit data.
- The Unit configuration in the backup file must match the actual Unit configuration where data is being restored.
- Each CJ-series Unit must meet the conditions for that Unit. (Refer to the manuals for the CJ-series Units for the specific conditions for each Unit.)

Precautions for Correct Use

If you restore data using the SD Memory Card backup functions or the Sysmac Studio Controller backup functions while CJ-series Units are connected, a CPU Unit Service Monitor Error will occur. This means that servicing the CJ-series Units from the CPU Unit was not completed within a specific amount of time. However, this is the result of the time that is required to restore the data and it does not indicate an error. The following will occur at this time.

- For communications-related CJ-series Units, the MS indicator flashes red.
- For CJ-series Units with seven-segment indicator, the indicator displays “HE”.
- For CJ-series Units that have an ERH indicator, the ERH indicator lights.
- An event code of 00000002 hex is recorded in the Controller event logs to indicate a CPU Unit Service Monitor Error or Refresh Timeout event.
- If a CJ-series CJ1W-CT021 High-speed Counter Unit is connected, an event code of 68010000 hex is recorded in the Controller event logs to indicate a Unit Error. The attached information will be 0002 hex.

9-14 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.

9-14-1 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 an SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer. These files are created when data is backed up.

● Automatic Transfer Command File

This file specifies the data groups to transfer when automatically transferring data from an SD Memory Card. You can edit this file with a text editor on a computer to specify the data groups to transfer. These files are created when data is backed up.

● Verification Results Files

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit.
There are four different verification results files, as described below. These files are generated 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.

Verification results files	Description
NX Unit verification results file*1	This file contains the verification results for each NX Unit on the CPU Unit. It is generated when the Unit and slave settings are set to be restored in the restore command file and the backup file contains settings of the NX Unit on the CPU Unit.
X Bus Unit verification results file*2	This file contains the verification results for each X Bus Unit. It is generated when the Unit and slave settings are set to be restored in the restore command file and the backup file contains settings of the X Bus Unit.
CJ-series Unit verification results file*3	This file contains the verification results for each CJ-series Unit. It is created when the Unit and slave settings are set to be restored in the restore command file and the CJ-series Unit settings are contained in the backup file.

*1. You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
*2. You can use X Bus Units only on the NX502 CPU Units.
*3. You can use CJ-series Units only on the NJ-series CPU Units.

9-14-2 Specifications of a Backup File

This section describes the file name, creation timing, and created directory for a backup file.

File Name

A different backup file name is given depending on the CPU Unit series.

File	CPU Units	File name
Backup file	NX-series CPU Units	NXBackup.dat
	NJ-series CPU Units	NJBackup.dat

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	CPU Unit Front-panel DIP Switch	When backup is executed	Root directory on the SD Memory Card
	System-defined variables	When backup is executed	Directory on the SD Memory Card that you specified with the system-defined variable
	SD Memory Card Window in Sysmac Studio	When backup is executed	Directory on the SD Memory Card that you specified with the Sysmac Studio
	Special instruction*1	When backup is executed	The directory on the SD Memory Card that you specified for the input variable of the BackupToMemoryCard instruction
Sysmac Studio Controller backups	Sysmac Studio Controller Backup Dialog Box	When backup is executed	Directory in the computer that you specified with the Sysmac Studio
Importing and exporting Sysmac Studio backup file data	Sysmac Studio Backup File Export Dialog Box	When data is exported	Directory in the computer that you specified with the Sysmac Studio

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

9-14-3 Specifications of a Restore Command File

This section describes the file name, creation timing, created directory, and data group specification method for a restore command file.

File Name

File	File name
Restore command file	RestoreCommand.ini

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	CPU Unit Front-panel DIP Switch	When backup is executed	Same directory as backup file
	System-defined variables	When backup is executed	Same directory as backup file
	SD Memory Card Window in Sysmac Studio	When backup is executed	Same directory as backup file
Sysmac Studio Controller backups	Sysmac Studio Controller Backup Dialog Box	When backup is executed	Same directory as backup file
Importing and exporting Sysmac Studio backup file data	Sysmac Studio Backup File Export Dialog Box	When data is exported	Same directory as backup file

Specifying the Data Groups to Restore

The restore command file allows you to specify the data groups to restore.
You can change the data group specifications by editing the file with a text editor on a computer.
For example, if you change *Variable=yes* on line 8 in the *File contents* that are shown in the following table to *Variable=no*, the *present values of variables* will not be restored.

File contents (defaults when the file is created)	Description
[Restore]	
; --- User Program and Configuration. --- ; Always select "yes". UserProgram=yes	User program and settings This data group is always restored. Always select “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.

File contents (defaults when the file is created)	Description
; --- Present values of memory used for CJ-series Units (Holding, EM, and DM Area data). --- ; "yes":will be restored, "no":will not be restored Memory=yes	Present values of memory used for CJ-series Units (only addresses that are set to be retained with the Retain attribute) yes/no: Restore/Do not restore.
:---Unit/Slave Parameters.--- ; "yes";will be restored."no";will not be restored UnitConfig=yes	Units and slaves settings yes/no: Restore/Do not restore.
; --- Absolute encoder home offset. --- ; "yes":will be restored, "no":will not be restored AbsEncoder=yes	Absolute encoder home offsets yes/no: Restore/Do not restore.
; --- X Bus Unit Parameters. --- ; "yes":will be restored, "no":will not be restored. XBusUnitConfig=yes	X Bus Unit settings yes/no: Restore/Do not restore.
; --- IP Address of X Bus Unit's Ethernet Port Settings. --- ; “yes”:will be restored, “no”:will not be restored. XBusUnitIPAdr=no	IP address of an EtherNet/IP port on an X Bus Unit 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.

Note 2. The restore command file lists the restorable data groups that were in the backup file when the backup file was created.

Note 3. Only single-byte alphanumeric characters are used. The text is not case sensitive.

Note 4. An entry of IP Address of Built-in EtherNet/IP Port Settings is not created if the backup is performed in the CPU Unit with unit version 1.13 or earlier.
In the CPU Unit 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 9-73 for compatibility between the restore command file with unit version 1.13 or earlier and the restore command file with unit version 1.14 or later.

Note 5. If an entry of IP Address of an EtherNet/IP port on an X Bus Unit for which the restore command file is not created is used, the operation is performed as “XBusUnitIPAdr=yes”.

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 1.13 or earlier and the restore command file with unit version 1.14 or later.

Unit version of CPU Unit that creates the restore command file and backup file	Unit version of CPU Unit where data is being restored	
	Version 1.13 or earlier	Version 1.14 or later
Version 1.13 or earlier	Restorable. IP Address of Built-in EtherNET/IP Port Settings is restored.	Restorable. IP Address of Built-in EtherNET/IP Port Settings is restored.

Unit version of CPU Unit that creates the restore command file and backup file	Unit version of CPU Unit where data is being restored	
	Version 1.13 or earlier	Version 1.14 or later
Version 1.14 or later	Not restorable. In the CPU Unit with unit version 1.13 or earlier, 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”.

9-14-4 Specifications of an Automatic Transfer Command File

This section describes the file name, creation timing, created directory, and data group specification method for an automatic transfer command file.

File Name

File	File name
Automatic transfer command file	AutoloadCommand.ini

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	CPU Unit front-panel DIP switch	When backup is executed	Same directory as backup file
	System-defined variables	When backup is executed	Same directory as backup file
	SD Memory Card Window in Sysmac Studio	When backup is executed	Same directory as backup file
Sysmac Studio Controller backups	Sysmac Studio Controller Backup Dialog Box	When backup is executed	Same directory as backup file
Importing and exporting Sysmac Studio backup file data	Sysmac Studio Backup File Export Dialog Box	When data is exported	Same directory as backup file

Specifying the Data Groups to Automatically Transfer

The automatic transfer command file allows you to specify the data groups to transfer automatically. You can change the data group specifications by editing the file with a text editor on a computer.
For example, if you change “Variable=yes” on line 8 in the file contents that are shown in the following table to “Variable=no,” the present values of variables will not be automatically transferred.

File contents (defaults when the file is created)	Description
[Autoload]	

File contents (defaults when the file is created)	Description
; --- User Program and Configuration. --- ; Always select "yes". UserProgram=yes	User program and settings This data group is always transferred. Always select “yes”.
; --- IP Address of Built-in EtherNet/IP Port Settings. --- ; "yes":will be transferred, "no":will not be transferred IPAdr=yes	IP address of built-in EtherNet/IP port yes/no: Transfer/Do not transfer.
; --- Present values of variables (Retained variables only). --- ; "yes":will be transferred, "no":will not be transferred Variable=yes	Present values of variables (only variables that are set to be retained with the Retain attribute) yes/no: Transfer/Do not transfer.
; --- Present values of memory used for CJ-series Units (Holding, EM, and DM Area data). --- ; "yes":will be transferred, "no":will not be transferred	Present values of memory used for CJ-series Units (only addresses that are set to be retained with the Retain attribute) yes/no: Transfer/Do not transfer.
; --- X Bus Unit Parameters. --- ; "yes":will be transferred, "no":will not be transferred. XBusUnitConfig=yes	X Bus Unit settings yes/no: Transfer/Do not transfer.
; --- IP Address of X Bus Unit's Ethernet Port Settings. --- ; “yes”:will be transferred, “no”:will not be transferred. XBusUnitIPAdr=yes	IP address of an EtherNet/IP port on an X Bus Unit yes/no: Transfer/Do not transfer.

Note 1. The default file contents when the automatic transfer command file is created are given above. All of the data groups that are listed in the file are set to be automatically transferred.

Note 2. The automatic transfer command file lists the transferable data groups that were in the backup file when the backup file was created.

Note 3. Only single-byte alphanumeric characters are used. The text is not case sensitive.

Note 4. An entry of IP Address of Built-in EtherNet/IP Port Settings is not created if the backup is performed in the CPU Unit with unit version 1.13 or earlier.
In the CPU Unit with unit version 1.14 or later, if an entry of IP Address of Built-in EtherNet/IP Port Settings for which the automatic transfer file is not created is used, the operation is performed as “IPAdr=yes”.
Refer to *Compatibility between Automatic Transfer Files* on page 9-75 for compatibility between the automatic transfer file with unit version 1.13 or earlier and the automatic transfer file with unit version 1.14 or later.

Note 5. If an entry of IP Address of an EtherNet/IP port on an X Bus Unit for which the restore command file is not created is used, the operation is performed as “XBusUnitIPAdr=yes”.

Precautions for Correct Use

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

Compatibility between Automatic Transfer Files

The following table shows the compatibility between the automatic transfer file with unit version 1.13 or earlier and the automatic transfer file with unit version 1.14 or later.

Unit version of CPU Unit that creates the automatic transfer file and backup file	Unit version of CPU Unit where data is being automatically transferred	
	Version 1.13 or earlier	Version 1.14 or later
Version 1.13 or earlier	Automatic transfer is possible. IP Address of Built-in EtherNET/IP Port Settings is automatically transferred.	Automatic transfer is possible. IP Address of Built-in EtherNET/IP Port Settings is automatically transferred.
Version 1.14 or later	Automatic transfer is not possible. In the CPU Unit with unit version 1.13 or earlier, because the entry of IP Address of Built-in EtherNET/IP Port Settings cannot be interpreted, an Error in Starting Automatic Transfer in a major fault level occurs. The error details will be “0104 hex: The contents of the automatic transfer command file are not correct”.	Automatic transfer is possible. IP Address of Built-in EtherNET/IP Port Settings corresponds to “yes/no” of “IPAdr”.

9-14-5 Specifications of a Controller Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for a Controller verification results file.

File Name

File	File name
Controller verification results file	VerifyResult.log

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group.
You can check the verification results in the portion that gives the verification results for each data group.
Result=Matched indicates a data group for which no differences were found. *Result=Not matched* indicates a data group for which differences were found.

In the file shown below, the user program and configuration data matched, and the Units and slave parameters did not match.

File contents	Description
[UserProgram] ; --- User Program and Configuration. --- Result=Matched	 User program and settings Matched: No differences were found, Not matched: Differences were found.
[UnitConfig] ; --- Unit/Slave Parameters. --- Result=Not matched	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.
Note 2. The verification results of IP Address of Built-in EtherNet/IP Port Settings are included in an entry of user program and settings even for the CPU Unit with unit version 1.14 or later.

9-14-6 Specifications of an EtherCAT Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for an EtherCAT verification results file.

File Name

File	File name
EtherCAT verification results file	VerifyResult_ECAT.log

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group.
You can check the verification results in the portion that gives the verification results for each EtherCAT slave.
“Result=Matched” indicates a data group for which no differences were found. “Result=Not matched” indicates a data group for which differences were found.
The following table gives an example of the verification results for the following file contents.

- Matched: EtherCAT slave called Master and EtherCAT Slave Terminal E022
- Not matched: EtherCAT slave E001

File contents	Description
[Verification Results] ; --- EtherCAT Parameters. --- : --- See the VerifyResult_ECAT_NX.log about detail result if NX mark is included in square brackets. [Master] Result=Matched [E001] Result=Not matched Factor=Verification error [E002:NX] Result=Matched	The slaves are indicated with the user-set device names. For an EtherCAT Slave Terminal, ":NX" is added to the end of the device name. *1 The verification results are given as follows: Result=Matched Same Result=Not matched Different

*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.

9-14-7 Specifications of an EtherCAT Slave Terminal Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for an EtherCAT Slave Terminal verification results file.

File Name

File	File name
EtherCAT Slave Terminal verification results file	VerifyResult_ECAT_NX.log

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group.
You can check the verification results in the portion that gives the verification results for 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

9-14-8 Specifications of an NX Unit Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for an NX Unit verification results file.

Precautions for Correct Use

You can use NX Units only on the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.

File Name

File	File name
NX Unit verification results file	VerifyResult_NXUnit.log

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group.
You can check the verification results in the portion that gives the verification results for the 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: NX Unit N1
- Not matched: NX Unit N3
- Not verified: NX Unit N2

File contents	Description
[Verification Results] ; --- NX Parameters. --- [N1:UnitNo.1 NX-AD2203] Result=Matched [N2:UnitNo.2 NX-DA2203] Result=Not verified [N3:UnitNo.3 NX-TS3201] 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 Result=Not verified No verification

9-14-9 Specifications of an X Bus Unit Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for an X Bus Unit verification results file.

Precautions for Correct Use

You can use X Bus Units only on the NX502 CPU Units.

File Name

File	File name
X Bus Unit verification results file	VerifyResult_XbusUnit.log

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group.
You can check the verification results in the portion that gives the verification results for each X Bus Unit.
Result=Matched indicates a data group for which no differences were found. *Result=Not matched* indicates a data group for which differences were found.
The following table gives an example of the verification results for the following file contents.

- Matched: X Bus Unit N1
- Not matched: X Bus Unit N3
- Not verified: X Bus Unit N2

File contents	Description
[Verification Results] ; --- XBus Parameters. --- [N1:UnitNo.1 NX-EIP201] Result=Matched [N2:UnitNo.2 NX-EIP201] Result=Not verified [N3:UnitNo.3 NX-EIP201] 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 4. 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 Result=Not verified No verification

9-14-10 Specifications of a CJ-series Unit Verification Results File

This section describes the file name, creation timing, created directory, and verification results confirmation method for a CJ-series Unit verification results file.

Precautions for Correct Use

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

File Name

File	File name
CJ-series Unit verification results file	VerifyResult_CJUnit.log

File Creation Timing and Created Directories

Function	Procedure	Creation timing	Created directory
SD Memory Card backups	SD Memory Card Window in Sysmac Studio	When verification is executed	Same directory as backup file
	System-defined variables	When verification is executed	Same directory as backup file
	CPU Unit front-panel DIP switch	When verification is executed	Same directory as backup file

Note However, if the SD Memory Card is write-protected, the verification results files will not be created.

How to Check the Verification Results

The verification results files contain the results of comparing the Controller data and the data in a backup file on the SD Memory Card in the CPU Unit for each data group.
You can check the verification results in the portion that gives the verification results for each CJ-series Unit.
“Result=Matched” indicates a data group for which no differences were found. “Result=Not matched” indicates a data group for which differences were found.
In the file shown below, CJ1W-CRM21 (MODE0) and CJ1W-EIP21 matched, and CJ1W-DRM21 and CJ1W-PRM21-DPV1 did not match.

File contents	Description
[Verification Results] ; --- CJ Unit Parameters. --- [Rack0 Slot0: CJ1W-CRM21(MODE0) UnitNo.10] Result=Matched [Rack0 Slot9: CJ1W-DRM21 UnitNo.0] Result=Not matched [Rack1 Slot0: CJ1W-EIP21 UnitNo.10] Result=Matched [Rack1 Slot1: CJ1W-PRM21-DPV1 UnitNo.1] Result=Not matched	The Units are indicated in the following format: Rack{Rack No.}[space]Slot{Slot No}:[space]{Unit model}[space]Unit No.{unit number} Rack No.: Text string of decimal numbers. The value will be between 0 and 3. Slot No.: Text string of decimal numbers. The value will be between 0 and 9. Unit Model: Text string that identifies the Unit model. The Unit model is obtained from the cyclic initialization data. Consecutive spaces at the end of the model number are deleted. Unit No.: Text string of decimal numbers. Leading zeros are suppressed. Range for a CPU Bus Unit: 0 to 15. Special I/O Units: 0 to 95. The match/no match results are given in the following format: Result=Matched Same Result=Not matched Different

Note The verification results are given only for the EtherCAT slaves that were compared.

9-15 Compatibility between Backup-related Files

The files may not be compatible if you back up and restore data under different conditions.
The files may not be compatible in these three cases:

- When the function that was used to back up data is different from the function that was used to restore it.
- When the model number of the CPU Unit where the data was backed up from does not match the model number where data is being restored.
- When the unit versions of the CPU Unit, other Units, or slaves where the data was backed up from do not match the unit versions where data is being restored.

In this context, the term *restore* is used collectively for these backup functions: *restore*, *automatic transfer*, *program transfer*, and *read* (back up).

9-15-1 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.

(○: Compatible, ×: Not compatible.)

Function used to back up data	Function used to restore data				
	Restoring with SD Memory Card backup functions (SD Memory Card to Controller)	Automatic transfer and program-transfer	Restoring with Sysmac Studio Controller backup functions (computer to Controller)	Restoring with Sysmac Studio variable and memory backup functions (computer to Controller)	Importing Sysmac Studio backup file data (computer to project)
Backing up with SD Memory Card backup functions (Controller to SD Memory Card)	○	○	○	×	○*1
Backing up with Sysmac Studio Controller backup functions (Controller to computer)	○	○	○	×	○*1
Backing up with Sysmac Studio variable and memory data backup functions (Controller to computer)	×	×	×	○	×
Exporting from a Sysmac Studio backup file (project to computer)	○*1	○*1	○*1	×	○

*1. The following data is not included.

- The built-in EtherNet/IP port name and built-in EtherNet/IP tag data link settings in the Controller Setup

- Words allocated to CPU Bus Units in the Unit Configuration
- Operation authority verification
- User authentication
- Data Trace Settings
- Time zone setting
- Present values of variables
- Present values of memory used for CJ-series Units
- Absolute encoder home offsets

Additional Information

The files that are handled for backing up variables and memory from the Sysmac Studio are not compatible with other backup files.
Refer to *9-8 Sysmac Studio Variable and Memory Backup Functions* on page 9-53 for details on these functions.

9-15-2 Compatibility between CPU Unit Models

The following table shows the file compatibility when the CPU Unit model where the data was backed up from is different from the group where the data is being restored.

(○: Compatible, ×: Not compatible.)

CPU Unit model where data was backed up	CPU Unit model to restore to											
	NX701-	NX502-	NX102-		NX1P2-				NJ501-	NJ301-	NJ101-	
	1700 1600	1500 1400 1300	1200 1100 1000	9000	1140DT 1140DT1 1040DT 1040DT1	9024DT 9024DT1	9B40DT 9B40DT1	9B24DT 9B24DT1	1500 1400 1300	1200 1100	100 0	900 0
NX701-1700 NX701-1600	○	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1
NX502-1300 NX502-1400 NX502-1500	×*1	○	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1
NX102-1200 NX102-1100 NX102-1000	×*1	×*1	○	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1
NX102-9000	×*1	×*1	×*1	○	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1
NX1P2-1140DT NX1P2-1140DT1 NX1P2-1040DT NX1P2-1040DT1	×*1	×*1	×*1	×*1	○	×*1	×*1	×*1	×*1	×*1	×*1	×*1
NX1P2-9024DT NX1P2-9024DT1	×*1	×*1	×*1	×*1	×*1	○	×*1	×*1	×*1	×*1	×*1	×*1
NX1P2-9B40DT NX1P2-9B40DT1	×*1	×*1	×*1	×*1	×*1	×*1	○	×*1	×*1	×*1	×*1	×*1
NX1P2-9B24DT NX1P2-9B24DT1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	○	×*1	×*1	×*1	×*1
NJ501-1500 NJ501-1400 NJ501-1300	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	○	×*1	×*1	×*1
NJ301-1200 NJ301-1100	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	○	×*1	×*1
NJ101-1000	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	○	×*1

CPU Unit model where data was backed up	CPU Unit model to restore to											
	NX701-	NX502-	NX102-		NX1P2-				NJ501-	NJ301-	NJ101-	
	1700 1600	1500 1400 1300	1200 1100 1000	9000	1140DT 1140DT1 1040DT 1040DT1	9024DT 9024DT1	9B40DT 9B40DT1	9B24DT 9B24DT1	1500 1400 1300	1200 1100	100 0	900 0
NJ101-9000	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	×*1	○

*1. The Sysmac Studio variable and memory backup functions are compatible. However, a CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher are required. Refer to *9-8-3 Compatibility between CPU Unit Models* on page 9-54 for the compatibility between CPU Unit models for the Sysmac Studio variable and memory backup functions.

Additional Information

Database Connection CPU Units, SECS/GEM CPU Units, NJ Robotics CPU Units, Robot Integrated CPU Units, and NC Integrated Controller are not compatible.
Refer to the relevant manuals for specific Units for details on the compatibility of these CPU Units.

Even if the CPU Unit models are compatible, there may be restrictions between various CPU Unit models.
The following table shows which restoration function can be used based on whether the CPU Unit models are compatible.

(○: Restored, ×: Not restored.)

Compatibility between CPU Unit Models	Function used to restore data				
	Restoring with SD Memory Card backup functions (SD Memory Card to Controller)	Automatic transfer and program transfer	Restoring with Sysmac Studio Controller backup functions (computer to Controller)*1	Restoring with Sysmac Studio variable and memory backup functions (computer to Controller)	Importing Sysmac Studio backup file data (computer to project)
Compatible	○*2	○*2	○	○	○
Not compatible	×*3	×*4	×	×	×

*1. Only the files that were backed up using this function can be restored.
*2. If the contents of the backup file are outside the range of specifications where the data is restored, the Controller will not operate normally. When you operate the Controller, a major fault level Controller error or a partial fault level Controller error will occur. For example, this error occurs if the number of controlled axes that is used is outside the specifications.
*3. A Restore Start Failed observation will occur.
*4. The Error in Starting Automatic Transfer (a major fault level Controller error) occurs during an automatic transfer. The SD Memory Card Program Transfer Failed to Start error (an observation level Controller error) occurs during a program transfer.

9-15-3 Compatibility between Unit Versions of CPU Units

The following table shows the compatibility of backup files when the unit versions of the CPU Unit are different between where the data was backed up and where it is being restored.
You can restore data without any restrictions if the unit versions are the same before and after the backup and restoration.

(○: Restored, ×: Not restored.)

Unit version of CPU Unit	Function used to restore data			
	Restoring with SD Memory Card backup functions (SD Memory Card to Controller)	Automatic transfer and program transfer	Restoring with Sysmac Studio Controller backup functions (computer to Controller)	Restoring with Sysmac Studio variable and memory backup functions (computer to Controller)
Version of CPU Unit where data is being restored is newer.	○	○	○	○
Unit version of CPU Unit where data is being restored is older.	×	×*1	×	○

*1. The Error in Starting Automatic Transfer (a major fault level Controller error) occurs during an automatic transfer. The SD Memory Card Program Transfer Failed to Start error (an observation level Controller error) occurs during a program transfer.

9-16 Functions that cannot be Executed during Backup Functions

The following functions cannot be executed at the same time as any of the backup functions. Do not execute any backup function while the CPU Unit is executing any of these functions. Also, do not execute any of these functions during execution of any of the backup functions.

- While a backup function is being performed
- Synchronization transfer from the computer to the Controller
- Execution of online editing
- Execution of Memory All Clear operation
- Time zone changes
- Execution of the Save Cam Table instruction (MC_SaveCamTable)
- Execution of CPU Unit name write operation
- Execution of transferring Slave Terminal parameters

10

Communications Setup

This section describes how to go online with the CPU Unit and how to connect to other devices.

10-1 Communications System Overview 10-2
10-1-1 Introduction 10-3
10-2 Connection with Sysmac Studio 10-9
10-2-1 Configurations That Allow Online Connections 10-9
10-2-2 Configurations That Do Not Allow Online Connections 10-11
10-2-3 Secure Communication 10-12
10-3 Connection with Other Controllers or Slaves 10-15
10-3-1 Connection Configurations between Controllers 10-15
10-3-2 Connection Configuration between Controllers and Slaves 10-18
10-4 Connection with HMIs or Serial Communications Devices 10-19
10-4-1 Connection with HMIs 10-19
10-4-2 Connection with Serial Communications Devices 10-20

10-1 Communications System Overview

This section gives an overview of the communications systems that are supported by NJ/NX-series Controllers.

The shaded steps in the overall procedure that is shown below are related to the communications systems.

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

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

Step 3. Mounting and Setting Hardware

Step 4. Wiring

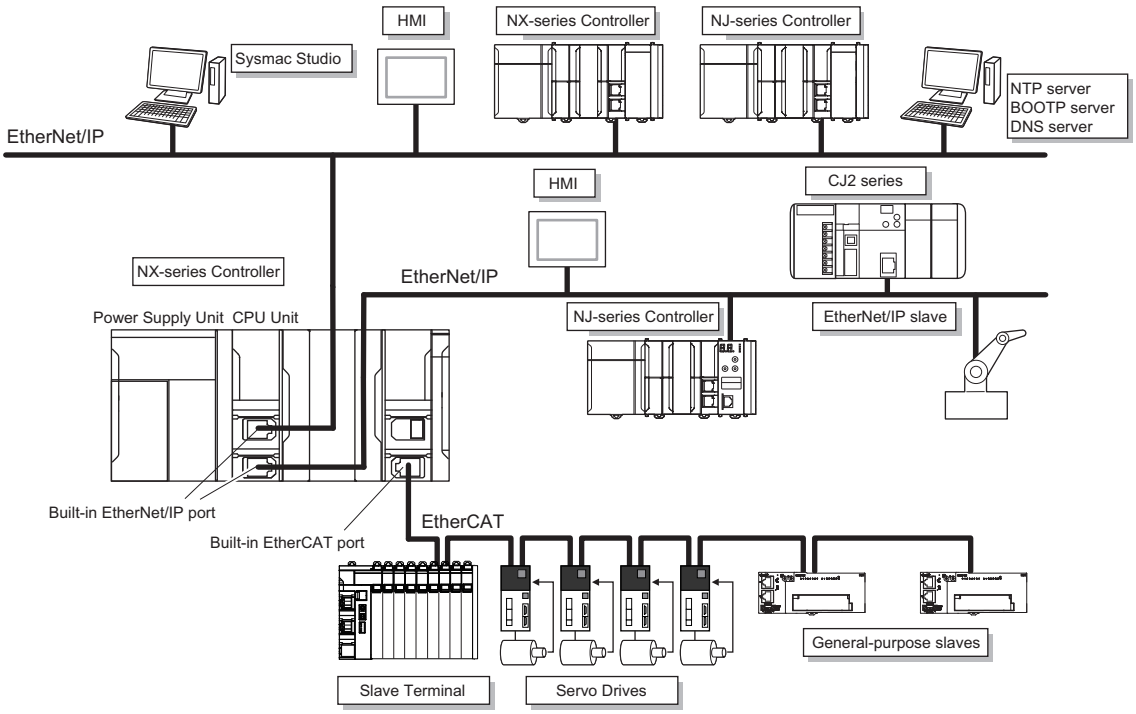
Step 5. Checking Operation and Starting Operation on the Actual System

Refer to *1-3 Overall Operating Procedure for the NJ/NX-series* on page 1-23 for details.

10-1-1 Introduction

● NX701 System

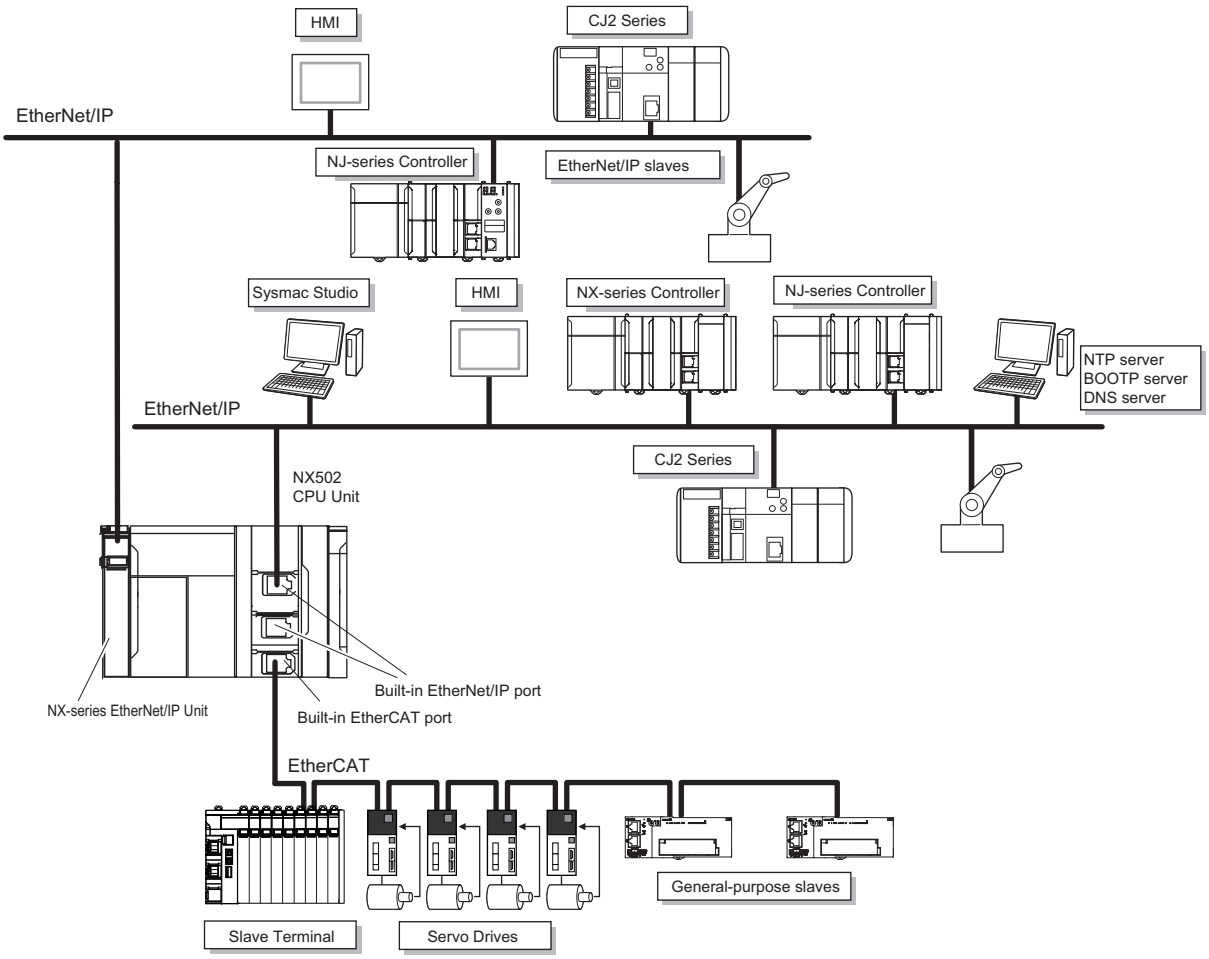
You can use the NX701 System to build the communications system shown below.



Connection		Connection method
Sysmac Studio connection		Use the built-in EtherNet/IP port.
Connections between Controllers	Connections with NJ/NX-series Controller or CJ2 CPU Unit	Use the built-in EtherNet/IP port.
Connections between Controllers and slaves	Connections to Servo Drives and general-purpose slaves	Use the built-in EtherCAT port.
Connections to HMIs		Use the built-in EtherNet/IP port.
Connections to servers	Connections to FTP servers, BOOTP servers, DNS servers, or NTP servers	Use the built-in EtherNet/IP port.

● NX502 System

You can use the NX502 System to build the communications system shown below.

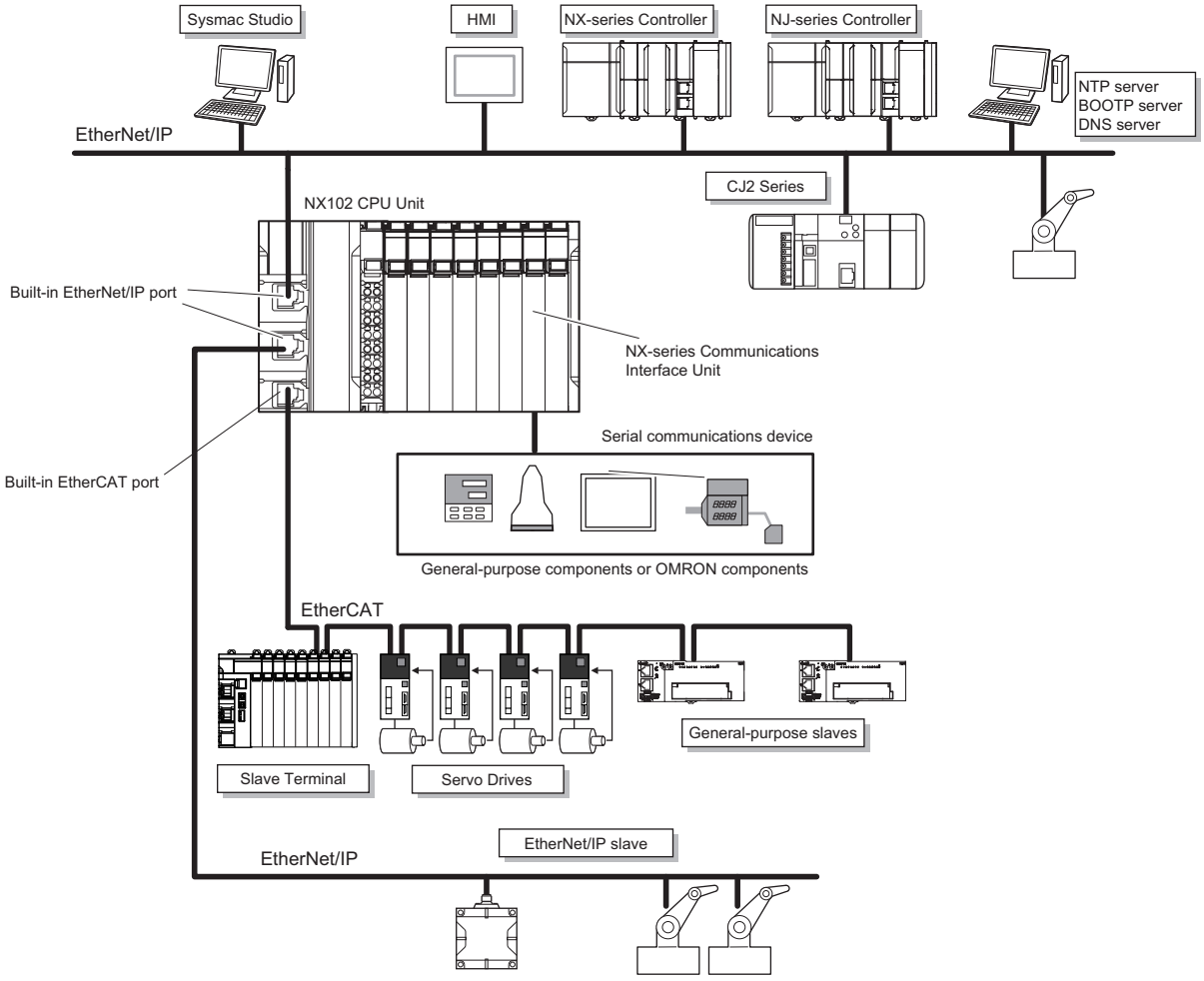


Connection		Connection method
Sysmac Studio connection		Use the built-in EtherNet/IP port or EtherNet/IP port*1 on the NX-series EtherNet/IP Unit.
Connections between Controllers	Connections with NJ/NX-series Controller or CJ2 CPU Unit	
Connections to HMIs		
Connections to servers	Connections to FTP servers, BOOTP servers, DNS servers, or NTP servers	
Connections between Controllers and slaves	Connections to Servo Drives and general-purpose slaves	Use the built-in EtherCAT port.

*1. For information on how to use the NX-series EtherNet/IP Unit, refer to the NX-series EtherNet/IP Unit User's Manual (Cat. No. W627).

NX102 System

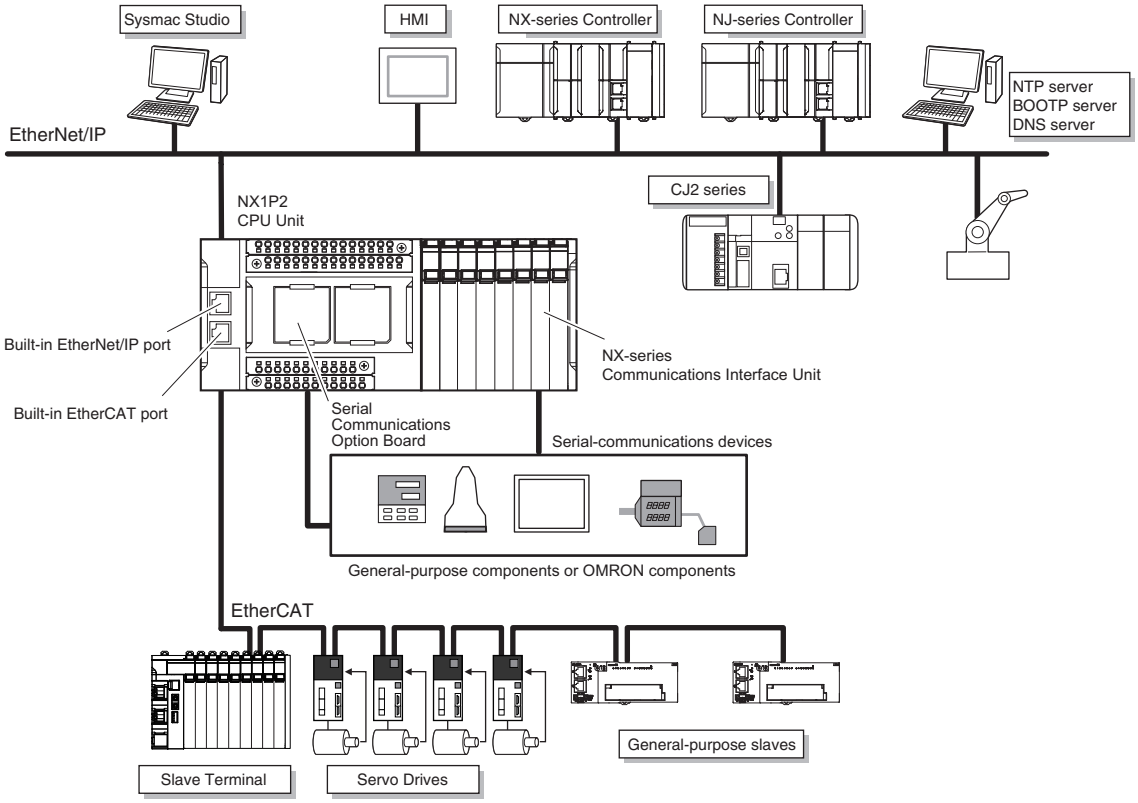
You can use the NX102 System to build the communications system shown below.



Connection		Connection method
Sysmac Studio connection		Use the Built-in EtherNet/IP port.
Connections between Controllers	Connections with NJ/NX-series Controller or CJ2 CPU Unit	Use the Built-in EtherNet/IP port.
Connections between Controllers and slaves	Connections to Servo Drives and generalpurpose slaves	Use the built-in EtherCAT port.
Connections to HMIs		Use the Built-in EtherNet/IP port.
Connections to servers	Connections to FTP servers, BOOTP servers, DNS servers, or NTP servers	Use the Built-in EtherNet/IP port.

● NX1P2 System

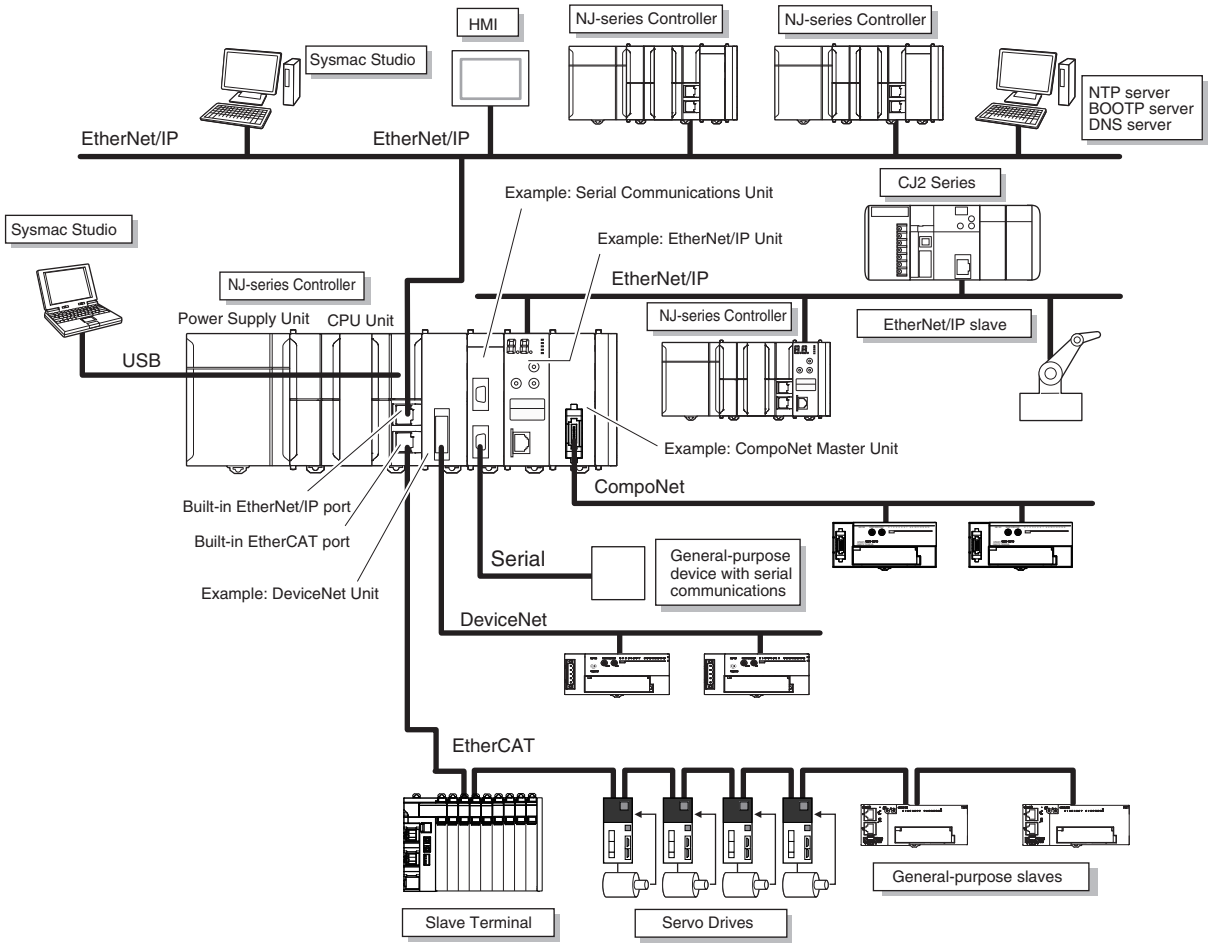
You can use the NX1P2 System to build the communications system shown below.



Connection		Connection method
Sysmac Studio connection		Use the built-in EtherNet/IP port.
Connections between Controllers	Connections with NJ/NX-series Controller or CJ2 CPU Unit	Use the built-in EtherNet/IP port.
Connections between Controllers and slaves	Connections to Servo Drives and general-purpose slaves	Use the built-in EtherCAT port.
Connections to HMIs		Use the built-in EtherNet/IP port.
Connections to servers	Connections to FTP servers, BOOTP servers, DNS servers, or NTP servers	Use the built-in EtherNet/IP port.

● NJ-series System

You can use the NJ-series System to build the communications system shown below.



Connection		Connection method
Sysmac Studio connection		Use USB or the built-in EtherNet/IP port.
Connections between Controllers	Connections with NJ/NX-series Controller or CJ2 CPU Unit	Use the built-in EtherNet/IP port or a port on an EtherNet/IP Unit.*1
Connections between Controllers and slaves	Connections to Servo Drives and general-purpose slaves	Use the built-in EtherCAT port.
	I/O controls	Mount a DeviceNet Unit and use DeviceNet or mount a CompoNet Master Unit and use CompoNet.
Connections to HMIs		Use the built-in EtherNet/IP port or a port on an EtherNet/IP Unit.*1*2
Connections for serial communications		Mount a Serial Communications Unit.
Connections to servers	Connections to FTP servers*3, BOOTP servers, DNS servers, or NTP servers	Use the built-in EtherNet/IP port or a port on an EtherNet/IP Unit.*1

*1. Use an EtherNet/IP Unit with unit version 2.1 or later.
Also use a CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher.
Refer to *A-20 Version Information for NJ-series Controllers* on page A-298 for information on version upgrades.

*2. Connect an NA-series PT to the built-in EtherNet/IP port on the CPU Unit. To perform troubleshooting from an NS-series PT, connect the PT to the built-in EtherNet/IP port on the CPU Unit.

*3. A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use FTP servers.

10-2 Connection with Sysmac Studio

This section describes the configurations for connecting the Sysmac Studio to an NJ/NX-series Controller.

10-2-1 Configurations That Allow Online Connections

You can connect online from the Sysmac Studio to the peripheral USB port or built-in EtherNet/IP port of the NJ/NX-series CPU Unit or the EtherNet/IP port of the NX-series EtherNet/IP Unit.

Precautions for Correct Use

For the following CPU Unit versions, use Sysmac Studio on Windows 10 version 1803 or higher OS on the computer. Sysmac Studio on Windows 10 version 1709 or lower cannot connect online to the Controller.

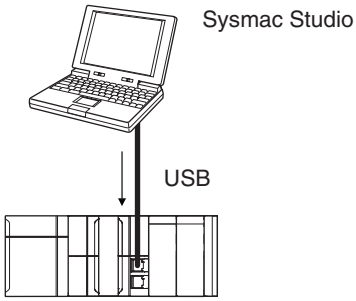
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

Precautions for Correct Use

For CPU Units with unit version 1.40 or later, the Controller reset is required in any of the following cases. Connect online according to the message on the Sysmac Studio.

- When an attempt is made to connect the Sysmac Studio where a project with project unit version 1.40 or later is opened online to a CPU Unit in the factory default condition. However, the Controller reset is not necessary for the NX1P2-9B□□□□ CPU Units only with unit version 1.40 or later.

● Connecting with USB



A direct connection is made from the computer that runs Sysmac Studio. You do not need to specify the connection device.

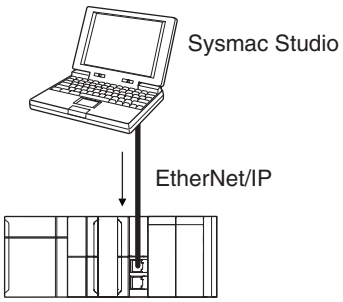
Note Connect a computer and the CPU Unit with a USB 2.0 certified cable. Do not use a USB hub to connect them.

Note You cannot connect a computer to an NX701 CPU Unit with hardware revision A or later, NX502 CPU Unit, NX102 CPU Unit, or NX1P2 CPU Unit via USB, because it does not provide a peripheral USB port.

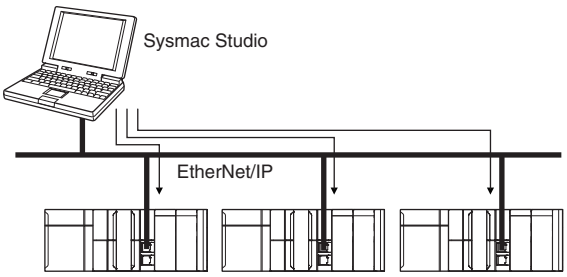
● Connecting to Built-in EtherNet/IP Ports

1:1 Connection

1:N Connections



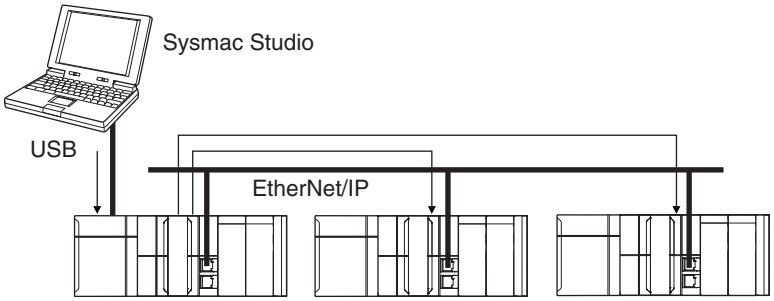
- A direct connection is made from the computer that runs Sysmac Studio. You do not need to specify the IP address or connection device.
- You can make the connection either with or without a Ethernet switch.
- You can use either a cross cable or a straight cable.
- For NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units, 1:1 connection is supported only for port 1.



- 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 Built-in EtherNet/IP Ports via USB

You can connect to the built-in EtherNet/IP port through a USB port of the NJ-series CPU Unit.



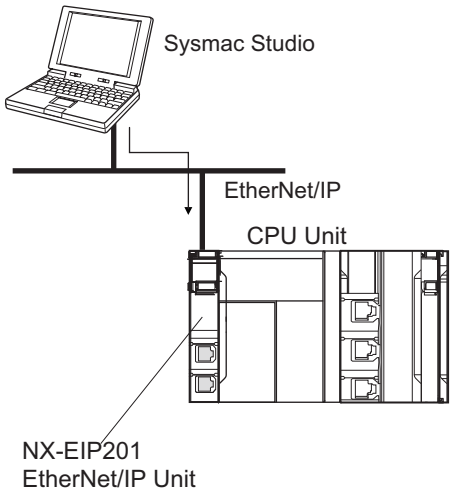
- Specify the IP address of the remote node from the Sysmac Studio.
- You can use either a cross cable or a straight cable.

Note Connect a computer and the CPU Unit with a USB 2.0 certified cable. Do not use a USB hub to connect them.

Note The NX701 CPU Units with hardware revision A or later, NX502, NX102, and NX1P2 CPU Units do not provide a peripheral USB port.

● Connecting through an NX-series EtherNet/IP Unit

You can connect through an NX-EIP201 EtherNet/IP Unit that is connected to an NX-series CPU Unit.

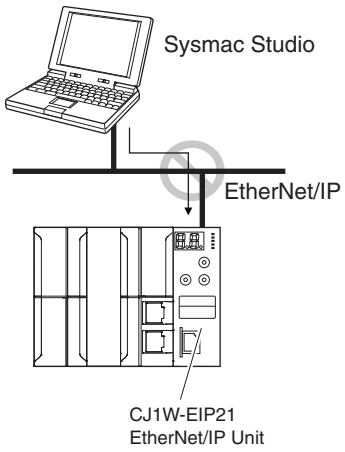


Note You can connect an NX-EIP201 EtherNet/IP Unit only on the NX502 CPU Units.

10-2-2 Configurations That Do Not Allow Online Connections

● Connecting through an CJ-series EtherNet/IP Unit

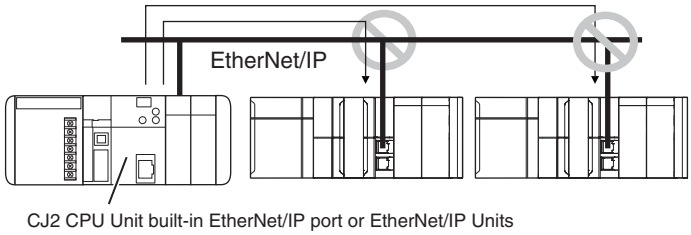
You cannot connect through a CJ1W-EIP21 EtherNet/IP Unit that is connected to an NJ-series CPU Unit.



Note You cannot use a CJ1W-EIP21 EtherNet/IP Unit for NX-series CPU Units.

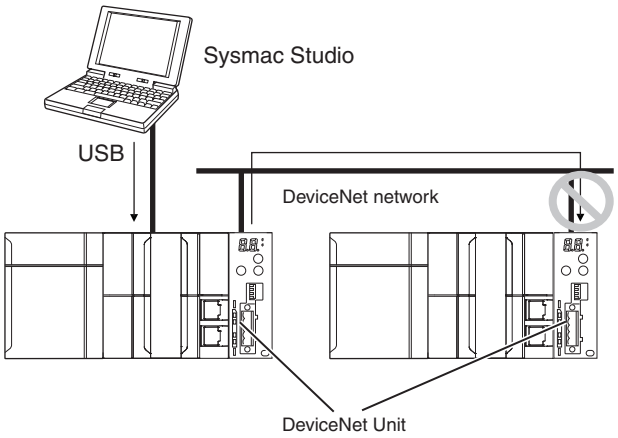
● Routing through CS/CJ-series EtherNet/IP Units/Ports

You cannot connect to an NJ/NX-series CPU Unit by routing through a CS/CJ-series Ethernet/IP Unit or port (CS1W-EIP2, CJ1W-EIP21, CJ2 CPU Unit built-in EtherNet/IP port, or CJ2M CPU Unit built-in EtherNet/IP port).



● Routing through Networks Other Than EtherNet/IP, Such as DeviceNet

You cannot route through any networks other than EtherNet/IP networks. (For example, routing is not possible for Controller Link networks and DeviceNet networks.)



Note You cannot connect a DeviceNet Unit with NX-series CPU Units.

10-2-3 Secure Communication

Secure communication function is intended to improve the security performance of communications between Sysmac Studio and the Controller.

Overview of Secure Communications

Since secure communication function encrypts communication data and sends and receives data with a hash value, it is useful to prevent data tapping and tampering by third parties.

Additional Information

To improve security, we recommend installing the Sysmac Studio on an operating system that is within the Microsoft Windows support period.

Version Information

Secure communication is available in the following CPU Units of the stated versions.
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

Precautions for Correct Use

The secure communication function may affect the communication response performance and the execution processing performance of some instructions.
When you use a CPU Unit that supports the secure communication, check the operation sufficiently in advance.
If there is a problem with the performance, consider changing settings to *allow connection to Sysmac Studio or an NA-series Programmable Terminal that do not support secure communication*.

Combinations of Versions for Which Secure Communication Functions Are Available

This section describes the combination of CPU Unit and Sysmac Studio versions and the combination of CPU Units and NA-series Programmable Terminal versions for which secure communication functions can be used.

● Combination of CPU Unit and Sysmac Studio

The combinations of the CPU Unit and Sysmac Studio versions for which the secure communication function can be used are as follows.

Unit version of CPU Unit	Sysmac Studio version	
	Ver.1.50 or higher	Ver. 1.49 or lower
Ver.1.49*1 or later	Secure communication available	Not connectable Follow the DIP switch settings of the CPU Unit. *3
Ver.1. 48 *2 or earlier	Can be connected without using secure communication	Can be connected without using secure communication

*1. This is the unit version for the NJ-series, NX102, or NX1P2 CPU Units. If you use the NX701 CPU Unit, read it as version 1.29.

*2. This is the unit version for the NJ-series, NX102, or NX1P2 CPU Units. If you use the NX701 CPU Unit, read it as version 1.28.

*3. For details on the DIP switch settings of the CPU Unit, refer to *How to Allow Connection to Sysmac Studio or NA-series Programmable Terminal that Do Not Support Secure Communication* on page 10-13.

● Combination of CPU Unit and NA-series Programmable Terminal

The combinations of the CPU Unit and the NA-series Programmable Terminal or Soft-NA versions for which the secure communication functions can be used are as follows.

Unit version of CPU Unit	NA-series Programmable Terminal		Soft-NA
	Ver.1.16 or later	Ver.1.15 or earlier	All versions
Ver.1.49*1 or later	Secure communication available	Follow the DIP switch settings of the CPU Unit. *3	Follow the DIP switch settings of the CPU Unit. *3
Ver.1. 48 *2 or earlier	Can be connected without using secure communication	Can be connected without using secure communication	Can be connected without using secure communication

*1. This is the unit version for the NJ-series, NX102, or NX1P2 CPU Units. If you use the NX701 CPU Unit, read it as version 1.29.

*2. This is the unit version for the NJ-series, NX102, or NX1P2 CPU Units. If you use the NX701 CPU Unit, read it as version 1.28.

*3. For details on the DIP switch settings of the CPU Unit, refer to *How to Allow Connection to Sysmac Studio or NA-series Programmable Terminal that Do Not Support Secure Communication* on page 10-13.

How to Allow Connection to Sysmac Studio or NA-series Programmable Terminal that Do Not Support Secure Communication

The following versions of Sysmac Studio, NA-series Programmable Terminal, and Soft-NA do not support secure communication.

- Sysmac Studio: Version 1.49 or lower
- NA-series Programmable Terminal: Version 1.15 or earlier
- Soft-NA: All versions

The following unit versions of the CPU Unit support secure communication.

- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later

When combining these for communication, it is necessary to set the DIP switches of the CPU Unit. Set the DIP switches are as follows. However, in these cases, normal communication is used and secure communication cannot be performed.

DIP switch				Operation
4	3	2	1	
OFF	OFF	ON	ON	Enables connections to the Sysmac Studio or NA-series Programmable Terminal that do not support secure communication.
ON	OFF	ON	ON	Enables connections to the Sysmac Studio or NA-series Programmable Terminal that do not support secure communication, and starts the CPU Unit in Safe Mode.
OFF	ON	ON	ON	Enables connections to the Sysmac Studio or NA-series Programmable Terminal that do not support secure communication and automatic transfer from SD Memory Card

- The DIP switch on the NX701 CPU Unit has 8 positions. When using the function, set the DIP switch positions 5 to 8 to OFF.
- This function operates when the power supply to the CPU Unit is turned ON after you change the DIP switch settings. If you want to use this function, turn ON power to the CPU Unit after you change the DIP switch settings.

Precautions for Correct Use

When you use a CPU Unit that supports secure communication and an NA-series Programmable Terminal or a Soft-NA that do not support secure communication, please note that there are following restrictions if the DIP switch is not set as shown above.

- The NA-series Programmable Terminal cannot be connected.
- The troubleshooter of the NS-series Programmable Terminal cannot be used.

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

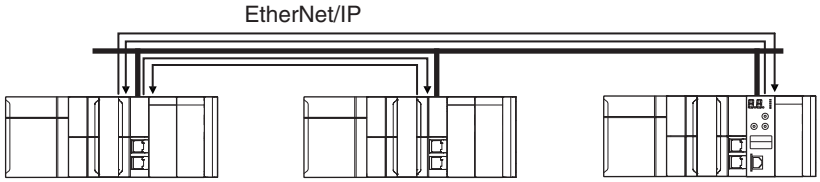
You can use a built-in EtherNet/IP port, a CJ2W-EIP21 EtherNet/IP Unit, or an NX-EIP201 EtherNet/IP Unit.

Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for details on the built-in EtherNet/IP port, *CJ-series EtherNet/IP Unit Operation Manual for NJ-series CPU Unit (Cat. No. W495)* for details on the CJ1W-EIP21 EtherNet/IP Unit, and *NX-series EtherNet/IP Unit User's Manual (Cat. No. W627)* for details on the NX-EIP201 EtherNet/IP Unit.

Note 1. You cannot use a CJ1W-EIP21 EtherNet/IP Unit for NX-series CPU Units.
Note 2. You can use an NX-EIP201 EtherNet/IP Unit only on the NX502 CPU Units.

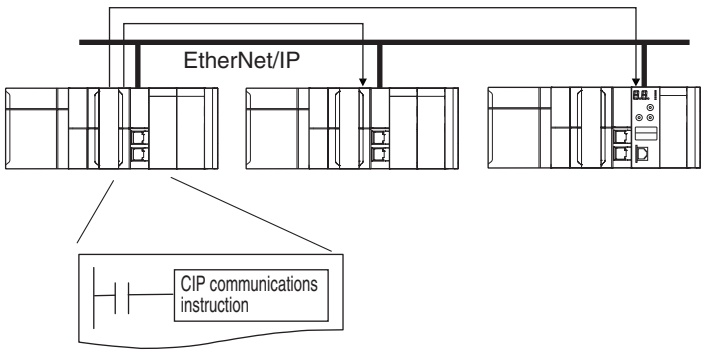
● Tag Data Links

You can create tag data links between NJ/NX-series CPU Units on an EtherNet/IP network.



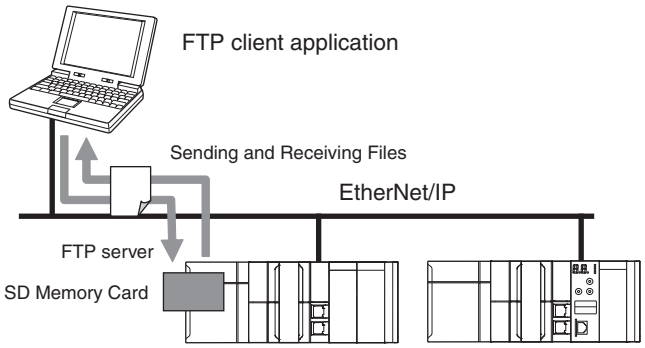
● Message Communications

You can send CIP messages from the user program.



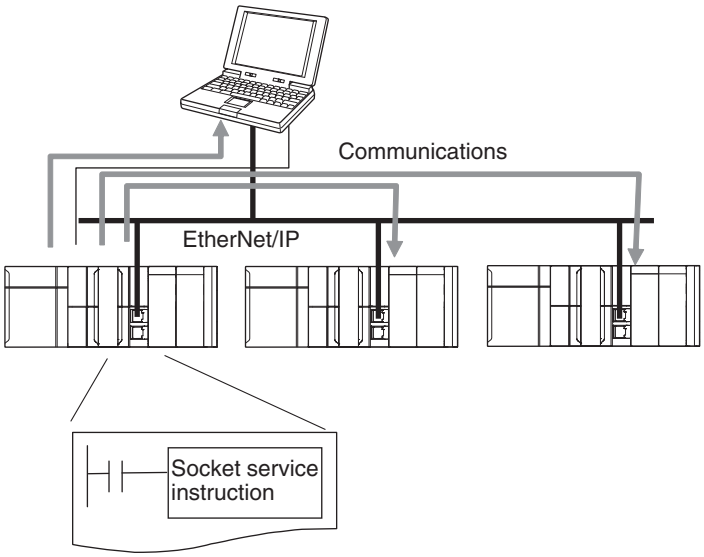
● Sending and Receiving Files

You can send and receive files on the SD Memory Card that is inserted in the NJ/NX-series CPU Unit from an FTP client application.



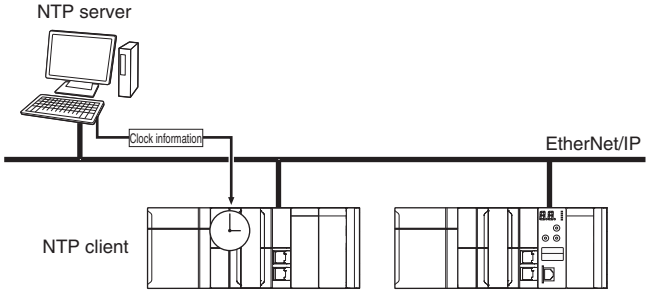
● Socket Services

You can directly use TCP or UDP from the user program to send and receive any data with remote nodes between a host computer and the Controller, or between Controllers.
The socket services are supported only for the built-in EtherNet/IP ports.



● Updating Clock Information

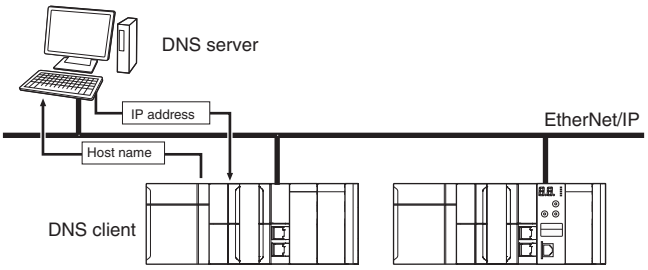
You can obtain clock information from an NTP server to update the built-in clock.



Specifying Host Names

You can use the DNS client or set up your Hosts so that you can specify the IP address of the NTP server or SNMP manager or the target destination of a socket instruction or CIP communications instruction with a host name instead of an IP address.

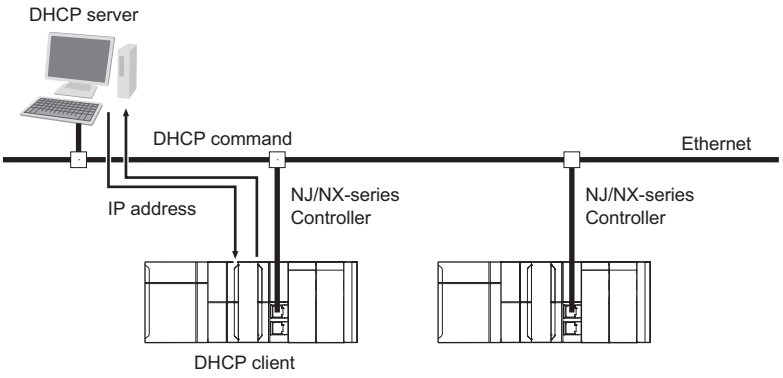
Example: Setting Host Names on the DNS Server



Obtaining an IP Address When the Power Is Turned ON

You can obtain an IP address for the built-in EtherNet/IP port from the BOOTP server when the power supply is turned ON.

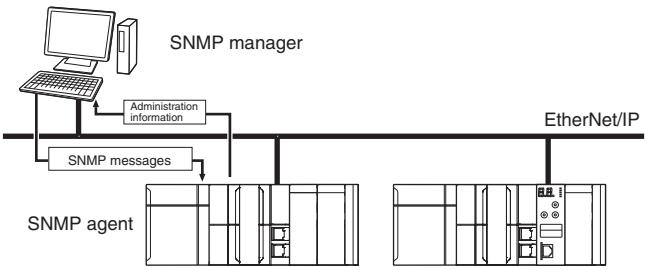
You can obtain an IP address for the built-in EtherNet/IP port from the DHCP server when the power supply is turned ON.



Note You can use the DHCP server only with the NX502 CPU Units.

Specifying an SNMP Agent

Built-in EtherNet/IP port internal status information is provided to network management software that uses an SNMP manager.

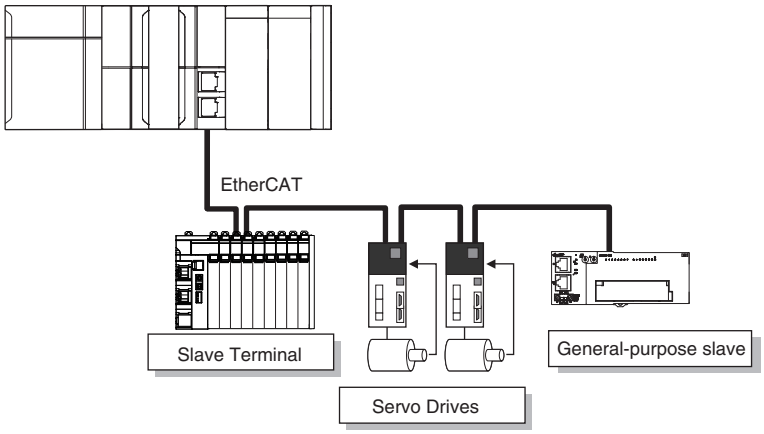


10-3-2 Connection Configuration between Controllers and Slaves

EtherCAT

High-speed, high-precision communications are possible with Servo Drives and general-purpose slaves.

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



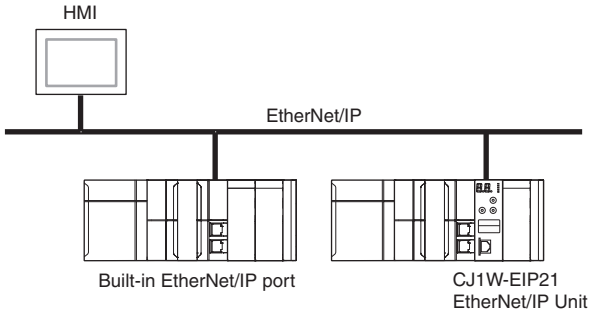
10-4 Connection with HMIs or Serial Communications Devices

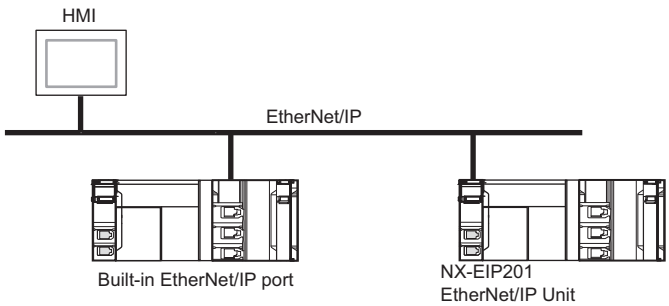
This section shows the connection configurations used to connect HMIs and devices with serial communications to the NJ/NX-series Controller.

10-4-1 Connection with HMIs

● EtherNet/IP

You can use a built-in EtherNet/IP port, a CJ2W-EIP21 EtherNet/IP Unit, or an NX-EIP201 EtherNet/IP Unit to connect to an HMI.





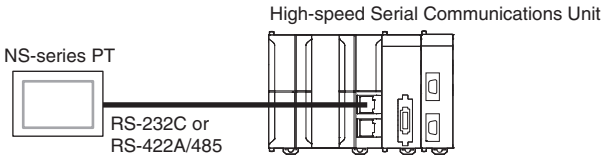
Note 1. You cannot use a CJ1W-EIP21 EtherNet/IP Unit for NX-series CPU Units.
Note 2. You can use an NX-EIP201 EtherNet/IP Unit only on the NX502 CPU Units.

Connect an NA-series PT to the built-in EtherNet/IP port on the CPU Unit.
To perform troubleshooting from an NS-series PT, connect the PT to the built-in EtherNet/IP port on the CPU Unit.

Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)* for details on the built-in EtherNet/IP port, *CJ-series EtherNet/IP Unit Operation Manual for NJ-series CPU Unit (Cat. No. W495)* for details on the CJ1W-EIP21 EtherNet/IP Unit, and *NX-series EtherNet/IP Unit User's Manual (Cat. No. W627)* for details on the NX-EIP201 EtherNet/IP Unit.

● Serial Communications

You can use a Serial Communications Unit to connect to an NS-series PT.



Note You cannot use a Serial Communications Unit for NX-series CPU Units.

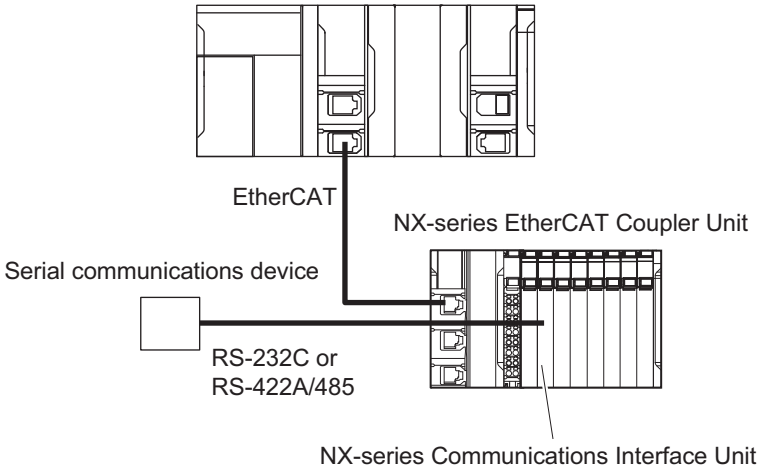
Refer to the *CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit* (Cat. No. W494) for details on the Serial Communications Unit.

10-4-2 Connection with Serial Communications Devices

NX701 CPU Units

Connect an NX-series Communications Interface Unit to the Communications Coupler Unit before use. Refer to the *NX-series Communications Interface Units User's Manual (Cat. No. W540)* for details of NX-series Communications Interface Units.

The following is an example where an NX-series EtherCAT Coupler Unit is used as the Communications Coupler Unit.

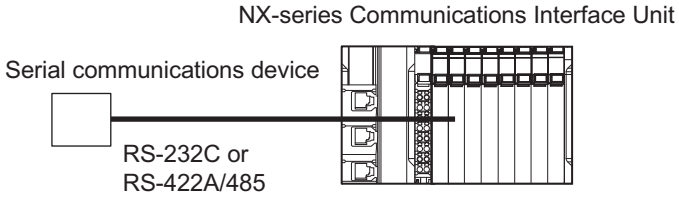


Precautions for Correct Use

The NX bus connector of the CPU Unit is provided for future expansion, so it cannot be used to connect any NX-series Communications Interface Unit.
You cannot use a CJ-series Serial Communications Unit or a Serial Communications Option Board to connect to an NX701 CPU Unit, either.

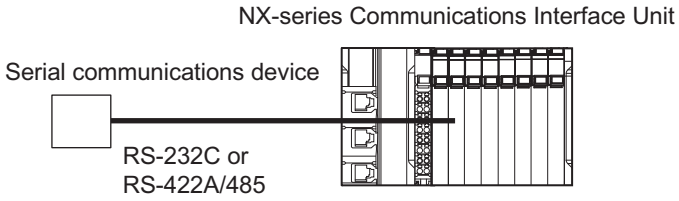
NX502 CPU Units

You can use an NX-series Communications Interface Unit.
Refer to the *NX-series Communications Interface Units User's Manual (Cat. No. W540)* for details of NX-series Communications Interface Units.



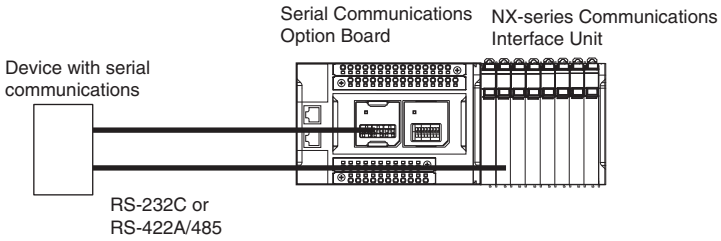
NX102 CPU Units

You can use an NX-series Communications Interface Unit.
Refer to the *NX-series Communications Interface Units User's Manual (Cat. No. W540)* for details of NX-series Communications Interface Units.



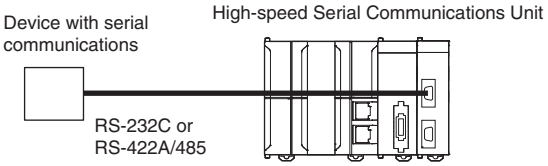
NX1P2 CPU Units

You can use a Serial Communications Option Board or an NX-series Communications Interface Unit to connect to an NX1P2 CPU Unit.
Refer to the *NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual* (Cat. No. W579) for details on the Serial Communications Option Board and *NX-series Communications Interface Units User's Manual* (Cat. No. W540) for details on the NX-series Communications Interface Unit.



NJ-series CPU Units

You can use a Serial Communications Unit to connect to an NJ-series CPU Unit.
Refer to the *CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit* (Cat. No. W494) for details on the Serial Communications Unit.


Device with serial communications
High-speed Serial Communications Unit
RS-232C or
RS-422A/485

11

Example of Actual Application Procedures

This section describes the procedures that are used to actually operate an NJ-series Controller.

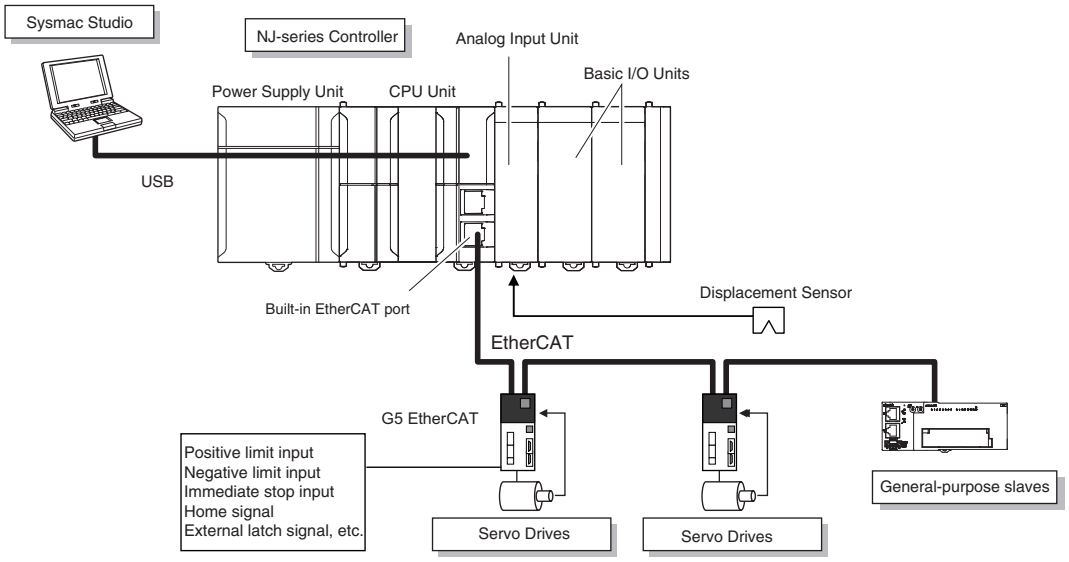
11-1 Example Application 11-2
11-1-1 System Configuration 11-2
11-1-2 Operation 11-2
11-2 Overview of the Example Procedure 11-3
11-2-1 Wiring and Settings 11-3
11-2-2 Software Design 11-3
11-2-3 Software Settings from the Sysmac Studio 11-4
11-2-4 Programming with the Sysmac Studio 11-7
11-2-5 Simulation with the Sysmac Studio 11-8
11-2-6 Checking Operation and Starting Operation on the Actual System 11-9

11-1 Example Application

This section describes an example application for an NJ-series Controller.

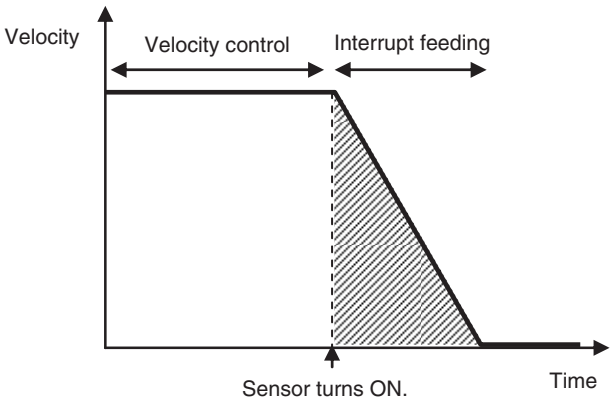
11-1-1 System Configuration

Unit name		Qty	Connected device
Power Supply Unit		1	---
CPU Unit		1	---
CJ-series Basic I/O Units		2	---
CJ-series Analog Input Unit		1	Displacement Sensor
EtherCAT slaves	Servo Drives (G5 EtherCAT)	2	---
	I/O Terminal	1	---



11-1-2 Operation

Interrupt feeding starts when the sensor signal changes to ON during velocity control.



The vertical position changes based on the analog input from the Displacement Sensor.

11-2 Overview of the Example Procedure

This section describes examples of the actual operating procedures for an NJ-series Controller.

11-2-1 Wiring and Settings

Wire the Controller and make the hardware settings.

11-2-2 Software Design

Design the I/O, tasks, POUs, and variables.

I/O Design

- Design the relationship between the external I/O and the Unit configuration.
- Determine the intervals at which to refresh external I/O.

Task and POU Design

Consider the following:

- What task configuration is required
- Which programs to assign to which tasks
- Which Units to assign to which tasks
- What processing to place in programs and what processing to place in function blocks and functions

Variable Design

Consider the following:

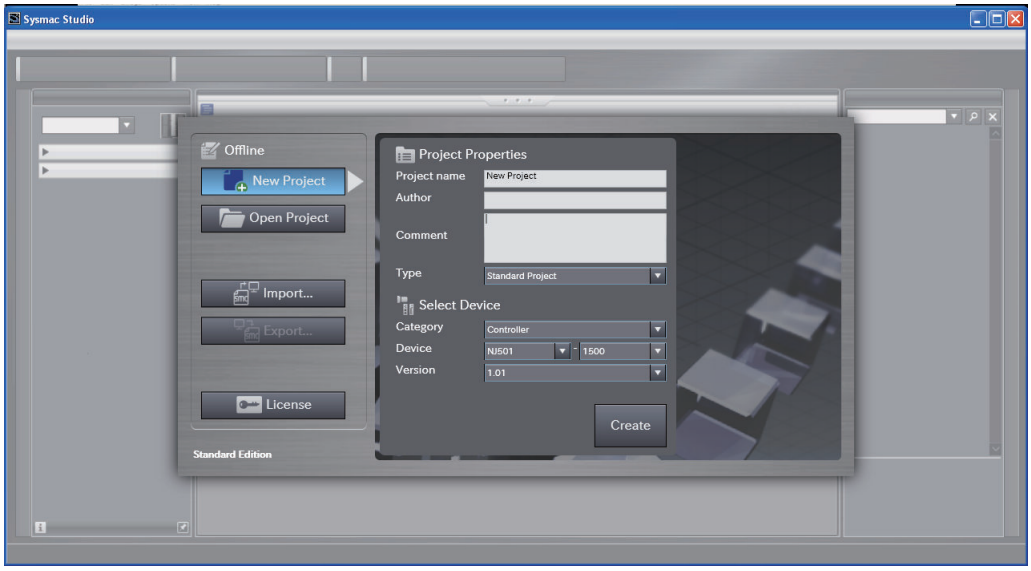
- The separation of variables into those that you use in more than one POU (global variables) and variables that you use in only specific POUs (local variables)
- Defining the variables names for the device variables that you use to access slaves and Units
- Defining the attributes of variables, such as the Name and Retain attributes
- Designing the data types of variables

11-2-3 Software Settings from the Sysmac Studio

On the Sysmac Studio, you set the Unit and slave configurations, register global variables and device variables, create axes (axis variables), and set the Controller Setup and Special Unit Setup.

Start the Sysmac Studio

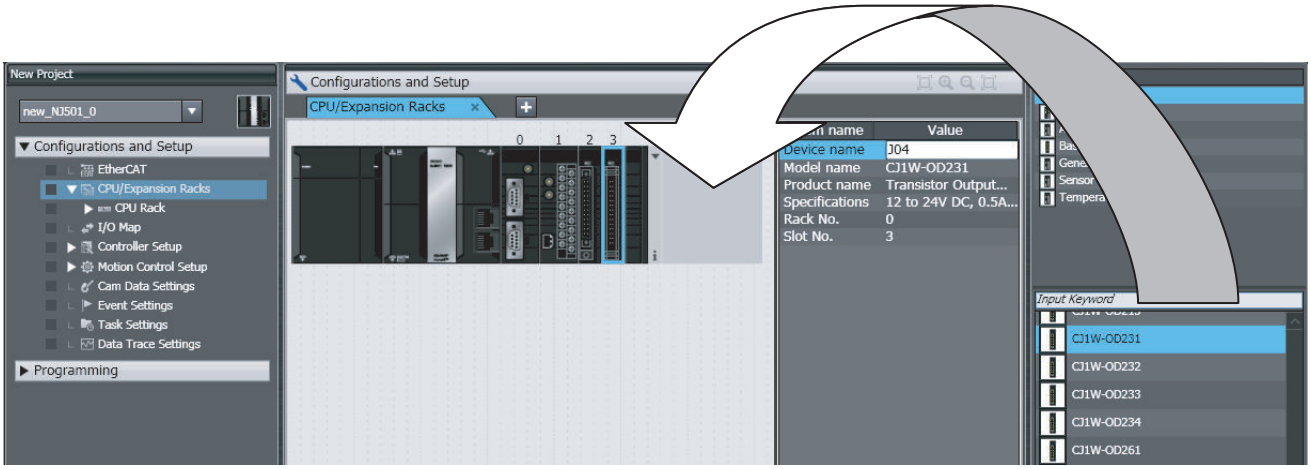
Create a project in Sysmac Studio.



Create the Unit Configuration

1 Double-click **CPU/Expansion Racks** under **Configurations and Setup**.

2 Create the Unit configuration by dragging Units.

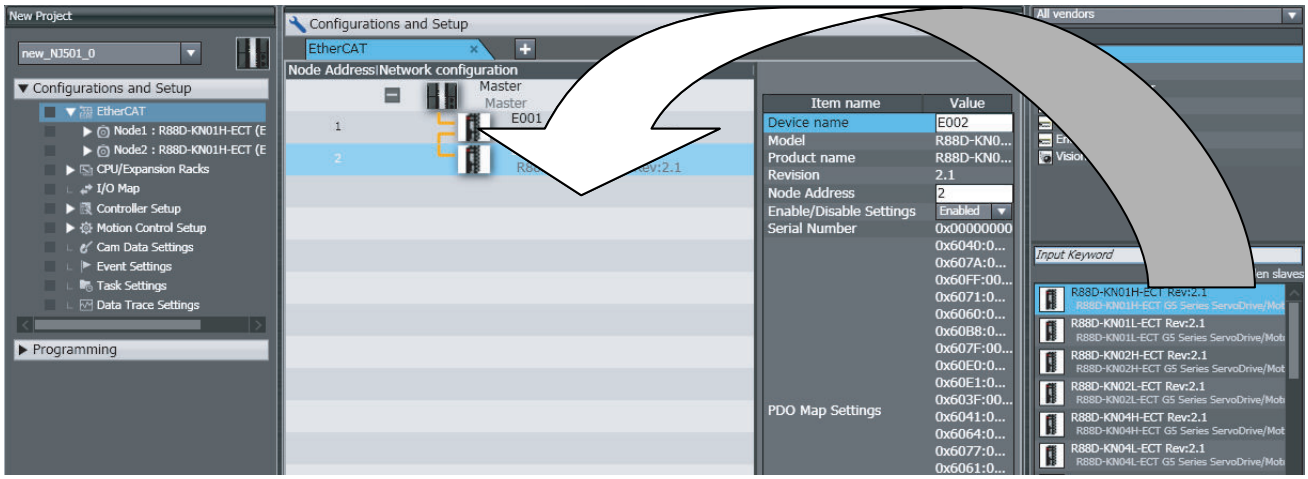


3 Select each Unit and make the required settings.

Create the EtherCAT Slave Configuration

1 Double-click **EtherCAT** under **Configurations and Setup**.

2 Create the slave configuration by dragging slaves.



3 Select the master and set the master parameters.

4 Select each slave and set the slave parameters.

Additional Information

At this point, you can use forced refreshing from the I/O Map to check the wiring.

Register the Global Variables and Device Variables

● Registering Global Variables

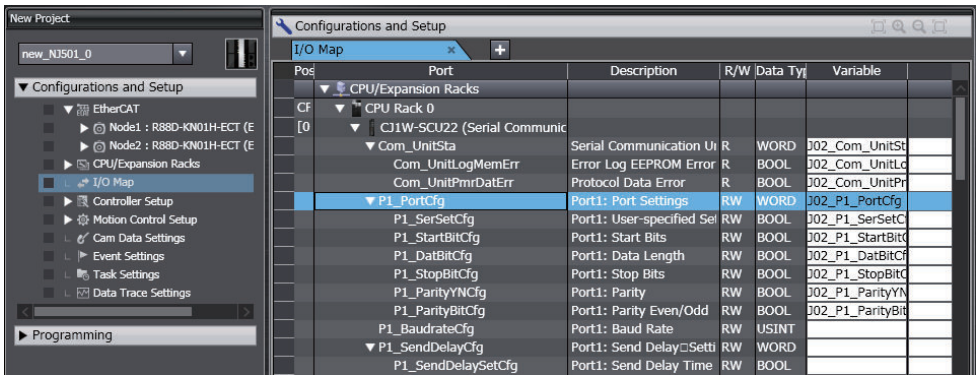
1 Double-click **Global Variables** under **Programming** - **Data**.

2 Register the global variables in the global variable table.

● Registering Device Variables

1 Double-click **I/O Map** under **Configurations and Setup**.

2 In the I/O Map, assign the variables to the I/O ports. (The I/O ports are created automatically from the Unit and slave configurations.)
You can automatically create device variable names with the Sysmac Studio. To do so, right-click an I/O port and select **Create Device Variable** from the menu.

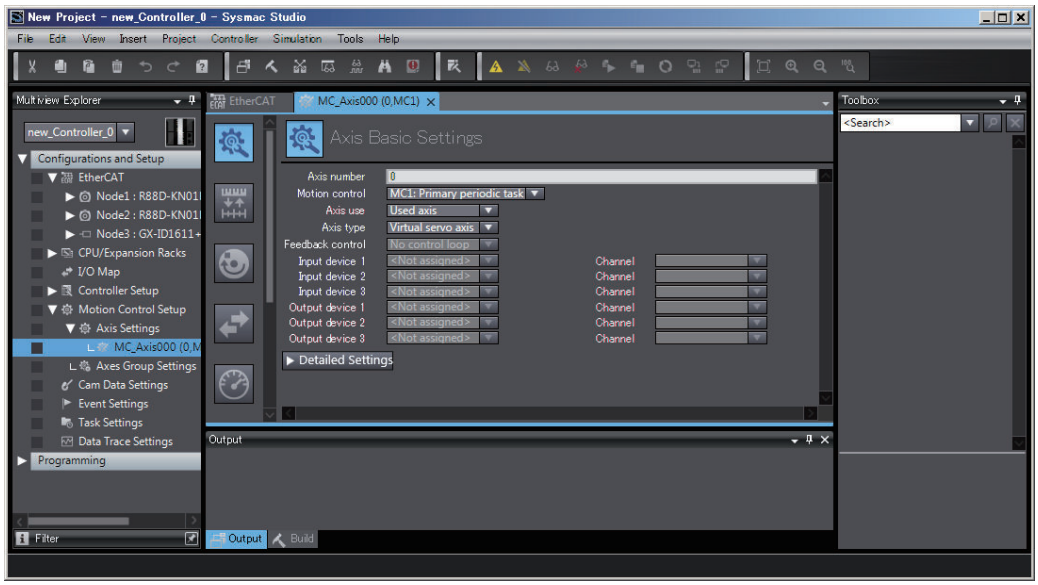


By default, device variables are registered in the global variable table. If necessary, you can change the variable type from a global variable to a local variable (internal variable) for a POU.

Create Axes (Axis Variables)

1 Right-click **Axis Settings** under **Configurations and Setups** - **Motion Control Setup** and select **Add** - **Axis Settings** from the menu.

2 Assign Servo Drives to the axes (axis variables) that you created in the EtherCAT configuration.



- Set the *Axis Use* parameter to *Used Axis*.
- For an NX-series CPU Unit, set the *Motion Control* parameter to *Primary periodic task*.
- Set the *Axis Type* parameter to *Servo Axis*.
- Set the *Input Device* and *Output Device* parameters to the EtherCAT slaves that you registered in the slave configuration.

Set the other parameters, such as the Unit Conversion Settings and Operation Settings.

Set the Controller Setup and the Special Unit Setup

- **Initial Settings for the PLC Function Module:**

  The Controller Setup includes the Startup Mode and other parameters.

- **Initial Settings for Special Units:**

  Unit Configuration and Setup: Set the initial settings of the Analog Input Unit.

11-2-4 Programming with the Sysmac Studio

On the Sysmac Studio, create the programs, set the tasks, and build the project.

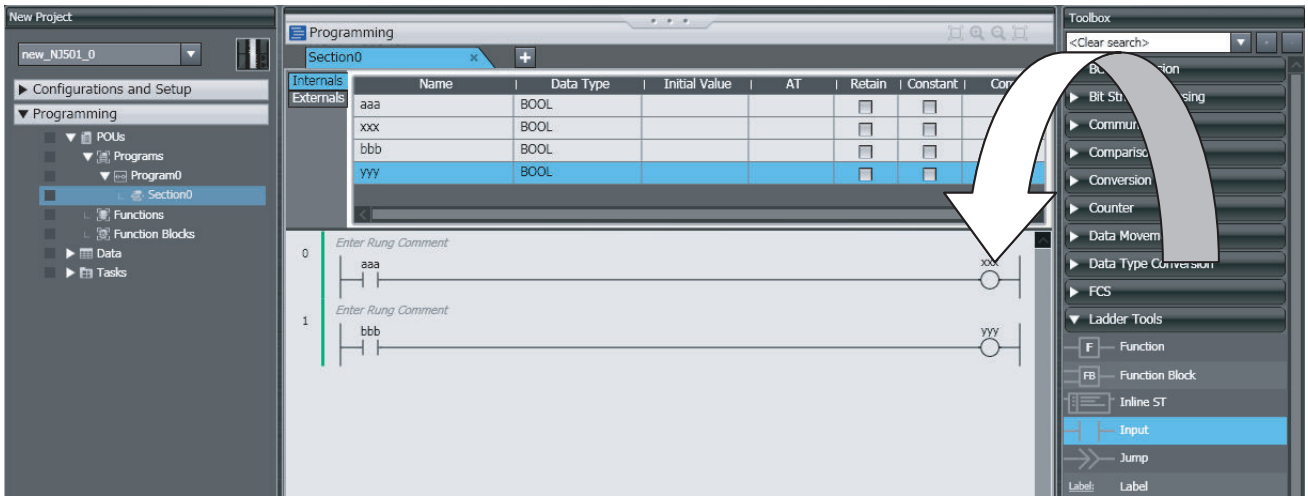
Write the Programs

1 Right-click **Programs** under **Programming** - **POU** and select **Add** - **Ladder** or **Add** - **ST** from the menu.

2 Double-click **Section**☐ under the program that you registered.

3 Register the local variables for each program.

4 Enter the programs.



Create a program with the following instructions.

- Homing: MC_Home instruction
- Velocity control: MC_MoveVelocity instruction
- Interrupt feeding: MC_MoveFeed instruction
- Positioning: MC_Move instruction

5 As required, right-click **Functions** or **Function Blocks** under **Programming** - **POU** and select **Add** - **Ladder** or **Add** - **ST** from the menu.

Double-click the **Function☐** or **FunctionBlock☐** that you registered. Register local variables for each function and function block. Create the algorithms.

Note For a ladder diagram, press the **R** Key and create the following rungs.

Set Up the Tasks

Double-click **Task Settings** under **Configurations and Setup**.

- In the **Task Settings**, set the task period and execution condition for the primary periodic task from the pulldown list.
- In the **I/O Control Task Settings**, select the task name to which to assign each Unit and slave.
- In the **Program Assignment Settings**, assign the programs to the primary periodic task or the priority-16 periodic task.

Build the Project

Select **Build** from the Project Menu.

11-2-5 Simulation with the Sysmac Studio

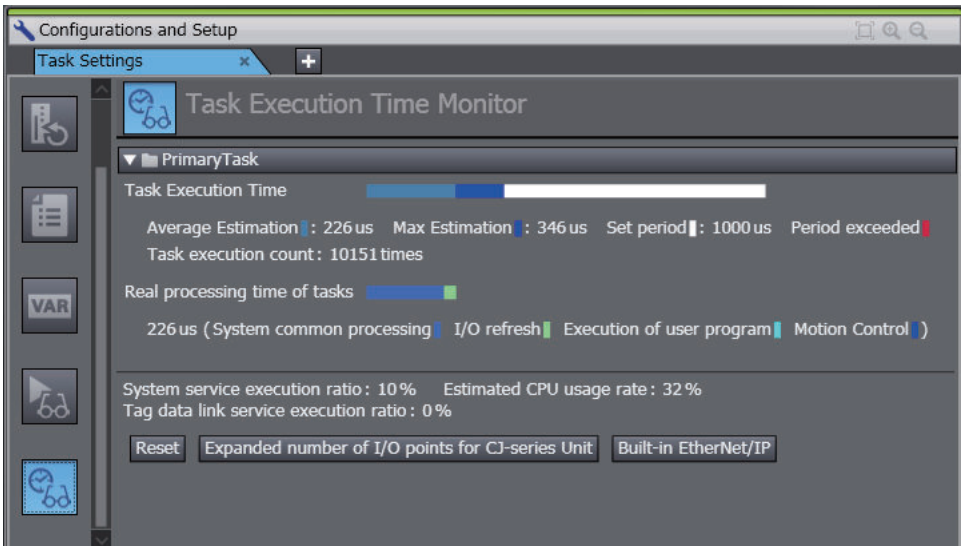
Simulation is used to perform desktop debugging.
Check the task execution times and the real processing times of tasks. Review the task design as required.

Starting the Simulator and Connecting to It

Select **Run in Execution Time Estimation Mode** from the Simulation Menu.
Select the relevant hardware revision in the Unit that the hardware revision is displayed.
The Simulator (i.e., the virtual Controller) starts. An online connection is created automatically.

Checking the Task Execution Time on the Simulator

Double-click **Task Settings** under **Configurations and Setup**.
Check to see if the task execution times in the **Task Execution Time Monitor** exceed the task periods.



If necessary, review the task configuration, program assignments, and task periods.

Saving the Project

Select **Save As** from the File Menu.

11-2-6 Checking Operation and Starting Operation on the Actual System

Go online with the Controller, download the project, check the wiring and perform test operation before you start actual operation.

Going Online

1. Turn ON the power supply to NJ-series Controller.
2. Connect the computer and the CPU Unit with a USB cable.
3. Select **Communications Setup** from the Controller Menu. Select the connection method for the connection configuration in the **Connection Type** Field.
4. Select **Online** from the Controller Menu.

Downloading the Project with the Synchronize Menu

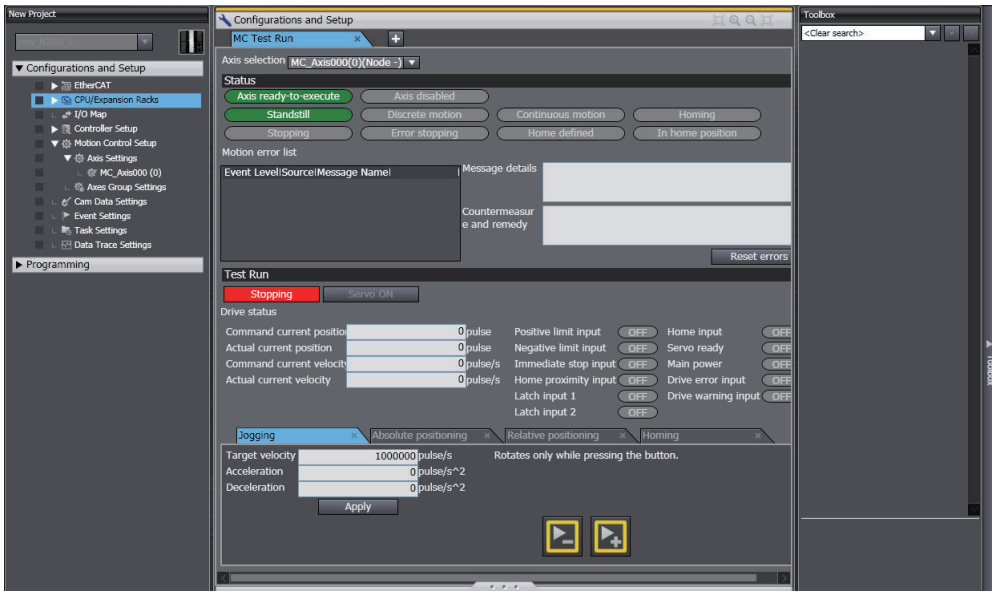
Select **Synchronize** from the Controller Menu and download the project to the Controller.
Note Use the Synchronize Menu of the Sysmac Studio to upload and download the project.

Checking Wiring

Check the wiring by using forced refreshing of real I/O from the I/O Map or Ladder Editor.

MC Test Run

1. Open the MC Test Run Tab Page.
2. Change the CPU Unit to PROGRAM mode.
3. Monitor input signals on the display to check the wiring.
4. Jog the axis from the display.



Manual Operation

Change the CPU Unit to RUN mode.

- Turning the Servo ON and OFF: Execute the MC_Power motion control instruction.
- Jogging: Execute the MC_MoveJog motion control instruction.

Homing

Homing: Execute the MC_Home instruction.

Actual Operation

Select **Operation Mode - RUN Mode** from the Controller Menu.

If an error occurs, investigate the cause and edit the user program.

12

Troubleshooting

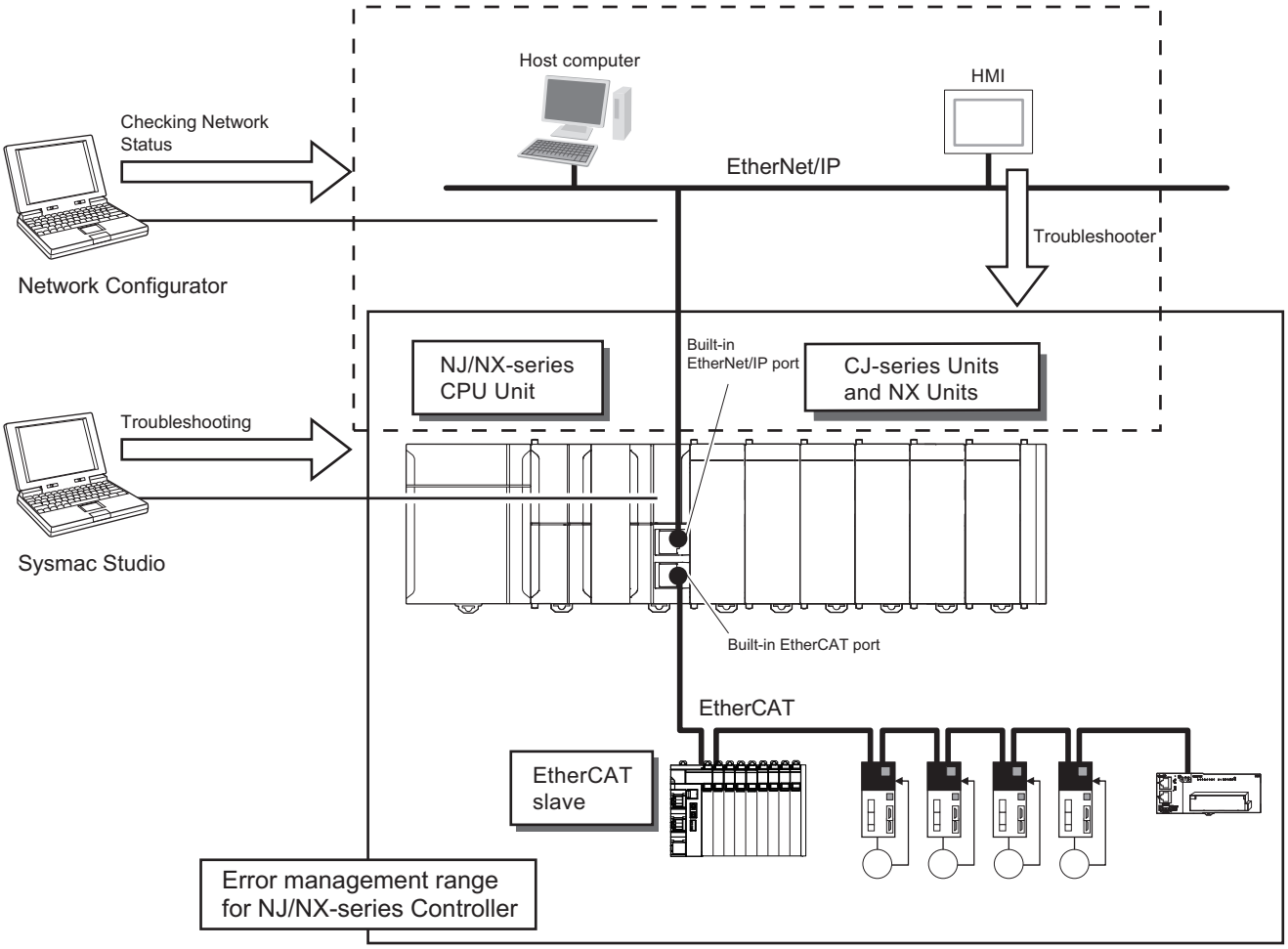
This section describes the overview of methods for checking errors.

12-1 Overview of Troubleshooting 12-2

12-1 Overview of Troubleshooting

You manage all of the errors that occur on the NJ/NX-series Controller as events.
This allows you to see what errors have occurred and find corrections for them with the same methods for the entire range of errors that is managed (i.e., CPU Unit, NX Units, NX-series Slave Terminals, EtherCAT slaves, *1 and CJ-series Units).

*1. Only Sysmac devices are supported.



You can use the troubleshooting functions of the Sysmac Studio or the Troubleshooter on an HMI to quickly check for errors that have occurred and find corrections for them.

Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for types of errors, meanings, specific corrections when errors occur and for troubleshooting information on the entire NJ/NX-series Controller.

A

Appendices

The appendices provide the CPU Unit specifications, real processing times of tasks, system-defined variable lists, and other supplemental information for the body of this manual.

A-1 Specifications A-4
A-1-1 General Specifications A-4
A-1-2 Performance Specifications A-4
A-1-3 Function Specifications A-18

A-2 Calculating Guidelines for the Real Processing Times of Tasks for the NX701 System A-30
A-2-1 Calculating the Average Real Processing Times of Tasks A-31
A-2-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-39

A-3 Calculating Guidelines for the Real Processing Times of Tasks for the NX502 System A-42
A-3-1 Calculating the Average Real Processing Times of Tasks A-42
A-3-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-49

A-4 Calculating Guidelines for the Real Processing Times of Tasks for the NX102 System A-53
A-4-1 Calculating the Average Real Processing Times of Tasks A-54
A-4-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-61

A-5 Calculating Guidelines for the Real Processing Times of Tasks for the NX1P2 System A-64
A-5-1 Calculating the Average Real Processing Times of Tasks A-65
A-5-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-72

A-6 Calculating Guidelines for the Real Processing Times of Tasks for the NJ-series System A-75
A-6-1 Calculating the Average Real Processing Times of Tasks A-76
A-6-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period A-87

A-7 System-defined Variables A-91
A-7-1 System-defined Variables for the Overall NJ/NX-series Controller (No Category) A-92
A-7-2 PLC Function Module, Category Name: _PLC A-102
A-7-3 PLC Function Module, Category Name: _CJB A-107

A-7-4 PLC Function Module, Category Name: _APB A-110
A-7-5 NX Bus Function Module, Category Name: _NXB A-111
A-7-6 X Bus Function Module, Category Name: _XBU A-114
A-7-7 Motion Control Function Module, Category Name: _MC A-117
A-7-8 EtherCAT Master Function Module, Category Name: _EC A-119
A-7-9 EtherNet/IP Function Module, Category Name: _EIP A-126
A-7-10 Meanings of Error Status Bits A-161

A-8 Specifications for Individual System-defined Variables A-163
A-8-1 System-defined Variables for the Overall NJ/NX-series Controller (No Category) A-163
A-8-2 PLC Function Module, Category Name: _PLC A-183
A-8-3 PLC Function Module, Category Name: _CJB A-189
A-8-4 PLC Function Module, Category Name: _APB A-193
A-8-5 NX Bus Function Module, Category Name: _NXB A-195
A-8-6 X Bus Function Module, Category Name: _XBU A-199
A-8-7 Motion Control Function Module, Category Name: _MC A-201
A-8-8 EtherCAT Master Function Module, Category Name: _EC A-204
A-8-9 EtherNet/IP Function Module, Category Name: _EIP A-214

A-9 Attributes of CPU Unit Data A-241

A-10 Contents of Memory Used for CJ-series Units A-247
A-10-1 CIO Area A-247
A-10-2 Internal I/O Area A-250
A-10-3 Holding Area A-250
A-10-4 DM Area A-250
A-10-5 EM Area A-251

A-11 Variable Memory Allocation Methods A-252
A-11-1 Variable Memory Allocation Rules A-252
A-11-2 Important Case Examples A-261

A-12 Registering a Symbol Table on the CX-Designer A-265

A-13 Enable/Disable EtherCAT Slave and Axes A-268
A-13-1 Project Settings When Using EtherCAT Slaves and Axes A-268
A-13-2 Using Instructions to Enable/Disable EtherCAT Slaves and Axes A-268
A-13-3 System-defined Variables That Indicate EtherCAT Slave or Axis Status A-269
A-13-4 Enabling/Disabling Execution of Program A-270
A-13-5 Checking Enabled/Disabled Program A-270
A-13-6 Settings with the Sysmac Studio A-271
A-13-7 Examples of Applications of Enabling/Disabling EtherCAT Slaves and Axes A-272

A-14 Size Restrictions for the User Program A-275
A-14-1 User Program Object Restrictions A-275
A-14-2 Counting User Program Objects A-279

A-15 Replacing CPU Units with Unit Version 1.02 or Earlier A-281
A-15-1 Uploading the Data from the CPU Unit A-281
A-15-2 Connecting the New CPU Unit A-284
A-15-3 Downloading the Data to the CPU Unit A-285

A-16 Possible Combination of Windows Versions and CPU Unit Versions . A-288

A-17 Device Variables A-289
A-17-1 X Bus Unit Common Function Module A-289

A-18 Specifications for Individual Device Variables A-291
A-18-1 X Bus Unit Common Function Module A-291

A-19 Version Information for NX-series Controllers A-293
A-19-1 Relationship between Unit Versions of CPU Units and Sysmac Studio Versions A-293
A-19-2 Functions That Were Added or Changed for Each Unit Version A-296

A-20 Version Information for NJ-series Controllers A-298
A-20-1 Relationship between Unit Versions of CPU Units and Sysmac Studio Versions A-298

A-20-2 Functions That Were Added or Changed for Each Unit Version A-300
A-20-3 Performance Improvements for Unit Version Upgrades A-303

A-1 Specifications

This section gives the specifications of the NJ/NX-series Controllers.

A-1-1 General Specifications

Refer to the following hardware manuals for general specifications.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)*

A-1-2 Performance Specifications

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Processing time	Instruction execution times	LD instruction		0.37 ns or more	0.53 ns	3.3 ns		1.2 ns (1.9 ns or less)*1	2.0 ns (3.0 ns or less)*2	3.3 ns (5.0 ns or less)*3
		Math instructions (for long real data)		3.2 ns or more	3.3 ns	70 ns or more		26 ns or more*4	42 ns or more*5	70 ns or more*6
Programming	Program capacity*7	Size		80 MB		5 MB	1.5 MB*8	20 MB	5 MB	3 MB
		Quantity	Number of POU definitions	6,000		3,000	450	3,000	750	450
			Number of POU instances	48,000		9,000	1,800	9,000 (*)	3,000 (*)	1,800
	Memory capacity for variables	Retain attributes*9	Size	4 MB	4MB *10	1.5MB*10	32 KB*10	2 MB	0.5 MB	
			Number of variables	40,000		10,000	5,000	10,000	5,000 (*)	
		No Retain attributes*11	Size	256 MB	256MB *10	32MB*10	2MB*10	4 MB	2 MB	
			Number of variables	360,000		90,000	90,000	180,000 (*)	90,000 (*)	22,500
	Data types	Number of data types		8,000		1,000	1,000	2,000	1,000	

Item			NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	Memory for CJ-series Units (Can be specified with AT specifications for variables.)	CIO Area	---	6,144 words (CIO 0 to CIO 6143)*12			6,144 words (CIO 0 to CIO 6143)		
		Work Area	---	512 words (W0 to W511)*12			512 words (W0 to W511)		
		Holding Area	---	1,536 words (H0 to H1535)*13			1,536 words (H0 to H1535)		
		DM Area	---	32,768 words (D0 to D32767)*13		16,000 words (D0 to D15999)*13	32,768 words (D0 to D32767)		
		EM Area	---	32,768 words × 25 banks (E0_00000 to E18_32767)*13		---	32,768 words × 25 banks (E0_00000 to E18_32767)	32,768 words × 4 banks (E0_00000 to E3_32767)	
Motion control			Refer to *Performance Specifications of Motion Control for Each Type of CPU Units* on page A-13.						
Peripheral (USB) port	Supported services		---*14	---	---	---	Sysmac Studio connection		
	Physical layer		---*14	---	---	---	USB 2.0-compliant B-type connector		
	Transmission distance		---*14	---	---	---	5 m max.		
Built-in EtherNet/IP port	Number of ports		2	2	2	1			
	Physical layer		10BASE-T/100BASE-TX/1000BASE-T		10Base-T or 100Base-TX				
	Frame length		1,514 bytes max.						
	Media access method		CSMA/CD						
	Modulation		Baseband						
	Topology		Star form						
	Baud rate		1 Gbps (1000BASE-T)		100 Mbps (100BASE-TX)				
	Transmission media		Shielded twisted-pair (STP) cable, Category 5, 5e, or higher						
	Maximum transmission distance between Ethernet switch and node		100 m						
	Maximum number of cascade connections		There is no limitation when an Ethernet switch is used.						

Item			NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	CIP service: Tag data links (cyclic communications)	Maximum number of connections	256 per port 512 total	64 per port 128 total	32 per port 64 total	32			
		Packet interval*15	Can be set for each connection. 0.5 to 10,000 ms in 0.5-ms increments	Can be set for each connection. 1 to 10,000 ms in 1-ms increments	Can be set for each connection. 1 to 10,000 ms in 1-ms increments	Can be set for each connection. 2 to 10,000 ms in 1-ms increments	Can be set for each connection. 1 to 10,000 ms in 1-ms increments (*)		
		Permissible communications band	40,000 pps*16 (including heartbeat)	20,000 pps*16 (including heartbeat)	12,000 pps*16 (including heartbeat, CIP Safety routing)	3,000 pps*16 (including heartbeat) (*)			
		Maximum number of tag sets	256 per port 512 total	64 per port 128 total	32 per port 40 total*17	32			
		Tag types	Network variables	Network variables, CIO, Work, Holding, DM, and EM Areas		Network variables, CIO, Work, Holding, and DM Areas	Network variables, CIO, Work, Holding, DM, and EM Areas		
		Number of tags per connection (i.e., per tag set)	8 (7 tags if Controller status is included in the tag set.)	64 (63 tags if Controller status is included in the tag set.)	8 (7 tags if Controller status is included in the tag set.)				
		Maximum number of tags	256 per port 512 total			256			
		Maximum link data size per node (total size for all tags)	369,664 bytes per port 739,328 bytes total	92,416 bytes per port 184,832 bytes total	19,200 bytes per port 38,400 bytes total	19,200 bytes			
		Maximum data size per connection	1,444 bytes		600 bytes				
		Maximum number of registrable tag sets	256 per port 512 total (1 connection = 1 tag set)	64 per port 128 total (1 connection = 1 tag set)	32 per port 40 total*17 (1 connection = 1 tag set)	32 (1 connection = 1 tag set)			
		Maximum tag set size	1,444 bytes (Two bytes are used if Controller status is included in the tag set.)		600 bytes (Two bytes are used if Controller status is included in the tag set.)				

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
		Multicast packet filter*18		Supported						
	CIP message service: Explicit messages	Class 3 (number of connections)		128 per port 256 total (clients plus server)		32 per port 64 total (clients plus server)	32 (clients plus server)			
		UCMM (non-connection type)	Maximum number of clients that can communicate at one time	32 per port 64 total			32			
			Maximum number of servers that can communicate at one time	32 per port 64 total			32			
	CIP Safety routing (*)	Maximum number of routable CIP Safety connections		---	---	16 total	---			
		Maximum routable safety data length per connection		---	---	32 bytes	---			
	Number of TCP sockets			30 (*)	60		30 (*)			
	Secure Socket Service*19	Number of secure sockets		---	60		30	---	---	---
		TLS version		---	1.2			---	---	---
	OPC UA Server*20*21	Support Profile/ Model		Micro Embedded Device Server Profile PLCopen Information Model			---	Micro Embedded Device Server Profile PLCopen Information Model	---	
		Default Endpoint/ Port		opc.tcp://192.168.250.1:4840/			---	opc.tcp://192.168.250.1:4840/	---	
		Maximum number of sessions (Client)		5			---	5	---	
		Maximum number of Monitored items per server		2,000			---	2,000	---	

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
		Sampling Rate (ms) of the Monitored Items		0*22, 50, 100, 250, 500, 1,000, 2,000, 5,000, 10,000			---	0*22, 50, 100, 250, 500, 1,000, 2,000, 5,000, 10,000	---	
		Maximum number of subscriptions per server		100			---	100	---	
		Maximum number of variables that can be published		10,000			---	10,000	---	
		Maximum number of value attributes for a variable that can be published*23		10,000			---	10,000	---	
		Structure's definitions that can be published		100			---	100	---	
		Variables unable to publish		*24			---	*24	---	
		Security Policy/ Mode		*25			---	*25	---	
		Application Authentication	Authentication	X.509			---	X.509	---	
			Maximum number of certification	Trusted certification: 32 Issuer certification: 32 Rejection list: 32			---	Trusted certification: 32 Issuer certification: 32 Rejection list: 32	---	
		User Authentication	Authentication	You can set the following items. User name/Password Anonymous			---	You can set the following items. User name/ Password Anonymous	---	
	DB connection			---	*26	---	---	---	---	---
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	Line, daisy chain, branching, and ring*27					

Item		NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	Transmission media	Twisted-pair cable of category 5 or higher (double-shielded straight cable with aluminum tape and braiding)						
	Maximum transmission distance between nodes	100 m						
	Maximum number of slaves	512	256	64	16*28	192		64
	Range of node addresses that can be set	1 to 512	1 to 256	1 to 192				
	Maximum process data size	Inputs: 11,472 bytes Outputs: 11,472 bytes *29	Inputs: 11,472 bytes Outputs: 11,472 bytes	Inputs: 5,736 bytes Outputs: 5,736 bytes *30	Inputs: 1,434 bytes Outputs: 1,434 bytes *31	Inputs: 5,736 bytes Outputs: 5,736 bytes *30		
	Maximum process data size per slave	Inputs: 1,434 bytes Outputs: 1,434 bytes						
	Communications cycle	• Primary periodic task 125 µs 250 µs to 8 ms (in 250-µs increments) • Priority-5 periodic task 125 µs 250 µs to 100 ms (in 250-µs increments)	250 µs to 8 ms (in 250-µs increments)	1,000 µs to 32,000 µs (in 250-µs increments)	2,000 µs to 8,000 µs (in 250-µs increments)*32	500, 1,000, 2,000, or 4,000 µs (*)		1,000, 2,000, or 4,000 µs
	Sync jitter	1 µs max.						
Serial communications	**Communications method**	---	---	---	Half duplex (when connected to the Serial Communications Option Board)	---		
	Synchronization method	---	---	---	Start-stop synchronization (when connected to the Serial Communications Option Board)	---		

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	Baud rate			---	---	---	1.2/2.4/4.8/9.6/19.2/38.4/57.6/115.2 kbps (when connected to the Serial Communications Option Board)	---		
Unit configuration	Maximum number of connectable Units	Maximum number of CJ Units per CPU Rack or Expansion Rack		---				10		
		Maximum number of NX Units per CPU Rack or Expansion Rack		---	63	32	8	---		
		Maximum number of CJ Units for entire controller		---				40		
		Maximum number of NX Units for entire controller		4,096		432	24	4,096		400
		Maximum number of X Bus Units for entire controller		---	4	---				
	Maximum number of Expansion Racks			0				3		
	I/O capacity	Maximum number of I/O points on CJ-series Units		---				2,560		
	Power Supply Unit for CPU Rack and Expansion Racks	Model		NX-PA9001 NX-PD7001	A non-isolated power supply for DC input is built into the CPU Unit.			NJ-P□3001		
		Power OFF detection time	AC power supply	30 to 45 ms	---			30 to 45 ms		
			DC power supply	5 to 20 ms	2 to 4 ms	2 to 8 ms		22 to 25 ms		
	Maximum NX Bus I/O data size			---	Inputs: 8,192 bytes Outputs: 8,192 bytes			---		
Option Board	Number of slots			---			*33	---		

Item			NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Built-in I/O	Input	Number of points	---			*34	---		
	Output	Number of points	---			*35	---		
		Load short-circuit protection	---			*36	---		
Internal Clock	Accuracy*37		At ambient temperature of 55°C: Monthly difference -4.5 minutes to +4.5 minutes At ambient temperature of 25°C: Monthly difference -3.5 minutes to +3.5 minutes At ambient temperature of 0°C: Monthly difference -4.5 minutes to +4.5 minutes	At ambient temperature of 55°C: Monthly difference -4.0 minutes to +4.0 minutes At ambient temperature of 25°C: Monthly difference -2.5 minutes to +2.5 minutes At ambient temperature of 0°C: Monthly difference -4.0 minutes to +4.0 minutes	At ambient temperature of 55°C: Monthly difference -3.0 minutes to +2.0 minutes At ambient temperature of 25°C: Monthly difference -2.0 minutes to +2.0 minutes At ambient temperature of 0°C: Monthly difference -3.0 minutes to +2.0 minutes		At ambient temperature of 55°C: Monthly difference -4.5 minutes to +4.5 minutes At ambient temperature of 25°C: Monthly difference -3.5 minutes to +3.5 minutes At ambient temperature of 0°C: Monthly difference -4.5 minutes to +4.5 minutes		
	Retention time of built-in capacitor		---		At ambient temperature of 40°C: 10 days		---		

*1. When the hardware revision for the Unit is *A* or later, the processing time is *1.1 ns (1.7 ns or less)*.
*2. When the hardware revision for the Unit is *A* or later, the processing time is *1.6 ns (2.5 ns or less)*.
*3. When the hardware revision for the Unit is *A* or later, the processing time is *3.0 ns (4.5 ns or less)*.
*4. When the hardware revision for the Unit is *A* or later, the processing time is *24 ns or more*.
*5. When the hardware revision for the Unit is *A* or later, the processing time is *35 ns or more*.
*6. When the hardware revision for the Unit is *A* or later, the processing time is *63 ns or more*.
*7. Execution objects and variable tables (including variable names)
*8. The size of program capacity is 1.0 MB for an NX1P2-9B□□□□□ CPU Unit.
*9. Does not include Holding, DM, and EM Area memory for CJ-series Units.
*10. Memory for CJ-series Units is included.
*11. Does not include CIO and Work Area memory for CJ-series Units.
*12. Variables without a Retain attribute are used. The value can be set in 1-word increments.
*13. Variables with a Retain attribute are used. The value can be set in 1-word increments.
*14. An NX701 CPU Unit without hardware revision provides a peripheral (USB) port of the following specifications.
- Supported service: Sysmac Studio connection
- Physical layer: USB 2.0-compliant B-type connector
- Transmission distance: 5 m max.

*15. Data will be refreshed at the set interval, regardless of the number of nodes.
*16. “pps” means packets per second, i.e., the number of communications packets that can be sent or received in one second.
*17. If more than 40 tag sets are registered in total, the *Number of Tag Sets for Tag Data Links Exceeded (840E0000 hex)* event will occur.

*18. 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.
*19. An NX502 CPU Unit with unit version 1.60 or later, an NX102-□□00 CPU Unit with unit version 1.46 or later, an NX102-□□20 CPU Unit with unit version 1.37 or later, or an NX1P2 CPU Unit with unit version 1.46 or later is required.
*20. The CPU Unit that supports OPC UA in NJ-series is an NJ501-1□00 CPU Unit with unit version 1.17 or later.
*21. An NX701-□□□□ with unit version 1.24 or later supports OPC UA.
*22. If set to 0, it is assumed that is set to 50 ms.
*23. The number of value attributes is the value defined in the following formula.
Number of value attributes = (number of variables of base type) + (number of elements specified by array) + (number of values in the structure)
*24. The following variables cannot be published. For details on the following maximum values, refer to the *NJ/NX-series CPU Unit OPC UA User's Manual (Cat. No. W588)*.
- Variable whose size exceeds the maximum value
- Two-dimensional or higher structure arrays
- Structures that include two-dimensional and higher arrays
- Structures where nesting exceeds the maximum value
- Unions
- Arrays whose index number suffix does not start from 0
- Array whose number of elements exceeds the maximum value
- Structure with more than the maximum number of members

*25. The following can be selected.
- None
- Sign - Basic128Rsa15
- Sign - Basic256
- Sign - Basic256Sha256
- SignAndEncrypt - Basic128Rsa15
- SignAndEncrypt - Basic256
- SignAndEncrypt - Basic256Sha256

*26. For the specifications of DB connection, refer to the *NJ-series Database Connection CPU Units User's Manual (Cat. No. W527)*.
*27. A ring topology can be used for project unit version 1.40 or later.
*28. The maximum number of slaves is 8 for an NX1P2-9B□□□□□ CPU Unit.
*29. However, for project unit version earlier than 1.40, the data must be within eight frames.
*30. However, for project unit version earlier than 1.40, the data must be within four frames.
*31. However, for project unit version earlier than 1.40, the data must be within one frame.
*32. The communications cycle is 4,000 μs to 8,000 μs (in 250-μs increments) for an NX1P2-9B□□□□□ CPU Unit.
*33. NX1P2-□□40□□□: 2, NX1P2-□□24□□□: 1
*34. NX1P2-□□40□□□: 24, NX1P2-□□24□□□: 14
*35. NX1P2-□□40□□□: 16, NX1P2-□□24□□□: 10
*36. NX1P2-□□□□□□: Not provided (NPN), NX1P2-□□□□□□1: Provided (PNP)
*37. These are values in continuous operation state.

Note Items that are marked with asterisks in the table are improvements that were made during version upgrades. Refer to *A-19 Version Information for NX-series Controllers* on page A-293 and *A-20 Version Information for NJ-series Controllers* on page A-298 for information on version upgrades.

Performance Specifications of Motion Control for Each Type of CPU Units

Item			NX701-	
			17□□	16□□
Number of controlled axes*1	Maximum number of controlled axes		256 axes	128 axes
		Motion control axes	256 axes	128 axes
		Single-axis position control axes	---	
	Maximum number of used real axes		256 axes	128 axes
		Used motion control servo axes	256 axes	128 axes
		Used single-axis position control servo axes	---	
	Maximum number of axes for linear interpolation axis control		4 axes per axes group	
	Number of axes for circular interpolation axis control		2 axes per axes group	
Maximum number of axes groups			64 axes groups	
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%	

*1. Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for descriptions of axes.

Item			NX502-		
			15□□	14□□	13□□
Number of controlled axes*1	Maximum number of controlled axes		128 axes	64 axes	32 axes
		Motion control axes	128 axes	64 axes	32 axes
		Single-axis position control axes	---		
	Maximum number of used real axes		64 axes	32 axes	16 axes
		Used motion control servo axes	64 axes	32 axes	16 axes
		Used single-axis position control servo 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%		

*1. Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for descriptions of axes.

Item			NX102-			
			12□□	11□□	10□□	90□□
Number of controlled axes*1	Maximum number of controlled axes		15 axes			4 axes
		Motion control axes	11 axes			---
		Single-axis position control axes	4 axes			
	Maximum number of used real axes		12 axes	8 axes	6 axes	4 axes
		Used motion control servo axes	8 axes	4 axes	2 axes	---
		Used single-axis position control servo axes	4 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			8 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	262,140 points			---
	Maximum number of cam tables		160 tables			---
Position units			Pulse, mm, μm, nm, degree, and inch			
Override factors			0.00%, or 0.01% to 500.00%			

*1. Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for descriptions of axes.

Item			NX1P2-			
			11□□□□□	10□□□□□	90□□□□□	9B□□□□
Number of controlled axes*1	Maximum number of controlled axes		12 axes	10 axes	4 axes	2 axes
		Motion control axes	8 axes	6 axes	---	
		Single-axis position control axes	4 axes			2 axes
	Maximum number of used real axes		8 axes	6 axes	4 axes	2 axes
		Used motion control servo axes	4 axes	2 axes	---	
		Used single-axis position control servo axes	4 axes			2 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			8 axes groups		---	
Motion control period			Same as the period for primary periodic task			
Cams	Number of cam data points	Maximum points per cam table	65,535 points		---	
		Maximum points for all cam tables	262,140 points		---	
	Maximum number of cam tables		80 tables		---	
Position units			Pulse, mm, μm, nm, degree, and inch			
Override factors			0.00%, or 0.01% to 500.00%			

*1. Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for descriptions of axes.

Item			NJ501-		
			□5□□	□4□□	□3□□
Number of controlled axes*1	Maximum number of controlled axes		64 axes	32 axes	16 axes
		Motion control axes	64 axes	32 axes	16 axes
		Single-axis position control axes	---		
	Maximum number of used real axes		64 axes	32 axes	16 axes
		Used motion control servo axes	64 axes	32 axes	16 axes
		Used single-axis position control servo 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%		

*1. Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for descriptions of axes.

Item			NJ301-	
			12□□	11□□
Number of controlled axes*1	Maximum number of controlled axes		15 axes (*)	
		Motion control axes	15 axes (*)	
		Single-axis position control axes	---	
	Maximum number of used real axes		8 axes	4 axes
		Used motion control servo axes	8 axes	4 axes
		Used single-axis position control servo 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	262,140 points	
	Maximum number of cam tables		160 tables	
Position units			Pulse, mm, μm, nm, degree, and inch	
Override factors			0.00%, or 0.01% to 500.00%	

*1. Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for descriptions of axes.

Note Items that are marked with asterisks in the table are improvements that were made during version upgrades. Refer to *A-19 Version Information for NX-series Controllers* on page A-293 and *A-20 Version Information for NJ-series Controllers* on page A-298 for information on version upgrades.

Item			NJ101-	
			10□□	90□□
Number of controlled axes*1	Maximum number of controlled axes		6 axes	---
		Motion control axes	6 axes	---
		Single-axis position control axes	---	
	Maximum number of used real axes		2 axes	---
		Used motion control servo axes	2 axes	---
		Used single-axis position control servo 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	262,140 points	---
	Maximum number of cam tables		160 tables	---
Position units			Pulse, mm, μm, nm, degree, and inch	---
Override factors			0.00%, or 0.01% to 500.00%	---

*1. Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for descriptions of axes.

A-1-3 Function Specifications

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Tasks	Function			I/O refresh and the user program are executed in units that are called *tasks*. Tasks are used to specify execution conditions and execution priority.						
		Periodically executed tasks	Maximum number of primary periodic tasks	1						
			Maximum number of periodic tasks	4	3	2		3		
		Conditionally executed tasks (*)	Maximum number of event tasks	32						
			Execution conditions	When Activate Event Task instruction is executed or when condition expression for variable is met						
	Setup	System Service Monitoring Settings		---				The execution interval and the percentage of the total user program execution time are monitored for the system services (processes that are executed by the CPU Unit separate from task execution).		
Programming	POUs (program organization units)	Programs		POUs that are assigned to tasks.						
		Function blocks		POUs that are used to create objects with specific conditions.						
		Functions		POUs that are used to create an object that determine unique outputs for the inputs, such as for data processing.						
	Programming languages	Types		Ladder diagrams*1 and structured text (ST)						
	Namespaces (*)			Namespaces are used to create named groups of POU definitions.						
	Variables	External access of variables	Network variables	The function which allows access from the HMI, host computers, or other Controllers						

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	Data types	Basic data types	Boolean	BOOL						
			Bit strings	BYTE, WORD, DWORD, and LWORD						
			Integer	INT, SINT, DINT, LINT, UINT, USINT, UDINT, and ULINT						
			Real numbers	REAL and LREAL						
			Durations	TIME						
			Dates	DATE						
			Time of day	TIME_OF_DAY						
			Dates and times	DATE_AND_TIME						
			Text strings	STRING						
		Derivative data type		Structure data types, union data types, and enumeration data types						
		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.						
		Unions	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						
		Enumerations	Function	A derivative data type that uses text strings called *enumerators* to express variable values.						
	Data type attributes	Array specification	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	Possible.						
		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.						

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Motion Control*2	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				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
		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.						
			Slave axis position compensation (*)	This function compensates the position of the slave axis currently in synchronized control.						
			Cam monitor(*)	Outputs the specified offset position for the slave axis in synchronous control.						
			Start velocity (*)	You can set the initial velocity when axis motion starts.						

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	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				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	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 are specified in input parameters.						
		Parameters	Writing MC settings	Some of the axis parameters or axes group parameters are overwritten temporarily.						
			Changing axis parameters (*)	Some of the axis parameters can be accessed or changed from the user program.						
	Auxiliary functions	Count modes		You can select either Linear Mode (finite length) or Rotary Mode (infinite length).						
		Unit conversions		You can set the display unit for each axis according to the machine.						
		Acceleration/deceleration control	Automatic acceleration/deceleration control	Jerk is set for the acceleration/deceleration curve for an axis motion or axes group motion.						
			Changing the acceleration and deceleration rates	You can change the acceleration or deceleration rate even during acceleration or deceleration.						
		In-position check		You can set an in-position range and in-position check time to confirm when positioning is completed.						
		Stop method		You can set the stop method to the immediate stop input signal or limit input signal.						
		Re-execution of motion control instructions		You can change the input variables for a motion control instruction during execution and execute the instruction again to change the target values during operation.						
		Multi-execution of motion control instructions (Buffer Mode)		You can specify when to start execution and how to connect the velocities between operations when another motion control instruction is executed during operation.						
		Continuous axes group motions (Transition Mode)		You can specify the Transition Mode for multi-execution of instructions for axes group operation.						

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
		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, 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 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 given below are used. Home signal, home proximity signal, positive limit signal, negative limit signal, immediate stop signal, and interrupt input signal.						
Unit (I/O) management	EtherCAT slaves	Maximum number of slaves		512	256	64	16	192		64
	CJ-series Units	Maximum number of Units		---				40		
Communications	Peripheral (USB) port			---*3	---	---		A port for communications with various kinds of Support Software running on a personal computer.		
	EtherNet/IP port	Communications protocol		TCP/IP or UDP/IP						
		CIP communications services	Tag data links	Programless cyclic data exchange is performed with the devices on the EtherNet/IP network.						
			Message communications	CIP commands are sent to or received from the devices on the EtherNet/IP network.						

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
		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.						
			Secure socket service (Client)*4	---	The TLS session is established by using the TCP protocol, and any data is sent and received by the secure socket communications instruction, between the server and any node on Ethernet.			---	---	---
			FTP client (*)	Files are transferred via FTP from the CPU Unit to computers or Controllers at other Ethernet nodes. FTP client communications instructions are used.						
			FTP server	Files can be read from or written to the SD Memory Card in the CPU Unit from computers at other Ethernet nodes.						
			Automatic clock adjustment	Clock information is read from the NTP server at the specified time or at a specified interval after the power supply to the CPU Unit is turned ON. The internal clock time in the CPU Unit is updated with the read time.						
			SNMP agent	Built-in EtherNet/IP port internal status information is provided to network management software that uses an SNMP manager.						
		OPC UA*5*6	Server Function	Functions to respond to requests from clients on the OPC UA network			---	Functions to respond to requests from clients on the OPC UA network	---	
	EtherCAT port	Supported services	Process data communications	A communications method to exchange control information in cyclic communications between the EtherCAT master and slaves. This communication method is defined by CoE.						
			SDO communications	A communications method to exchange control information in noncyclic event communications between the EtherCAT master and slaves. This communication method is defined by CoE.						
		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*7		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.						

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	Serial communications	Protocol		---			Host link (FINS), no-protocol, and Modbus-RTU master (when connected to the Serial Communications Option Board)	---		
	Communications instructions			FTP client instructions, CIP communications instructions, socket communications instructions, SDO message instructions, no-protocol communications instructions, Modbus RTU protocol instructions(*), and Modbus TCP protocol instructions*8				FTP client instructions (*), CIP communications instructions, socket communications instructions, SDO message instructions, no-protocol communications instructions, protocol macro instructions, and Modbus RTU protocol instructions (*)		
Operation management	RUN output contacts			The output on the Power Supply Unit turns ON in RUN mode.	---			The output on the Power Supply Unit turns ON in RUN mode.		
System management	Event logs	Function		Events are recorded in the logs.						
		Maximum number of events	System event log	2,048	2,560*13	768*9	576*10	1,024	512	
			Access event log	1,024	1,152*14	576*11	528*12	1,024	512	
			User-defined event log	1,024	1,024	512	512	1,024	512	
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 monitored variables	Device variables for EtherCAT slaves	64						
			Device variables for CJ-series Units and variables with AT specifications	---				64		
	MC Test Run*2			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				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
	Data tracing	Types	Triggered traces	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		2	2	4	2	
		Maximum number of records		10,000						
		Maximum number of sampled variables		192 variables		48 variables	48 variables	192 variables	48 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.						
	Automation playback			---	A function that supports all three activities of system maintenance, recording, reproduction, and analysis, in an integrated manner.	---				
Reliability functions	Self-diagnosis	Controller errors	Level	Major faults, partial faults, minor faults, observation, and information						
			Maximum number of message languages	9 (Sysmac Studio) 2 (NS-series PT)	9 (Sysmac Studio) 2 (NA-series PT)	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.						
			Level	8						
			Maximum number of message languages	9						

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
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	Storage type			SD card or SDHC card						
	Application	Automatic transfer from SD Memory Card (*)		When the power supply to the Controller is turned ON, the data that is stored in the auto-load directory of the SD Memory Card is transferred to the Controller.						
		Program transfer from SD Memory Card (*)		With the specification of the system-defined variable, you can transfer a program that is stored in the SD Memory Card to the Controller.						
		SD Memory Card operation instructions		You can access SD Memory Cards from instructions in the user program.						
		File operations from the Sysmac Studio		You can perform file operations to save and read for Controller files in the SD Memory Card and general-purpose document files on the computer.						
		SD Memory Card life expiration detection		Notification of the expiration of the life of the SD Memory Card is provided in a system-defined variable and event log.						

Item				NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Backing up data (*)	SD Memory Card backups	Operating methods	CPU Unit front-panel DIP switch	Backup, verification, and restoration operations are performed by manipulating the front-panel DIP switch on the CPU Unit.						
			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 an SD Memory Card is prohibited.						
	Safety unit restore (*)			---	You can perform to restore data in a Safety CPU Unit with an SD Memory Card and the front-panel DIP switch on the Safety CPU Unit.		---			
	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. This function is not available for NJ101-90□□ CPU Units.
*3. An NX701 CPU Unit without hardware revision provides a peripheral (USB) port of the following specifications.
- Function: A port for communications with various kinds of Support Software running on a personal computer.

*4. An NX102-□□00 CPU Unit with unit version 1.46 or later, an NX102-□□20 CPU Unit with unit version 1.37 or later, or an NX1P2 CPU Unit with unit version 1.46 or later is required.
*5. The CPU Unit that supports OPC UA in NJ-series is an NJ501-1□00 CPU Unit with unit version 1.17 or later.
*6. An NX701-□□□□ with unit version 1.24 or later supports OPC UA.
*7. For project unit version 1.40 or later, packet monitoring cannot be used.
*8. Available only with NX102-□□□□ CPU Units.
*9. This is the total of 512 events for the CPU Unit and 256 events for the NX Unit.
*10. This is the total of 512 events for the CPU Unit and 64 events for the NX Unit.
*11. This is the total of 512 events for the CPU Unit and 64 events for the NX Unit.
*12. This is the total of 512 events for the CPU Unit and 16 events for the NX Unit.
*13. This is the total of 2048 events for the CPU Unit and 512 events for the NX Unit.
*14. This is the total of 1024 events for the CPU Unit and 128 events for the NX Unit.

Note Items that are marked with asterisks in the table were added for version upgrades. Refer to *A-19 Version Information for NX-series Controllers* on page A-293 and *A-20 Version Information for NJ-series Controllers* on page A-298 for information on version upgrades.

A-2 Calculating Guidelines for the Real Processing Times of Tasks for the NX701 System

This section describes how to calculate guidelines for the average real processing times of tasks on paper for the NX701 System.

You must use the physical Controller to check the real processing times of tasks and task execution times. Refer to *5-12 Task Design Methods and I/O Response Times* on page 5-133 for details.

Precautions for Safe Use

The task execution times in the physical Controller depend on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors.
Before starting actual operation, you must test performance under all foreseeable conditions on the actual system and make sure that the task periods are not exceeded and that suitable communications performance is achieved.

Precautions for Correct Use

- Calculation of the average real processing times of tasks for the priority-5 periodic task will require the value of the **PDO Communications Cycle 2: Transmission Delay Time**, which is displayed in the EtherCAT Tab Page after the EtherCAT configuration is created with the Sysmac Studio.
- Processing performance of I/O refresh, user program, and motion control has been improved in the hardware revision *B* or later, so the actual processing time may be shorter.

Additional Information

Periodic tasks may be interrupted for the execution of tasks with higher execution priorities. The real processing time of a task does not include the time for which the task is interrupted. It is the task execution time that gives the actual time from when the task is started until it is finished, including the interrupted time. For a detailed description of the differences between the real processing times of tasks and the task execution times, refer to *Meaning of the Task Execution Time and the Real Processing Time of the Task* on page 5-132.

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.

○: Performed, ---: Not performed.

Processing		Processing contents	Primary periodic task	Priority-5 periodic task	Priority-16 periodic task	Priority-17 and priority-18 periodic tasks
I/O refresh processing		I/O is refreshed for EtherCAT slaves.	○	○	---	---
User program execution		Programs assigned to tasks are executed in the order that they are assigned.	○	○	○	○
Motion control processing		• Motion control commands from the user program are executed. • Motion output processing	○	○	---	---
Common processing time	System common processing 1	• Variable refresh processing (if there are accessing tasks) is performed. • Motion input processing • Data trace processing	○	○	○	○
	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	○	○	○	○
	System overhead time	Other system common processing	○	○	○	○

Guidelines are provided below for calculating the various processing times.

I/O Refresh Processing Time

Use the following formula for the I/O refresh processing time.

I/O refresh processing time = EtherCAT slave processing time

The formula for calculating the EtherCAT slave processing time is different between the primary periodic task and the priority-5 periodic task.

The following describes how to determine the EtherCAT slave processing time for each type of tasks.

Precautions for Correct Use

- When calculating for the priority-5 periodic task, you need to determine the task period of the primary periodic task in advance.
- Performance of I/O refresh execution has been improved in the hardware revision *B* or later, so the actual processing time may be shorter.

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 for the EtherCAT slave processing time in the primary periodic task.

EtherCAT slave processing time [μs] = 0.0006 × pDout + 0.0001 × pDin + 0.082 × pDinout
+ (1.24 × Snum + 0.01 × Clen) + 25.58

pDout	:	Total output processing data size [byte] of EtherCAT slaves assigned to the primary periodic task or the priority-16 periodic task
pDin	:	Total input processing data size [byte] of EtherCAT slaves assigned to the primary periodic task or the priority-16 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 or the priority-16 periodic task [byte]
Snum	:	Total number of EtherCAT slaves connected to the built-in EtherCAT port
Clen	:	Total length of cables connected to the built-in EtherCAT port [m]

● EtherCAT Slave Processing Time in Priority-5 Periodic Task

Use the following formula for the EtherCAT slave processing time in the priority-5 periodic task.

EtherCAT slave processing time [μs] = 0.0006 × pDout + (Larger of the following A and B)
+ sTsend + 0.0001 × sDin + 16

A [μs] = ((0.0623 × sDout + 37) × pTcycle) ÷ (pTcycle - (0.0243 × pDout + 35))
B [μs] = 0.082 × pDinout

pDout	:	Total output processing data size [byte] of EtherCAT slaves assigned to the primary periodic task
sTsend*1	:	Transmission delay time for PDO communications cycle 2 [μs] displayed on the Sysmac Studio
sDin	:	Total input processing data size [byte] of EtherCAT slaves assigned to the priority-5 periodic task
sDout	:	Total output processing data size [byte] of EtherCAT slaves assigned to the priority-5 periodic task
pTcycle	:	Task period of the primary periodic task [μs]
pDinout	:	Total of the larger of the input and output processing data size of each EtherCAT slave assigned to the primary periodic task [byte]

*1. The value of sTsend (transmission delay time) cannot be calculated on paper. Assign the value of the **PDO Communications Cycle 2: Transmission Delay Time**, which is displayed in the EtherCAT Tab Page after the EtherCAT configuration is created with the Sysmac Studio, to sTsend (transmission delay time). Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for how to display the **PDO Communications Cycle 2: Transmission Delay Time** on the Sysmac Studio.

User Program Execution Time

The user program execution time depends on the specific instructions multiplied by the numbers of instructions used.

As a guideline, instructions are divided into three groups and the number of instructions in each group is used for measurements and estimates.

- Standard instructions
- Arithmetic instructions for LREAL data
- Trigonometric instructions for LREAL data

Different instructions are used in a ladder diagram and in ST. Refer to *Instruction Configuration for Standard Ladder Diagram Instructions* on page A-34 and *Instruction Configuration for Standard ST Instructions* on page A-36 for information on the instruction configuration.

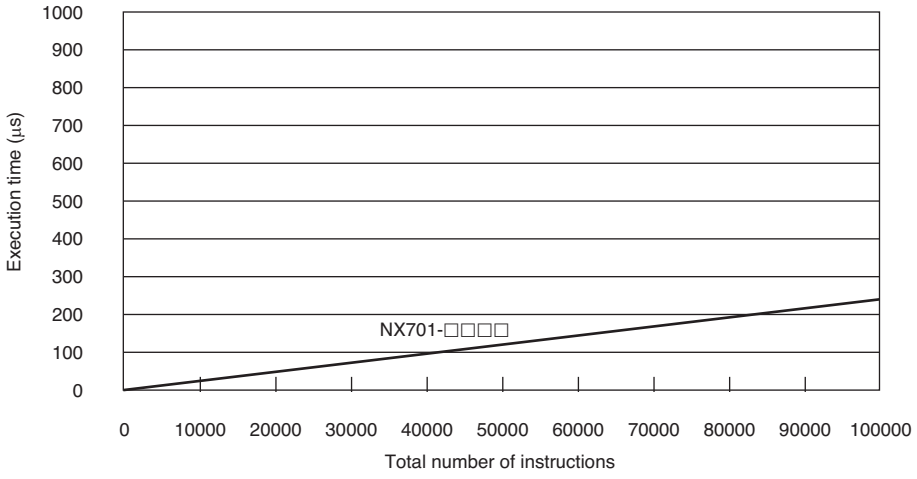
● Simple Estimate

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



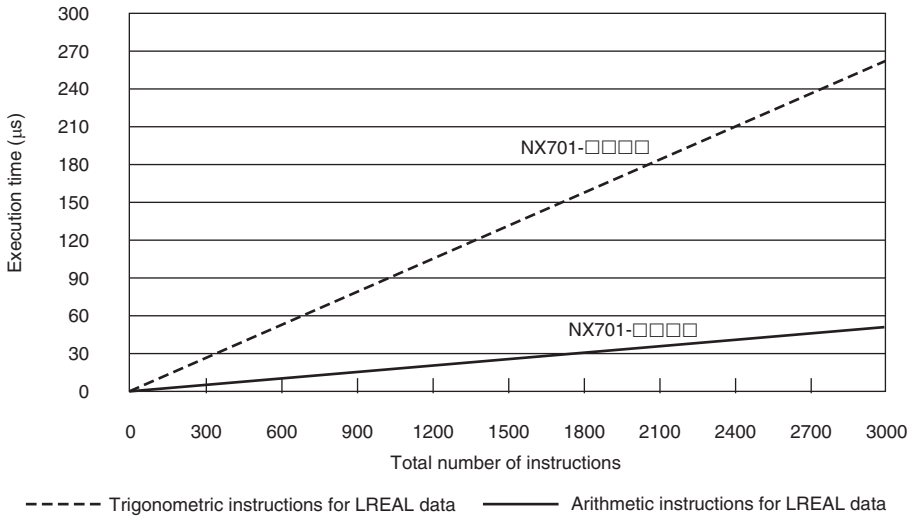
- Instruction Configuration for Standard Ladder Diagram Instructions

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

Types of instructions	Instructions	Percent of instructions [%]	Percent of execution time in instruction group [%]
Timer and counter instructions	Timer, TON/TOF, and CTU/CTD	1.6	7.3
Math instructions	+, -, *, /, ADD, SUB, MUL, and DIV	2.4	6.5
BCD conversion instructions and data conversion instructions	INT_TO_DINT and WORD_BCD_TO_UINT	0.2	1.2
Bit string processing instructions	AND and OR	6.2	13.0
Data movement instructions	MOVE	4.6	23.5
Total		100.0	100.0

Execution Times for Ladder Diagram Arithmetic and Trigonometric Instructions for LREAL Data



- Structure of *Arithmetic instructions for LREAL data*

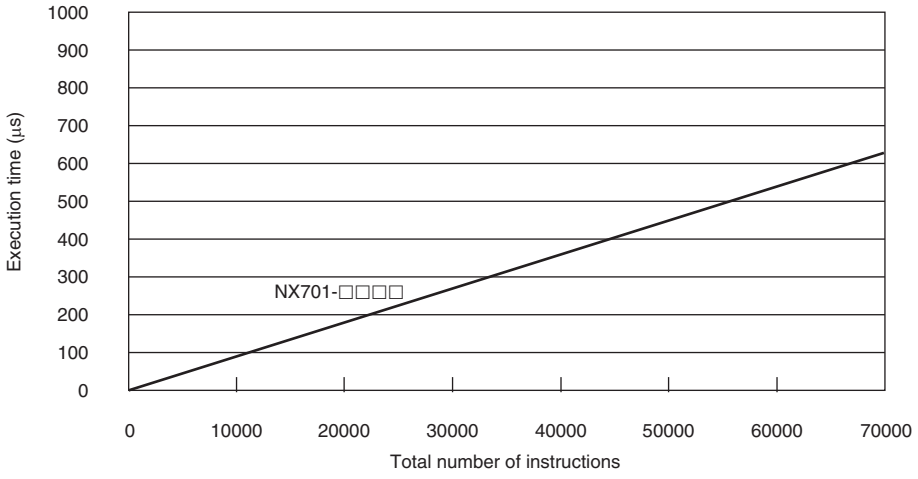
Instructions	Percent of instructions [%]
Addition instructions for LREAL data	20.0
Subtraction instructions for LREAL data	20.0
Multiplication instructions for LREAL data	30.0
Division instructions for LREAL data	30.0
Total	100.0

- Structure 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

Instructions	Percent of instructions [%]
Cos^{-1} of LREAL data	16.7
Tan^{-1} of LREAL data	16.7
Total	100.0

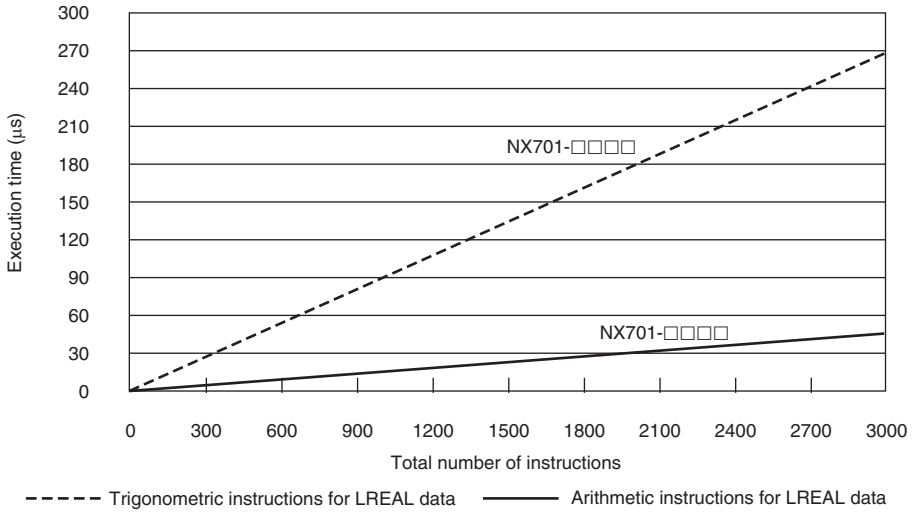
Execution Time for Standard ST Instructions



- 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



- Structure 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

- Structure 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

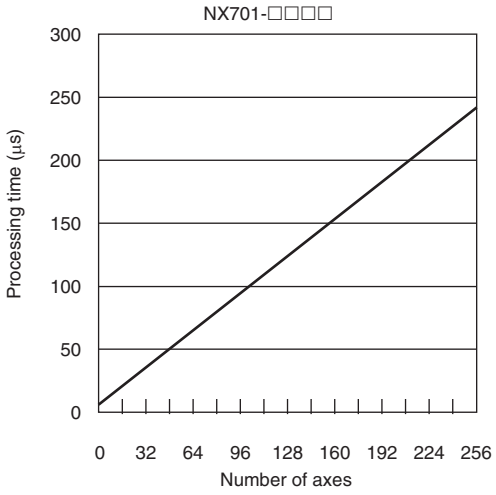
Precautions for Correct Use

Performance of user program execution has been improved in the hardware revision *B* or later, so the actual execution time may be shorter.

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.



Precautions for Correct Use

Processing performance of motion control has been improved in the hardware revision *B* or later, so the actual processing time may be shorter.

Common Processing Time

The common processing time is the following values by the total time for system overhead, system common processing 1, and system common processing 2. The common processing time depends on the type of task.

Type of task	Common processing times [µs] (reference values)
	NX701-□□□□
Primary periodic task	15
Priority-5 periodic task	55
Priority-16, 17, or 18 periodic task	10

A-2-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period

Example of Calculating the Average Real Processing Times of Tasks

If you are using an NX701-□□□□ CPU Unit with a unit version of 1.10, first find the average real processing time of the task for the following conditions.
The task is the primary periodic task.

Item		Conditions	
Slaves/Units that are used	**EtherCAT slaves**	Input slave	GX-ID1611 (Ver. 1.1): 1
		Output slave	GX-OD1611 (Ver. 1.1): 1
		Servo Drives	R88D-1SN□□□-ECT: 4
		EtherCAT Slave Terminal: 1	
		EtherCAT Coupler Unit	NX-ECC203: 1
		DC Input Unit	NX-ID5342: 1
		Transistor Output Unit	NX-OD5121: 1
		Analog Input Unit	NX-AD3608: 1
		Analog Output Unit	NX-DA2605: 1
		Communications Interface Units	NX-CIF101: 1 NX-CIF105: 1
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	pDout: Output processing data size in bytes	pDin: Input processing data size in bytes	pDinout: Larger of the input and output data size
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 × 4 + 62 = 156 [byte]

pDin = 3 + 1 + 26 × 4 + 88 = 196 [byte]

pDinout = 3 + 2 + 26 × 4 + 88 = 197 [byte]

From these values, the I/O refresh time is calculated by the following formula.

I/O refresh processing time

= EtherCAT slave processing time

= 0.0006 × pDout + 0.0001 × pDin + 0.082 × pDinout + (1.24 × Snum + 0.01 × Clen) + 25.58

= 0.0006 × 156 + 0.0001 × 196 + 0.082 × 197 + (1.24 × 7 + 0.01 × 10) + 25.58

= 0.0936 + 0.0196 + 16.154 + (8.68 + 0.1) + 25.58

≈ 51 [μs]

● User Program Execution Time

The graphs show the following values.

- Standard instruction configuration
  From the graph of the execution time for standard ladder diagram instructions, the user program execution time of 5,000 instructions for the NX701-□□□□ is 12 μs.
- Arithmetic instructions for LREAL data
  From the graph of the execution time for ladder diagram arithmetic and trigonometric instructions for LREAL data, the user program execution time of 200 instructions for the NX701-□□□□ is 4 μs.
- Trigonometric instructions for LREAL data
  From the graph of the execution time for ladder diagram arithmetic and trigonometric instructions for LREAL data, the user program execution time of 100 instructions for the NX701-□□□□ is 9 μs.

Therefore, the user program execution time is the total of the above values, which is given by the following formula.

User program execution time = 12+ 4 + 9

= 25 [μs]

● Motion Control Processing Time

From the graph of the execution time for motion control processing, the execution time of the motion control processing for four axes for the NX701-□□□□ with a unit version 1.10 is read as 10 μs.

● Common Processing Time

Because the task is the primary periodic task, the common processing time for the NX701-□□□□ is 15 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
=51 + 25 + 10 + 15
= 101 [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 x 1.1

Because the average real processing time of the task that is calculated above is 101 μs, the task period is set to 125 μs, which is larger than 1.1 times the average time.

The task execution times in the physical Controller depend on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors. The task execution time for a periodic task depends on whether it is interrupted for the execution of tasks with higher execution priorities.
Use the physical Controller and verify the task execution time with the Task Execution Time Monitor.

A-3 Calculating Guidelines for the Real Processing Times of Tasks for the NX502 System

This section describes how to calculate guidelines for the average real processing times of tasks on paper for the NX502 System.

You must use the physical Controller to check the real processing times of tasks and task execution times. Refer to *5-12 Task Design Methods and I/O Response Times* on page 5-133 for details.

Precautions for Safe Use

The task execution times in the physical Controller depend on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors.
Before starting actual operation, you must test performance under all foreseeable conditions on the actual system and make sure that the task periods are not exceeded and that suitable communications performance is achieved.

Additional Information

Periodic tasks will be interrupted for the execution of tasks with higher execution priorities. The *real processing time of a task* does not include the time for which the task is interrupted. It is the *task execution time* that gives the actual time from when the task is started until it is finished, including the interrupted time. For a detailed description of the differences between the *real processing times of tasks* and the *task execution times*, refer to *Meaning of the Task Execution Time and the Real Processing Time of the Task* on page 5-132.

A-3-1 Calculating the Average Real Processing Times of Tasks

The average real processing time of a task is the total of the I/O refresh processing time, user program execution time, motion control processing time and common processing time.

Average real processing time of task = I/O refresh processing time + User program execution time + Motion control processing time + Common processing time

The following processing is performed.

○: Performed, ---: Not performed.

Processing	Processing contents	Primary periodic task	Priority-16 periodic task	Priority-17 and priority-18 periodic task
I/O refresh processing	I/O is refreshed for NX Units on the CPU Unit, X Bus Units, and EtherCAT slaves.	○	---	---
User program execution	Programs assigned to tasks are executed in the order that they are assigned.	○	○	○

Processing		Processing contents	Primary periodic task	Priority-16 periodic task	Priority-17 and priority-18 periodic task
Motion control processing		• Motion control commands from the user program are executed. • Motion output processing	○	---	---
Common processing time	System common processing 1	• Variable refresh processing (if there are accessing tasks) is performed. • Motion input processing • Data trace processing	○	○	○
	System common processing 2	• Variable refresh processing (if there are tasks that update a variable) is performed. • Variable access processing external to the Controller to ensure concurrency with task execution	○	○	○
	System overhead time	Other system common processing	○	○	○

Guidelines are provided below for calculating the various processing times.

I/O Refresh Processing Time

Use the following formula to calculate the I/O refresh processing time.

I/O refresh processing time = The largest value of EtherCAT slave processing time, NX Unit processing time, and X Bus Unit processing time

The following describes how to determine the EtherCAT slave processing time, NX Unit processing time, and X Bus Unit processing time used in the above formula.

● EtherCAT Slave Processing Time

Use the following formula for the EtherCAT slave processing time.

EtherCAT slave processing time [μs] = 0.002 x pDout + 0.0014 x pDin + 0.082 x pDinout
+ (1.60 x Snum + 0.01 x Clen) + 66.964

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]

Additional Information

The EtherCAT slave processing time is *0* if EtherCAT slaves are not connected.

● NX Unit Processing Time

Use the following formula for the NX Unit processing time.
The calculation must include all NX Units on the CPU Unit.

NX Unit processing time [μs] = 0.152 x Dnxout + 0.053 x Dnxin + 0.549 x Dnxnum + 50.501

Dnxout	:	: Total output data size [byte]*1
Dnxin	:	: Total input data size [byte]*1
Dnxnum	:	Number of connected NX Units

*1. If the data size of an NX Unit is less than 2 bytes, calculate as 2 bytes.

Additional Information

The NX Unit processing time is *0* if NX Units are not connected to the CPU Unit.

● X Bus Unit Processing Time

Use the following formula for the X Bus Unit processing time.
The calculation must include all X Bus Units.
The processing times of tag data link communications and CIP Safety communications used with an NX-series EtherNet/IP Unit are calculated as parameters.

X Bus Unit processing time [μs] = CIP Safety communications processing time + Tag data link communications processing time + 2.25 x Number of connected X Bus Units + 36.18

CIP Safety communications processing time [μs] = 0.290 x Number of connections for CIP Safety communications

Tag data link communications processing time is the total communications processing time for each connection between the originator and target in tag data link communications for the NX-series EtherNet/IP Unit.
Communications processing time per connection [μs] = (0.9 + 0.0035 x DLSize) × (Primary period/DLRPI)
If the result of the calculation of (Primary period/DLRPI) is greater than 1, calculate it as (Primary period/DLRPI) = 1. Also, when calculating, the units of the primary period and DLRPI must be the same.

DLSize	:	Data size of the connection for tag data link communications [byte]
DLRPI	:	Packet interval of the connection for tag data link communications described above [ms]

If DLRPI is more than 100 ms, calculate it as DLRPI = 100 ms.

Precautions for Correct Use

If the connection for tag data link communications that an NX-series EtherNet/IP Unit is the target is added, the X Bus Unit processing time may be increased and the task execution time may be extended.

Additional Information

The X Bus Unit processing time is *0* if X Bus Units are not connected.

● 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-56 and *Instruction Configuration for Standard ST Instructions* on page A-58 for information on the instruction configuration.

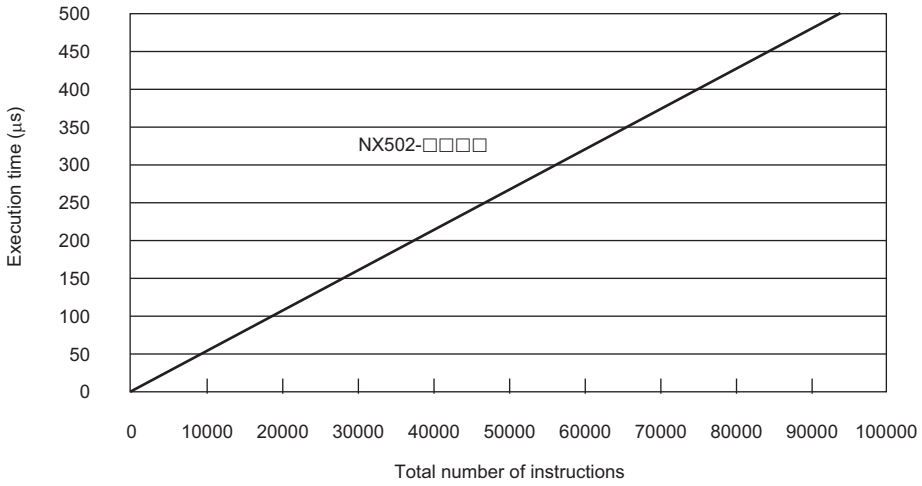
■ Simple Estimate

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

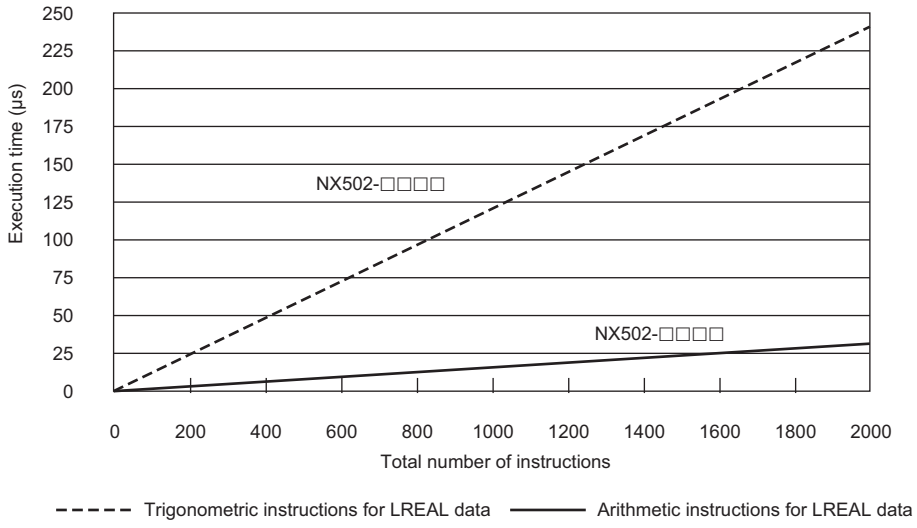


- Instruction Configuration for *Standard Ladder Diagram Instructions*

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
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*

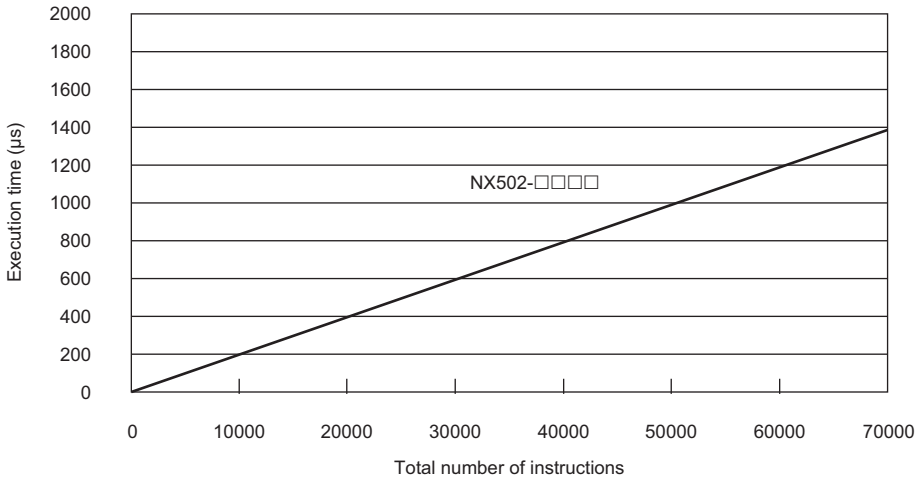
Instructions	Percent of instructions [%]
Addition instructions for LREAL data	20.0
Subtraction instructions for LREAL data	20.0
Multiplication instructions for LREAL data	30.0
Division instructions for LREAL data	30.0
Total	100.0

- Configuration of *Trigonometric Instructions for LREAL Data*

Instructions	Percent of instructions [%]
Sin of LREAL data	16.7

Instructions	Percent of instructions [%]
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

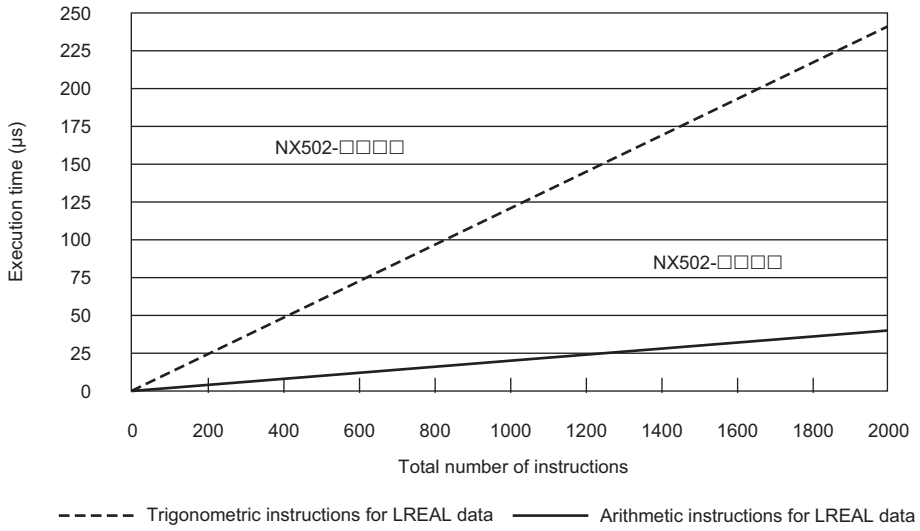
Execution Time for Standard ST Instructions



- 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

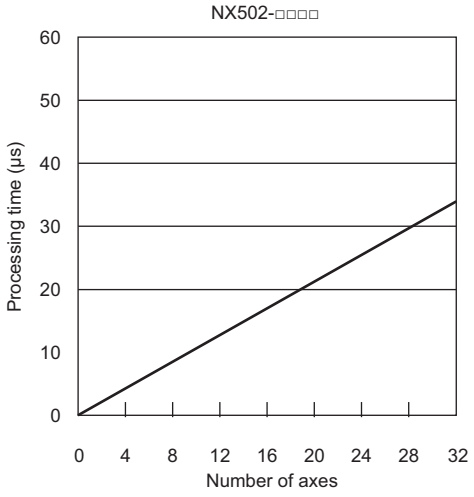
- Configuration of *Trigonometric Instructions for LREAL Data*

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)*1
	NX502-□□□□
Primary periodic task	15
Priority-16 periodic task	55
Priority-17 or 18 periodic task	25

*1. The common processing time is required to add the variable access time to the value in the table. Refer to *Calculating Variable Access Time* on page 5-122 for how to calculate the variable access time.

A-3-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period

Example of Calculating the Average Real Processing Times of Tasks

If you are using an NX502-□□□□ CPU Unit, first find the average real processing time of the task for the following conditions.

The task is the primary periodic task, and the primary period is 1 ms.

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
	NX Units (on CPU Rack)	• NX-PF0630 Additional I/O Power Supply Unit: 1 • NX-ID5342 Digital Input Units: 3 • NX-OD3153 Digital Output Units: 2 • NX-AD4608 Analog Input Unit: 1 • NX-DA3605 Analog Output Unit: 1 • NX-SL5500 Safety Control Unit: 1

Item		Conditions
	X Bus Units	NX-EIP201 EtherNet/IP Units: 3
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.

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

Total number of bytes are given below for pDout, pDin and pDinout.

pDout = 2 + 23 × 4 = 94 [byte]

pDin = 3 + 1 + 26 × 4 = 108 [byte]

pDinout = 3 + 2 + 26 × 4 = 109 [byte]

EtherCAT slave processing time

= 0.002 x pDout + 0.0014 x pDin + 0.082 x pDinout + 1.60 x Snum + 0.01 x Clen

+ 66.964

= 0.002 x 94 + 0.0014 x 108 + 0.082 x 109 + (1.60 x 6 + 0.01 x 10) + 66.964

= 0.188 + 0.1512 + 8.938 + (9.6 + 0.1) + 66.964

≈ 86 [μs]

NX Unit processing time:

The following table gives the Dnxout (output data size), Dnxin (input data size) values of the NX-PF0630 Additional I/O Power Supply Unit, NX-ID5342 Digital Input Unit, NX-OD3153 Digital Output Unit, NX-AD4608 Analog Input Unit, NX-DA3605 Analog Output Unit, and NX-SL5500 Safety Control Unit.

NX Unit	Dnxout: Output data size	Dnxin: Input data size
NX-PF0630	0	0
NX-ID5342	0	2
NX-OD3153	2	0
NX-AD4608	0	16
NX-DA3605	8	0
NX-SL5500	0	2

Total number of bytes are given below for Dnxout and Dnxin.

Dnxout = 2 × 2 + 8 = 12 [byte]

Dnxin = 2 × 3 + 16 + 2 = 24 [byte]

NX Unit processing time

= 0.152 x Dnxout + 0.053 x Dnxin + 0.549 x Dnxnum + 50.501

= 0.152 x 12 + 0.053 x 24 + 0.549 x 4 + 50.501

= 1.824 + 1.272 + 2.196 + 50.501

≈ 56 [μs]

X Bus Unit processing time:

The following table gives the DLSize (data size) and DLRPI (packet interval) values, and the number of connections for CIP Safety communications for each of tag data link connection for three NX-EIP201 EtherNet/IP Units.

X Bus Unit	DLSize: Data size of tag data links	DLRPI: Packet interval	Number of connections for CIP Safety communications
NX-EIP201	2	2,000	0
	4	4,000	
NX-EIP201	8	2,000	0
NX-EIP201	0	0	4

X Bus Unit processing time

= 0.290 x 4

+ (0.9 + 0.0035 × 2) × (1000/2000)) + (0.9 + 0.0035 × 2) × (1000/4000)) + (0.9 + 0.0035 × 2) × (1000/2000))

+ 2.25 x 3 + 36.18

= 1.16 + 0.4535 + 0.22675 + 0.4535 + 6.75 + 36.18

≈ 46 [μs]

I/O refresh time:

The largest value of EtherCAT slave processing time, NX Unit processing time, and X Bus Unit processing time is 86 [μs] for EtherCAT slave processing time.

● 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 NX502-□□□□ is 27 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 NX502-□□□□ is 3 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 NX502-□□□□ is 12 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 = 27 + 3 + 12
= 42 [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 NX502-□□□□ is read as 19 μs.

● Common Processing Time

Because the task is the primary periodic task, the common processing time for the NX502-□□□□ is 15 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
= 86 + 42 + 19 + 15
= 162 [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 x 1.1

Because the average real processing time of the task that is calculated above is 178 μs, the task period is set to 250 μs, which is larger than 1.1 times the average time.

The task execution times in the physical Controller depend on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors. The task execution time for a periodic task depends on whether it is interrupted for the execution of tasks with higher execution priorities.
Use the physical Controller and verify the task execution time with the Task Execution Time Monitor.

A-4 Calculating Guidelines for the Real Processing Times of Tasks for the NX102 System

This section describes how to calculate guidelines for the average real processing times of tasks on paper for the NX102 System.

You must use the physical Controller to check the real processing times of tasks and task execution times. Refer to *5-12 Task Design Methods and I/O Response Times* on page 5-133 for details.

Precautions for Safe Use

The task execution times in the physical Controller depend on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors.
Before starting actual operation, you must test performance under all foreseeable conditions on the actual system and make sure that the task periods are not exceeded and that suitable communications performance is achieved.

Additional Information

Periodic tasks will be interrupted for the execution of tasks with higher execution priorities. The real processing time of a task does not include the time for which the task is interrupted. It is the task execution time that gives the actual time from when the task is started until it is finished, including the interrupted time. For a detailed description of the differences between the real processing times of tasks and the task execution times, refer to *Meaning of the Task Execution Time and the Real Processing Time of the Task* on page 5-132.

A-4-1 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.

○: Performed, ---: Not performed.

Processing		Processing contents	Primary periodic task	Priority-17 and priority-18 periodic tasks
I/O refresh processing		I/O is refreshed for NX Units on the CPU Unit and EtherCAT slaves.	○	----
User program execution		Programs assigned to tasks are executed in the order that they are assigned.	○	○
Motion control processing		• Motion control commands from the user program are executed. • Motion output processing	○	---
Common processing time	System common processing 1	• Variable refresh processing (if there are accessing tasks) is performed. • Motion input processing • Data trace processing	○	○
	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	○	○
	System overhead time	Other system common processing	○	○

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 + NX Unit processing time

The following describes how to determine the EtherCAT slave processing time and NX Unit processing time used in the above formula.

● EtherCAT Slave Processing Time

Use the following formula for the EtherCAT slave processing time.

EtherCAT slave processing time [μs] = 0.017 × pDout + 0.010 × pDin + 0.082 × pDinout + (1.24(*1) × Snum + 0.01 × Clen) + 151.11

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]

*1 For project unit version 1.40 or later, this value is 1.6.

Additional Information

The EtherCAT slave processing time is 0 if EtherCAT slaves are not connected.

● NX Unit Processing Time

Use the following formula for the NX Unit processing time.
The calculation must include all NX Units on the CPU Unit.

NX Unit processing time [μs] = 0.149 × Dnxout + 0.064 × Dnxin × 0.549 × Dnxnum + 97.48

Dnxout	:	: Total output data size [byte]*1
Dnxin	:	: Total input data size [byte]*1
Dnxnum	:	Number of connected NX Units

*1. If the data size of an NX Unit is less than 2 bytes, calculate as 2 bytes.

Additional Information

The NX Unit processing time is 0 if NX Units are not connected to the CPU Unit.

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-56 and *Instruction Configuration for Standard ST Instructions* on page A-58 for information on the instruction configuration.

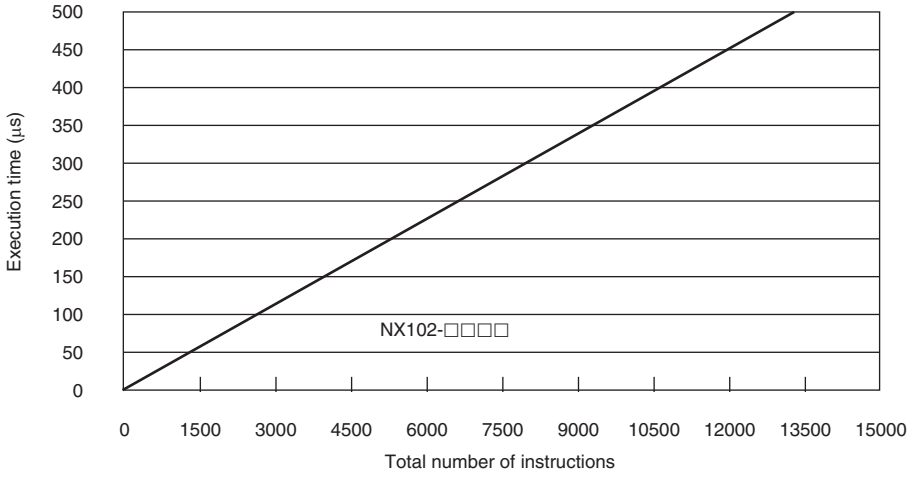
● Simple Estimate

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



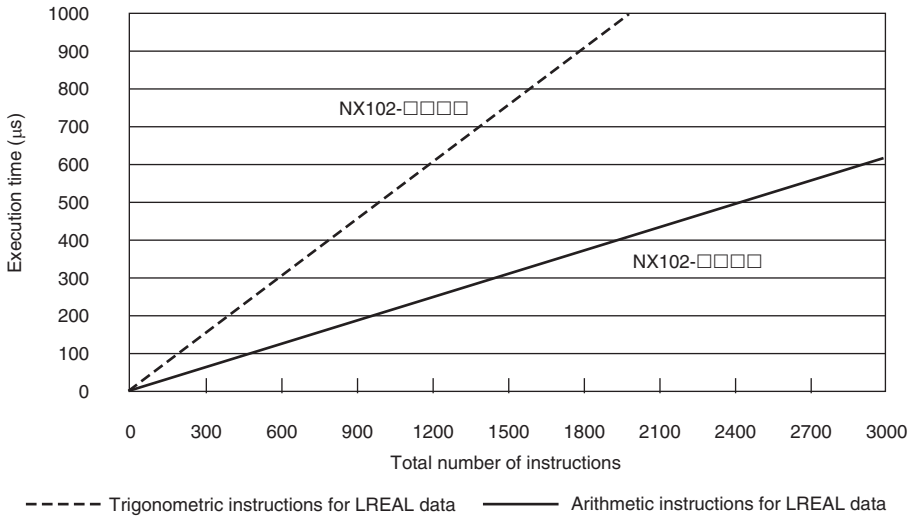
- Instruction Configuration for Standard Ladder Diagram Instructions

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

Types of instructions	Instructions	Percent of instructions [%]	Percent of execution time in instruction group [%]
Timer and counter instructions	Timer, TON/TOF, and CTU/CTD	1.6	7.3
Math instructions	+, -, *, /, ADD, SUB, MUL, and DIV	2.4	6.5
BCD conversion instructions and data conversion instructions	INT_TO_DINT and WORD_BCD_TO_UINT	0.2	1.2
Bit string processing instructions	AND and OR	6.2	13.0
Data movement instructions	MOVE	4.6	23.5
Total		100.0	100.0

Execution Times for Ladder Diagram Arithmetic and Trigonometric Instructions for LREAL Data



- Configuration of Arithmetic Instructions for LREAL Data

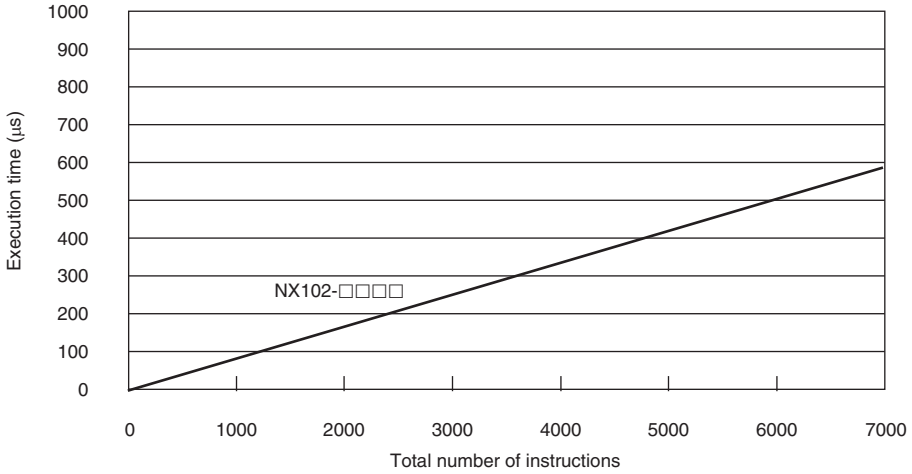
Instructions	Percent of instructions [%]
Addition instructions for LREAL data	20.0
Subtraction instructions for LREAL data	20.0
Multiplication instructions for LREAL data	30.0
Division instructions for LREAL data	30.0
Total	100.0

- Configuration of Trigonometric Instructions for LREAL Data

Instructions	Percent of instructions [%]
Sin of LREAL data	16.7
Cos of LREAL data	16.7
Tan of LREAL data	16.7

Instructions	Percent of instructions [%]
Sin^{-1} of LREAL data	16.7
Cos^{-1} of LREAL data	16.7
Tan^{-1} of LREAL data	16.7
Total	100.0

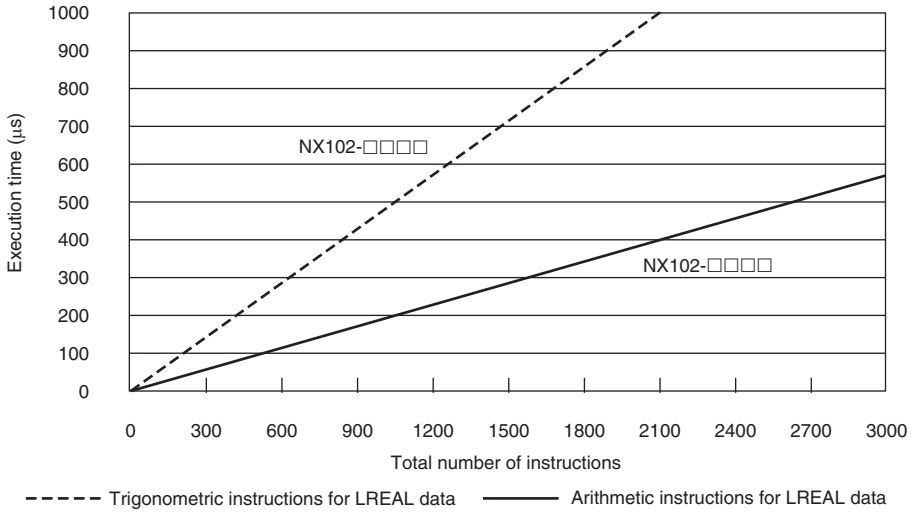
Execution Time for Standard ST Instructions



- 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

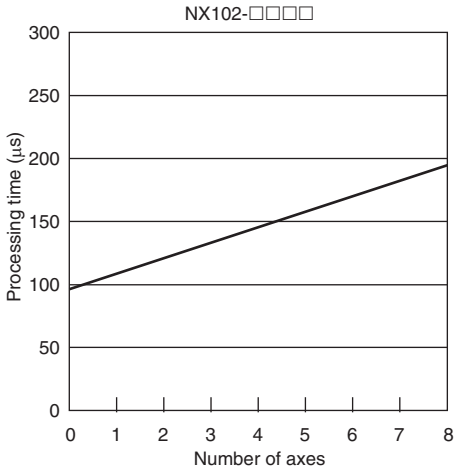
- Configuration of Trigonometric Instructions for LREAL Data

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)
	NX102-□□□□
Primary periodic task	133
Periodic task	175

A-4-2 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 NX102-□□□□ CPU Unit with a unit version of 1.30, 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
	NX Units (on CPU Rack)	• NX-PF0630 Additional I/O Power Supply Unit: 1 • NX-ID5342 Digital Input Units: 3 • NX-OD3153 Digital Output Units: 2 • NX-AD4608 Analog Input Unit: 1 • NX-DA3605 Analog Output Unit: 1
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.

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

Total number of bytes are given below for pDout, pDin and pDinout.

pDout = 2 + 23 × 4 = 94 [byte]
pDin = 3 + 1 + 26 × 4 = 108 [byte]
pDinout = 3 + 2 + 26 × 4 = 109 [byte]

NX Unit processing time:

The following table gives the Dnxout (output data size), Dnxin (input data size) values of the NX-PF0630 Additional I/O Power Supply Unit, NX-ID5342 Digital Input Unit, NX-OD3153 Digital Output Unit, NX-AD4608 Analog Input Unit, and NX-DA3605 Analog Output Unit.

NX Units	Dnxout: Output data size	Dnxin: Input data size
NX-PF0630	0	0
NX-ID5342	0	2
NX-OD3153	2	0
NX-AD4608	0	16
NX-DA3605	8	0

Total number of bytes are given below for Dnxout and Dnxin.

Dnxout = 2 × 2 + 8 = 12 [byte]

Dnxin = 2 × 3 + 16 = 22 [byte]

From these values, the I/O refresh time is calculated by the following formula.

I/O refresh processing time

= EtherCAT slave processing time + NX Unit processing time

= 0.017 × pDout + 0.010 × pDin + 0.082 × pDinout + (1.24 × Snum + 0.01 × Clen)

+ 151.11 + 0.149 × Dnxout + 0.064 × Dnxin + 0.549 × Dnxnum + 97.48

= 0.017 × 94 + 0.010 × 108 + 0.082 × 109 + (1.24 × 6 + 0.01 × 10) + 151.11

+ 0.149 × 12 + 0.064 × 22 + 0.549 × 8 + 97.48

= 1.598 + 1.080 + 8.938 + (7.24 + 0.1) + 151.11

+ 1.788 + 1.408 + 4.392 + 97.48

≈ 275 [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 NX102-□□□□ is 189 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 NX102-□□□□ is 41 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 NX102-□□□□ is 51 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 = 189 + 41 + 51

= 281 [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 NX102-□□□□ with a unit version 1.30 is read as 148 μs.

● Common Processing Time

Because the task is the primary periodic task, the common processing time for the NX102-□□□□ is 133 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
= 275 + 281 + 148 + 133
= 837 [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 x 1.1

Because the average real processing time of the task that is calculated above is 837 μs, the task period is set to 1000 μs, which is larger than 1.1 times the average time.

The task execution times in the physical Controller depend on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors. The task execution time for a periodic task depends on whether it is interrupted for the execution of tasks with higher execution priorities.
Use the physical Controller and verify the task execution time with the Task Execution Time Monitor.

A-5 Calculating Guidelines for the Real Processing Times of Tasks for the NX1P2 System

This section describes how to calculate guidelines for the average real processing times of tasks on paper for the NX1P2 System.

You must use the physical Controller to check the real processing times of tasks and task execution times. Refer to *5-12 Task Design Methods and I/O Response Times* on page 5-133 for details.

Precautions for Safe Use

The task execution times in the physical Controller depend on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors.
Before starting actual operation, you must test performance under all foreseeable conditions on the actual system and make sure that the task periods are not exceeded and that suitable communications performance is achieved.

Additional Information

Periodic tasks will be interrupted for the execution of tasks with higher execution priorities. The real processing time of a task does not include the time for which the task is interrupted. It is the task execution time that gives the actual time from when the task is started until it is finished, including the interrupted time. For a detailed description of the differences between the real processing times of tasks and the task execution times, refer to *Meaning of the Task Execution Time and the Real Processing Time of the Task* on page 5-132.

A-5-1 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.

○: Performed, ---: Not performed.

Processing		Processing contents	Primary periodic task	Priority-17 and priority-18 periodic tasks
I/O refresh processing		I/O is refreshed for NX Units on the CPU Unit, built-in I/Os, and EtherCAT slaves.	○	---
User program execution		Programs assigned to tasks are executed in the order that they are assigned.	○	○
Motion control processing		• Motion control commands from the user program are executed. • Motion output processing	○	---
Common processing time	System common processing 1	• Variable refresh processing (if there are accessing tasks) is performed. • Motion input processing • Data trace processing	○	○
	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	○	○
	System overhead time	Other system common processing	○	○

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 + NX Unit processing time

The following describes how to determine the EtherCAT slave processing time and NX Unit processing time used in the above formula.

● EtherCAT Slave Processing Time

Use the following formula for the EtherCAT slave processing time.

EtherCAT slave processing time [μs] = 0.017 × pDout + 0.010 × pDin + 0.082 × pDinout + (1.24(*1) × Snum + 0.01 × Clen) + 151.11

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]

*1 For project unit version 1.40 or later, this value is 1.6.

Additional Information

The EtherCAT slave processing time is 0 if EtherCAT slaves are not connected.

● NX Unit Processing Time

Use the following formula for the NX Unit processing time.
The calculation must include all NX Units on the CPU Unit.

NX Unit processing time [μs]= 0.248 × Dnxout + 0.241 × Dnxin + 0.549 × Dnxnum + 125.36

Dnxout	:	Total output data size [byte]*1
Dnxin	:	Total input data size [byte]*1
Dnxnum	:	Number of connected NX Units

*1. If the data size of an NX Unit is less than 2 bytes, calculate as 2 bytes.

Additional Information

The NX Unit processing time is 0 if NX Units are not connected to the CPU Unit.

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* page A-67 and *Instruction Configuration for Standard ST Instructions* page A-69 for information on the instruction configuration.

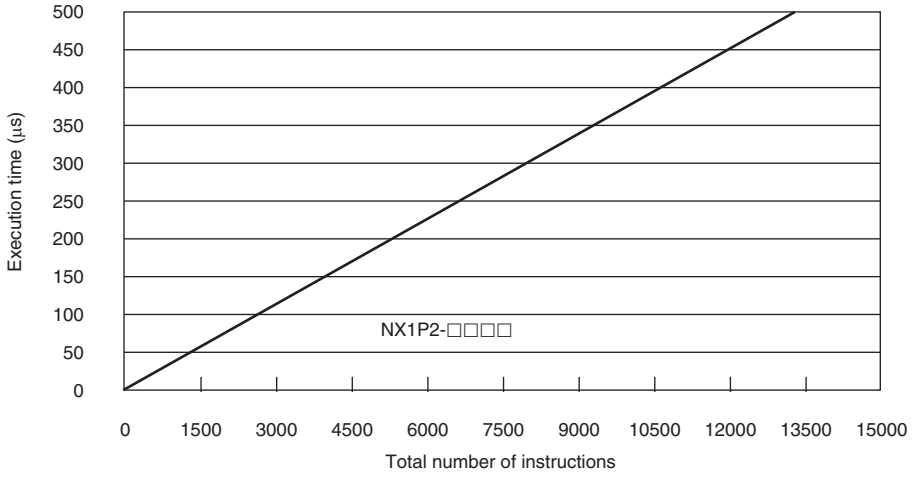
● Simple Estimate

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



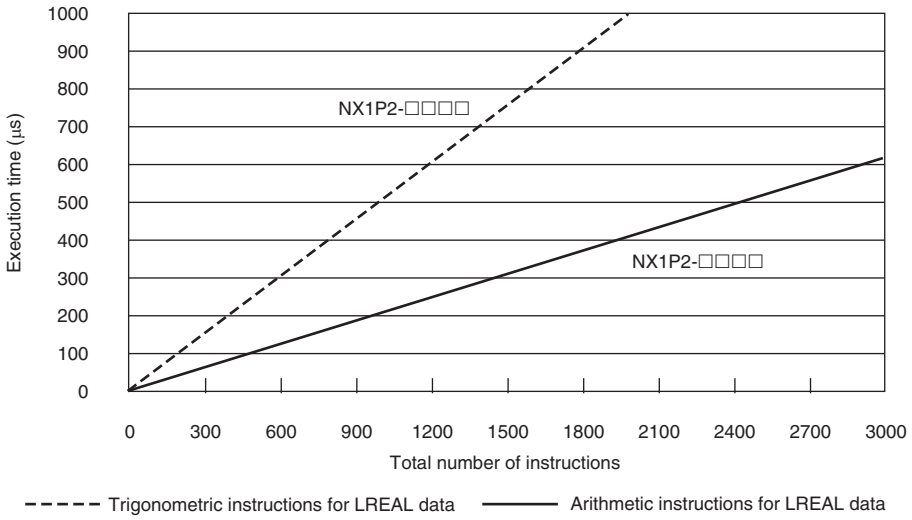
- Instruction Configuration for Standard Ladder Diagram Instructions

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

Types of instructions	Instructions	Percent of instructions [%]	Percent of execution time in instruction group [%]
Timer and counter instructions	Timer, TON/TOF, and CTU/CTD	1.6	7.3
Math instructions	+, -, *, /, ADD, SUB, MUL, and DIV	2.4	6.5
BCD conversion instructions and data conversion instructions	INT_TO_DINT and WORD_BCD_TO_UINT	0.2	1.2
Bit string processing instructions	AND and OR	6.2	13.0
Data movement instructions	MOVE	4.6	23.5
Total		100.0	100.0

Execution Times for Ladder Diagram Arithmetic and Trigonometric Instructions for LREAL Data



- Configuration of Arithmetic Instructions for LREAL Data

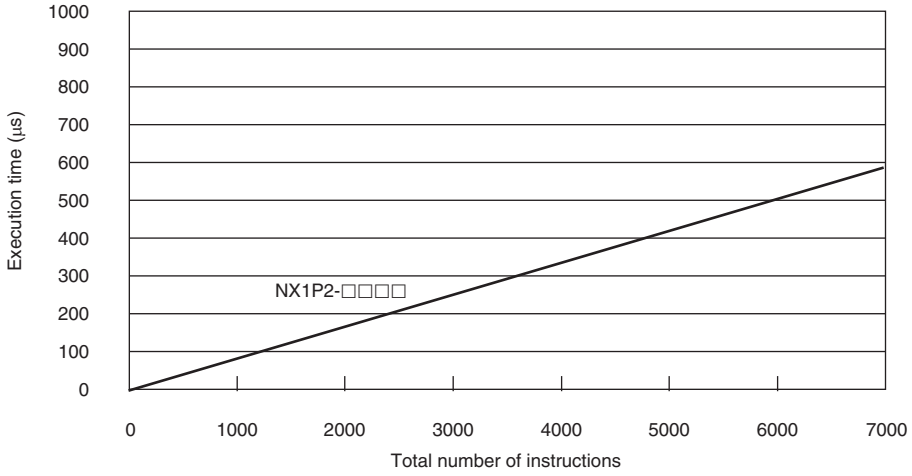
Instructions	Percent of instructions [%]
Addition instructions for LREAL data	20.0
Subtraction instructions for LREAL data	20.0
Multiplication instructions for LREAL data	30.0
Division instructions for LREAL data	30.0
Total	100.0

- Configuration of Trigonometric Instructions for LREAL Data

Instructions	Percent of instructions [%]
Sin of LREAL data	16.7
Cos of LREAL data	16.7
Tan of LREAL data	16.7

Instructions	Percent of instructions [%]
Sin^{-1} of LREAL data	16.7
Cos^{-1} of LREAL data	16.7
Tan^{-1} of LREAL data	16.7
Total	100.0

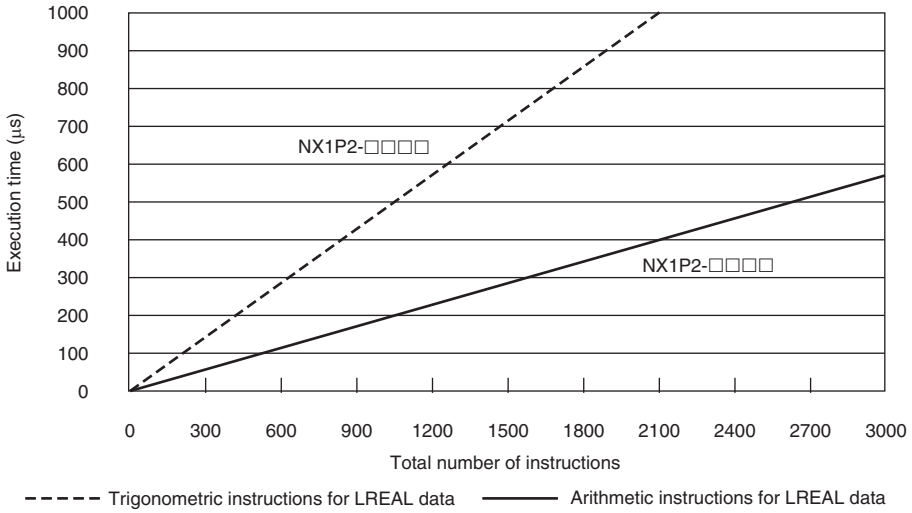
Execution Time for Standard ST Instructions



- 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

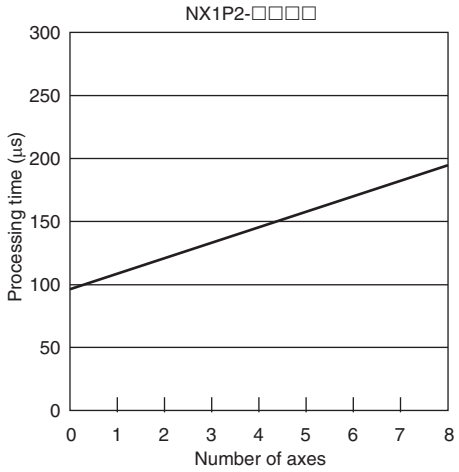
Instructions	Percent of instructions [%]
Addition instructions for LREAL data	20.0
Subtraction instructions for LREAL data	20.0
Multiplication instructions for LREAL data	30.0
Division instructions for LREAL data	30.0
Total	100.0

- 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)
	NX1P2-□□□□
Primary periodic task	133
Periodic task	175

A-5-2 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 NX1P2-□□□□ CPU Unit with a unit version of 1.13, 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
	NX Units (on CPU Rack)	• NX-PF0630 Additional I/O Power Supply Unit: 1 • NX-ID5342 Digital Input Units: 3 • NX-OD3153 Digital Output Units: 2 • NX-AD4608 Analog Input Unit: 1 • NX-DA3605 Analog Output Unit: 1
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.

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

Total number of bytes are given below for pDout, pDin and pDinout.

pDout = 2 + 23 × 4 = 94 [byte]
pDin = 3 + 1 + 26 × 4 = 108 [byte]
pDinout = 3 + 2 + 26 × 4 = 109 [byte]

NX Unit processing time:

The following table gives the Dnxout (output data size), Dnxin (input data size) values of the NX-PF0630 Additional I/O Power Supply Unit, NX-ID5342 Digital Input Unit, NX-OD3153 Digital Output Unit, NX-AD4608 Analog Input Unit, and NX-DA3605 Analog Output Unit.

NX Units	Dnxout: Output data size	Dnxin: Input data size
NX-PF0630	0	0
NX-ID5342	0	2
NX-OD3153	2	0
NX-AD4608	0	16
NX-DA3605	8	0

Total number of bytes are given below for Dnxout and Dnxin.

Dnxout = 2 × 2 + 8 = 12 [byte]

Dnxin = 2 × 3 + 16 = 22 [byte]

From these values, the I/O refresh time is calculated by the following formula.

I/O refresh processing time

= EtherCAT slave processing time + NX Unit processing time

= 0.017 × pDout + 0.010 × pDin + 0.082 × pDinout + (1.24 × Snum + 0.01 × Clen)

+ 151.11 + 0.248 × Dnxout + 0.241 × Dnxin + 0.549 × Dnxnum + 125.36

= 0.017 × 94 + 0.010 × 108 + 0.082 × 109 + (1.24 × 6 + 0.01 × 10) + 151.11

+ 0.248 × 12 + 0.241 × 22 + 0.549 × 8 + 125.36

= 1.598 + 1.080 + 8.938 + (7.24 + 0.1) + 151.11

+ 2.976 + 5.302 + 4.392 + 125.36

≈ 308 [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 NX1P2-□□□□ is 189 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 NX1P2-□□□□ is 41 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 NX1P2-□□□□ is 51 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 = 189 + 41 + 51

= 281 [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 NX1P2-□□□□ with a unit version 1.13 is read as 148 μs.

● Common Processing Time

Because the task is the primary periodic task, the common processing time for the NX1P2-□□□□ is 133 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
= 308 + 281 + 148 + 133
= 870 [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 x 1.1

Because the average real processing time of the task that is calculated above is 870 μs, the task period is set to 2000 μs, which is larger than 1.1 times the average time.

The task execution times in the physical Controller depend on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors. The task execution time for a periodic task depends on whether it is interrupted for the execution of tasks with higher execution priorities.
Use the physical Controller and verify the task execution time with the Task Execution Time Monitor.

A-6 Calculating Guidelines for the Real Processing Times of Tasks for the NJ-series System

This section describes how to calculate guidelines for the average real processing times of tasks on paper for the NJ-series System.

You must use the physical Controller to check the real processing times of tasks and task execution times. Refer to *5-12 Task Design Methods and I/O Response Times* on page 5-133 for details.

Precautions for Safe Use

- The task execution times in the physical Controller depend 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.
- The performance may be different if the hardware revisions are different. Before you transfer the user program, data, and parameter settings to the CPU Units with the different hardware revisions, check them for proper execution and then use them for actual operation.

Precautions for Correct Use

- When the hardware revisions are different, the actual execution timing of the event tasks, tag data link service and system services may be different even if the user program, data, and parameter settings are same.
- Processing performance of I/O refresh, user program, and motion control has been improved in the hardware revision *E* or later, so the actual processing time may be shorter.

Additional Information

Periodic tasks will be interrupted for the execution of tasks with higher execution priorities. The real processing time of a task does not include the time for which the task is interrupted. It is the task execution time that gives the actual time from when the task is started until it is finished, including the interrupted time. For a detailed description of the differences between the real processing times of tasks and the task execution times, refer to *Meaning of the Task Execution Time and the Real Processing Time of the Task* on page 5-132.

A-6-1 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.

○: Performed, ---: Not performed.

Processing		Processing contents	Primary periodic task	Priority-16 periodic task	Priority-17 and priority-18 periodic tasks
I/O refresh processing		I/O is refreshed for CJ-series Units (Basic I/O Units, Special I/O Units, and CPU Bus Units) and EtherCAT slaves.	○	○	---
User program execution		Programs assigned to tasks are executed in the order that they are assigned.	○	○	○
Motion control processing		• Motion control commands from the user program are executed. • Motion output processing	○	---	---
Common processing time	System common processing 1	• Variable refresh processing (if there are accessing tasks) is performed. • Motion input processing • Data trace processing	○	○	○
	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	○	○	○
	System overhead time	Other system common processing	○	○	○

Guidelines are provided below for calculating the various processing times.

I/O Refresh Processing Time

Use the following formula for the I/O refresh processing time.

I/O refresh processing time = I/O refresh overhead time
+ (Larger of the EtherCAT slave processing time and the CJ-series Unit processing time)

The following describes how to determine the I/O refresh overhead time, EtherCAT slave processing time, and CJ-series Unit processing time used in the above formula.

Precautions for Correct Use

Performance of I/O refresh execution has been improved in the hardware revision *E* or later, so the actual processing time may be shorter.

● I/O Refresh Overhead Time

The I/O refresh overhead time is given by the following table, depending on whether there are Ether-CAT slaves and CJ-series Units, and also depending on the models of the CPU Units.

EtherCAT slave	CJ-series Unit	I/O refresh overhead time [μs]		
		NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Present	Present	65*1	95	135
Present	None	35*2	50	65
None	Present	30	45	70

*1. The value is 90 for a CPU Unit with unit version 1.01 or earlier.
*2. The value is 60 for a CPU Unit with unit version 1.01 or earlier.

● EtherCAT Slave Processing Time

Use the following formula for the EtherCAT slave processing time.

EtherCAT slave processing time [μs] = Tout × Dout + Tin × Din + Tref × Dinout
+ (1.24(*1) × Snum + 0.01 × Clen - Tec)

Tout	: Output processing time per byte [μs]
Dout	: Total output processing data size [byte] of EtherCAT slaves assigned to the primary periodic task or the priority-16 periodic task
Tin	: Input processing time per byte [μs]
Din	: Total input processing data size [byte] of EtherCAT slaves assigned to the primary periodic task or the priority-16 periodic task
Snum	: Number of connected EtherCAT slaves
Tref	: Refresh processing time per byte [μs]
Dinout	: Total of the larger of the input and output processing data size of each EtherCAT slave assigned to the primary periodic task or the priority-16 periodic task [byte]
Clen	: Total cable length [m]
Tec	: EtherCAT communications adjustment time [μs]

*1 For project unit version 1.40 or later, this value is 1.6.

The values of the output processing time, input processing time, refresh processing time, and EtherCAT communications adjustment time in the above formula are fixed. They are determined by the model of the CPU Unit as given in the following table.

CPU Unit	Tout: Output processing time per byte [µs]	Tin: Input processing time per byte [µs]	Tref: Refresh processing time per byte [µs]	Tec: EtherCAT communications adjustment time [µs]
NJ501-□□□□	0.004	0.011	0.082	55*1
NJ301-□□□□	0.005	0.013	0.082	90
NJ101-□□□□	0.010	0.004	0.082	145

*1. The value is 70 for a CPU Unit with unit version 1.01 or earlier.

If the result that is calculated from the part (1.24 × Snum + 0.01 × Clen - Tec) of the above formula is a negative number, the result is regarded as 0.

● CJ-series Unit Processing Time

Use the following formula for the CJ-series Unit processing time.

Σ (I/O refresh time for each CJ-series Unit × Number of Units) − Tcj [µs]

In the above formula, Σ represents the total processing time for all CJ-series Units.
If the result that is calculated from the above formula is a negative number, the CJ-series Unit processing time is regarded as 0 µs.
The method for calculating the I/O refresh time for each CJ-series Unit is provided later.
The value of Tcj depends on the model and the unit version of the CPU Unit.

Model	Unit version	Tcj [µs]
NJ501-□□□□	Ver. 1.01 or earlier	230
	Ver. 1.02 or later	110
NJ301-□□□□	Ver. 1.01 or earlier	230
	Ver. 1.02 or later	165
NJ101-□□□□	Ver. 1.10 or later*1	225

*1. There is no NJ101-□□□□ CPU Unit with unit version 1.09 or earlier.

If any of the following CJ-series Units is used, add Tcj [µs] to the result that is calculated from the above formula, regardless of the number of Units.

- CJ1W-PH41U Isolated-type Units with Universal Inputs
- CJ1W-AD04U Isolated-type Units with Universal Inputs
- CJ1W-PDC15 Isolated-type DC Input Unit
- CJ1W-V680C11 ID Sensor Unit
- CJ1W-V680C12 ID Sensor Unit
- CJ1W-CRM21 CompoNet Master Unit

I/O Refresh Times for CJ-series Units

This section gives the I/O refresh times for CJ-series Units.

Basic I/O Units

The following table gives the I/O refresh times for Basic I/O Units that are used on the CPU Rack. The I/O refresh times for these Units on Expansion Racks are approx. 1.5 times the values that are given in the table.

Unit name	Model numbers	I/O refresh time per Unit [μs]
8/16-point DC Input Units	CJ1W-ID201/211/212	1
32-point DC Input Units	CJ1W-ID231/232/233	2
64-point DC Input Units	CJ1W-ID261/262	4
8/16-point AC Input Units	CJ1W-IA201/111	1
16-point Interrupt Input Unit	CJ1W-INT01	1
16-point Quick-response Input Unit	CJ1W-IDP01	1
Relay Contact Output Units	CJ1W-OC201/211	1
Triac Output Unit	CJ1W-OA201	1
8/16-point Transistor Output Units	CJ1W-OD201/202/203/204//211/212/213	1
32-point Transistor Output Units	CJ1W-OD231/232/233/234	2
64-point Transistor Output Units	CJ1W-OD261/262/263	4
24-VDC Input/Transistor Output Units (16 inputs/16 outputs)	CJ1W-MD231/232/233	1
24-VDC Input/Transistor Output Units (32 inputs/32 outputs)	CJ1W-MD261/263	2
TTL I/O Unit (16 inputs/16 outputs)	CJ1W-MD563	4
B7A Interface Units	CJ1W-B7A04	4
	CJ1W-B7A14	4
	CJ1W-B7A22	4

Special I/O Units

The following table gives the I/O refresh times for Special I/O Units that are used on the CPU Rack. The I/O refresh times for these Units on Expansion Racks are approx. 1.5 times the values that are given in the table.

Unit name	Model numbers	I/O refresh time per Unit [μs]
Isolated-type Units with Universal Inputs	CJ1W-AD04U	66
Analog Input Units	CJ1W-AD041-V1/081-V1/042	24
Analog Output Units	CJ1W-DA021/041/042V/08V/08C	24
Analog I/O Units	CJ1W-MAD42	24
Isolated-type Units with Universal Inputs	CJ1W-PH41U	80 (180)*1
Isolated-type DC Input Unit	CJ1W-PDC15	60 (100)*1
Temperature Control Units	CJ1W-TC□□□	114
ID Sensor Units	CJ1W-V680C11	76
	CJ1W-V680C12	86
High-speed Counter Unit	CJ1W-CT021	54

Unit name	Model numbers		I/O refresh time per Unit [μs]
CompoNet Master Unit	CJ1W-CRM21	Communications mode 0	34
		Communications mode 1	52
		Communications mode 2	88
		Communications mode 3	84
		Communications mode 8	14 + (1.0 × Number of allocated words)*2

*1. The values in parentheses are the I/O refresh times when an expansion allocation area is used.
*2. The number of allocated words is the total number of I/O area words that are allocated to all of the slaves.

● CPU Bus Units

The following table gives the I/O refresh times for CPU Bus Units. The times are the same regardless of whether the Units are used on the CPU Rack or an Expansion Rack.

Unit name	Model numbers	I/O refresh time per Unit [μs]
Serial Communications Units	CJ1W-SCU42 CJ1W-SCU32 CJ1W-SCU22	Coefficient*1 x 25*2
DeviceNet Unit	CJ1W-DRM21	Coefficient*1 x (Number of allocated words*3 + 25)
EtherNet/IP Unit	CJ1W-EIP21	7.0 + 0.8 x Number of tags*4

*1. The coefficient is determined by the model of the CPU Unit as follows.
NJ501-□□□□: 0.1
NJ301-□□□□: 0.2
NJ101-□□□□: 0.3
*2. The following maximum time is added if a protocol macro is executed:
Coefficient × Number of refresh words [μs]
*3. The number of allocated words is the total number of I/O area words that are allocated to all of the slaves.
*4. For a CPU Unit with unit version 1.03 or later, the I/O refresh times for EtherNet/IP Units are not added.

User Program Execution Time

The user program execution time depends on the specific instructions multiplied by the numbers of instructions used.

As a guideline, instructions are divided into three groups and the number of instructions in each group is used for measurements and estimates.

- Standard instructions
- Arithmetic instructions for LREAL data
- Trigonometric instructions for LREAL data

Different instructions are used in a ladder diagram and in ST. Refer to *Instruction Configuration for Standard Ladder Diagram Instructions* on page A-81 and *Instruction Configuration for Standard ST Instructions* on page A-84 for information on the instruction configuration.

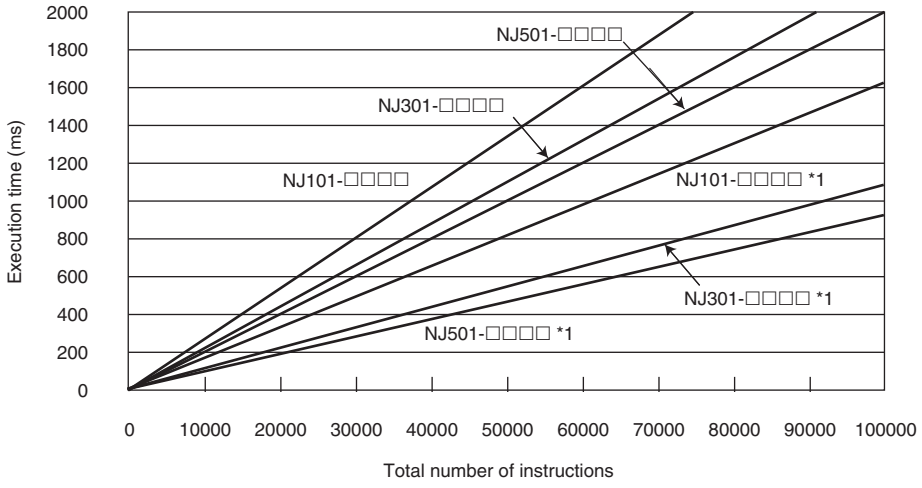
● Simple Estimate

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



*1. It is the case when the hardware revision for the Unit is *A* or later. The other cases are for the Units that the hardware revisions are in blank.

- Instruction Configuration for Standard Ladder Diagram Instructions

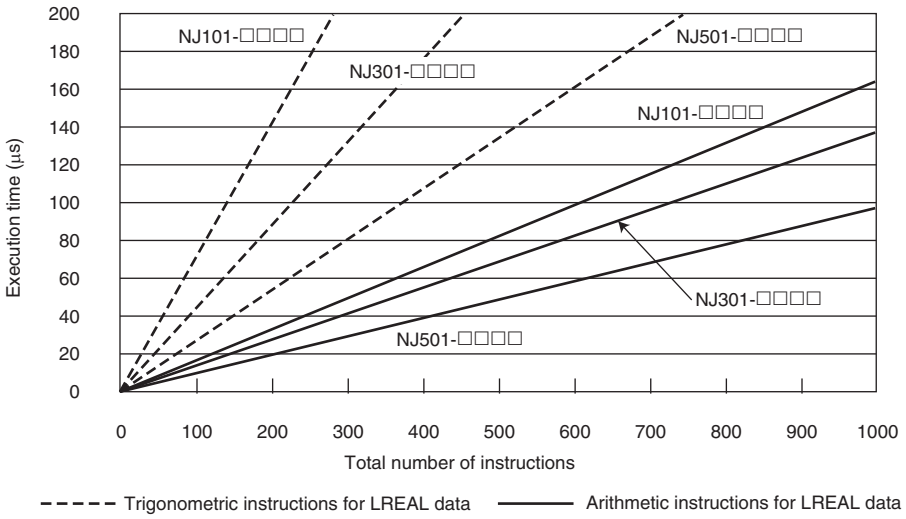
The instruction execution ratio for this configuration is 20%.

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

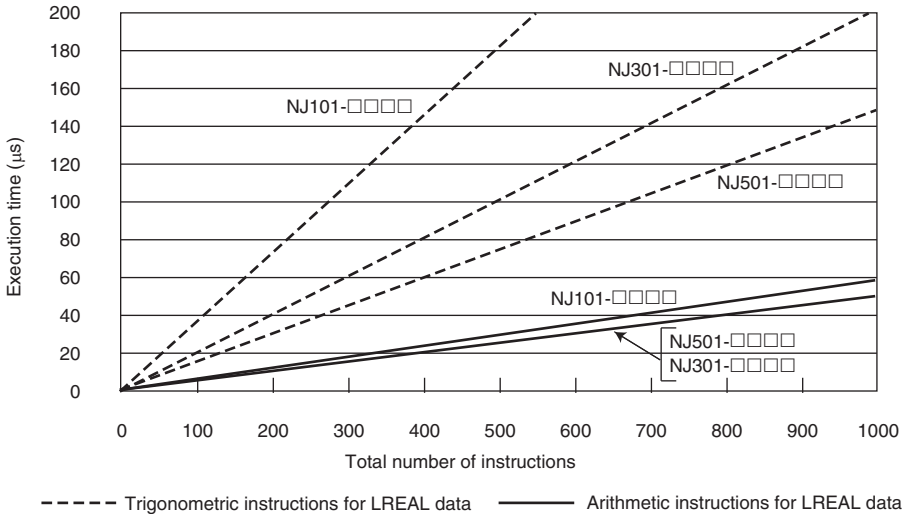
Types of instructions	Instructions	Percent of instructions [%]	Percent of execution time in instruction group [%]
Comparison instructions	EQ and LT	4.1	8.3
Timer and counter instructions	Timer, TON/TOF, and CTU/CTD	1.6	7.3
Math instructions	+, -, *, /, ADD, SUB, MUL, and DIV	2.4	6.5
BCD conversion instructions and data conversion instructions	INT_TO_DINT and WORD_BCD_TO_UINT	0.2	1.2
Bit string processing instructions	AND and OR	6.2	13.0
Data movement instructions	MOVE	4.6	23.5
Total		100.0	100.0

Execution Times for Ladder Diagram Arithmetic and Trigonometric Instructions for LREAL Data

- When hardware revisions for the Units are in blank



- When hardware revisions for the Units are *A* or later



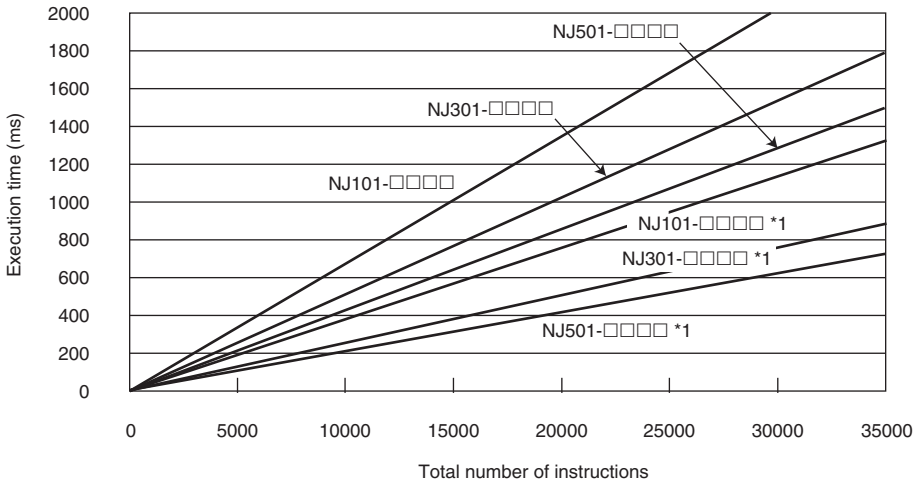
- Configuration of *Arithmetic Instructions for LREAL Data*

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

- Composition 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



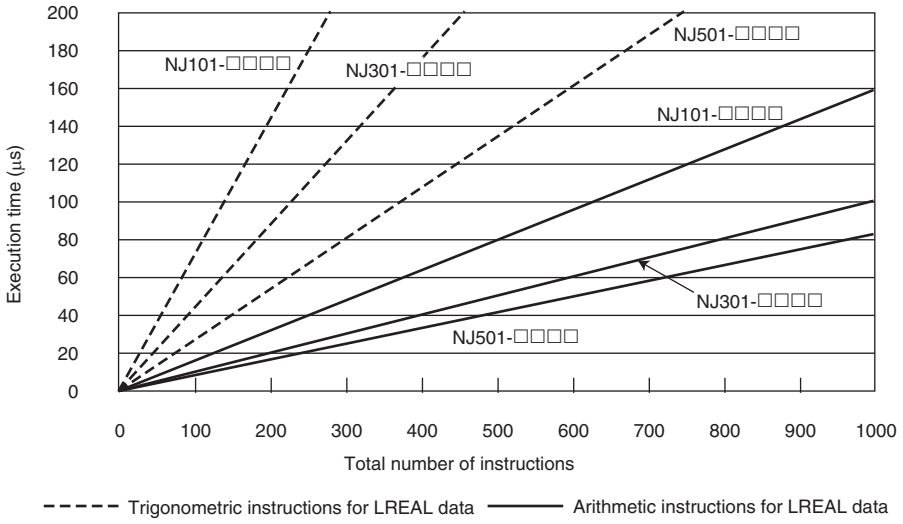
*1. It is the case when the hardware revision for the Unit is *A* or later. The other cases are for the Units that the hardware revisions are in blank.

- Instruction Configuration for Standard ST Instructions

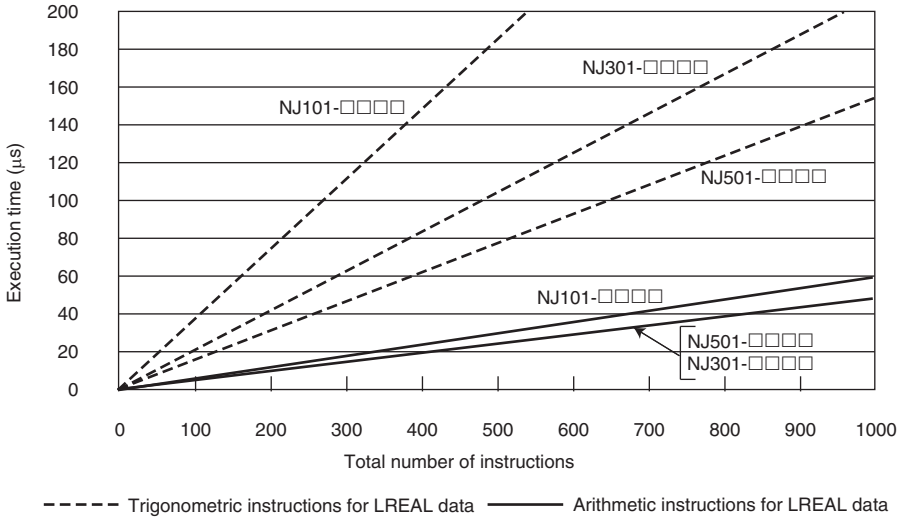
Types of instructions	Instructions	Percent of instructions [%]	Percent of execution time in instruction group [%]
ST constructs	IF ELSIF END_IF	75.4	41.6
Comparison instructions	EQ and LT	5.2	8.7
Timer and counter instructions	Timer, TON/TOF, and CTU/CTD	2.1	18.8
Math instructions	+, -, *, and /	3.1	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

- When hardware revisions for the Units are in blank



- When hardware revisions for the Units are *A* or later



- 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

Instructions	Percent of instructions [%]
Tan^{-1} of LREAL data	16.7
Total	100.0

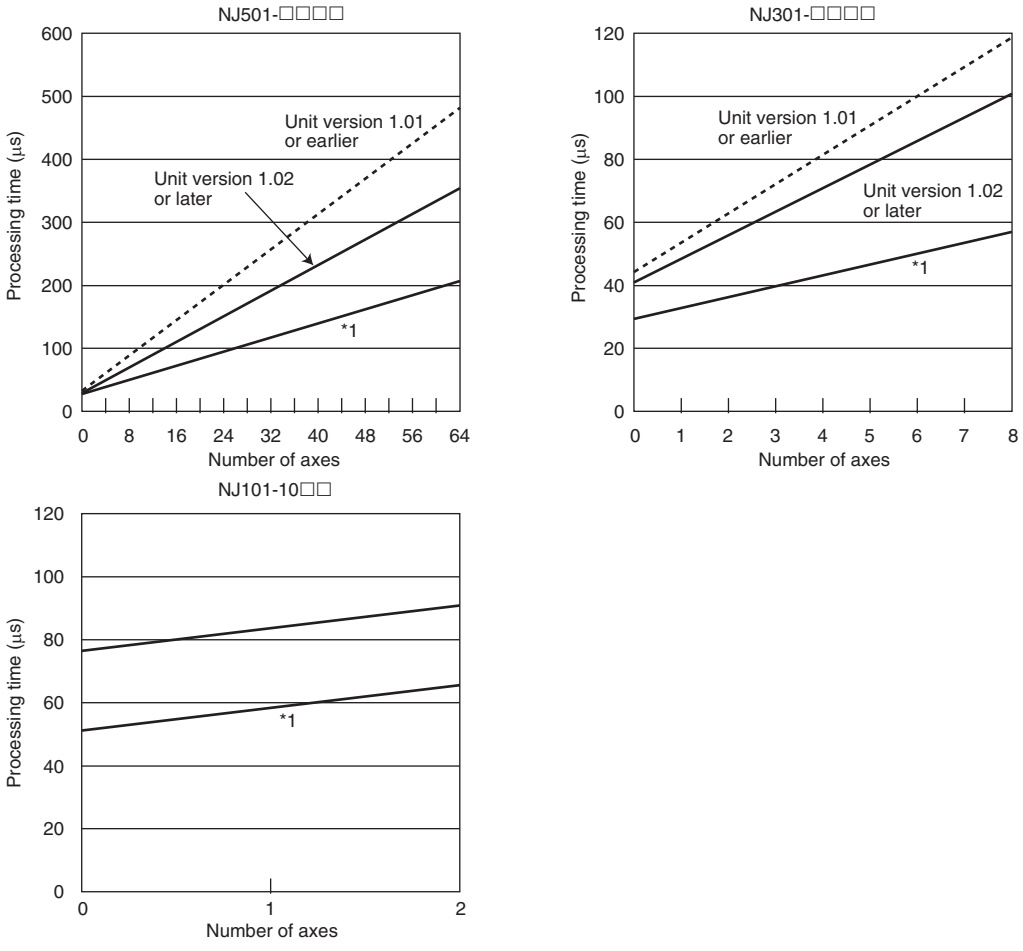
Precautions for Correct Use

Performance of user program execution has been improved in the hardware revision *E* or later, so the actual execution time may be shorter.

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.



*1. It is the case when the hardware revision for the Unit is *A* or later. The other cases are for the Units that the hardware revisions are in blank.

Note You cannot use the motion control functions with an NJ101-90□□ CPU Unit.

Precautions for Correct Use

Processing performance of motion control has been improved in the hardware revision *E* or later, so the actual processing time may be shorter.

Common Processing Time

The common processing time is the following values by the total time for system overhead, system common processing 1, and system common processing 2. The common processing time depends on the type of task.

Type of task	Common processing times [μs] (reference values)		
	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Primary periodic task	150*1	240*2	300
Periodic task	10	32	82

*1. The processing time is 265 μs for a CPU Unit with unit version 1.01 or earlier.

*2. The processing time is 360 μs for a CPU Unit with unit version 1.01 or earlier.

A-6-2 Example of Calculating the Average Real Processing Time of a Task and Setting the Task Period

Example of Calculating the Average Real Processing Times of Tasks

If you are using an NJ501-□□□□ CPU Unit with a unit version of 1.02 and the hardware revision for the Unit is in blank, 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
	CJ-series Units (on CPU Rack)	• CJ1W-ID211 DC Input Unit: 1 • CJ1W-OD211 Transistor Output Unit: 1 • CJ1W-AD042 Analog Input Unit: 1 • CJ1W-DA021 Analog Output Unit: 1 • CJ1W-SCU42 Serial Communications Unit: 1 (Protocol macros are not used.)
User program	Language	Ladder diagrams
	Standard instruction configuration	Number of instructions: 5,000
	Arithmetic instructions for LREAL data	Number of instructions: 200
	Trigonometric instructions for LREAL data	Number of instructions: 100
Motion control processing	Number of axes	4

Note Total length of cables connected to the built-in EtherCAT port is 10 [m].

● I/O Refresh Time

I/O refresh overhead time:

The I/O refresh overhead time is 65 μs because the EtherCAT slaves and CJ-series Units are connected.

EtherCAT slave processing time:

The following table gives the Tout (output processing time per byte), Tin (input processing time per byte), and Tref (refresh processing time per byte) values when an NJ501-□□□□ CPU Unit is used.

Tout: Output processing time per byte [μs]	Tin: Input processing time per byte [μs]	Tref: Refresh processing time per byte [μs]	Tec: EtherCAT communications adjustment time [μs]
0.004	0.011	0.082	55

The following table gives the Dout (output processing data size), Din (input processing data size), and Dinout (larger of the input and output data size) values of the GX-ID1611 (Ver. 1.1) Input Slave, GX-OD1611 (Ver. 1.1) Output Slave, R88D-1SN□□□-ECT Servo Drives.

EtherCAT slave	Dout: Output processing data size in bytes	Din: Input processing data size in bytes	Dinout: Larger of the input and output data size
GX-ID1611 (Ver. 1.1)	0	3	3
GX-OD1611 (Ver. 1.1)	2	1	2
R88D-1SN□□□-ECT	23	26	26

Total number of bytes are given below for Dout, Din and Dinout.

Dout = 2 + 23 × 4 = 94 [byte]

Din = 3 + 1 + 26 × 4 = 108 [byte]

Dinout = 3 + 2 + 26 × 4 = 109 [byte]

From these values, the I/O refresh time is calculated by the following formula.

I/O refresh processing time

= EtherCAT slave processing time

= Tout × Dout + Tin × Din + Tref × Dinout + (1.24 × Snum + 0.01 × Clen - Tec)

= 0.004×94 + 0.011×108 + 0.082×109 + (1.24×6 + 0.01×10 - 55)

= 0.376 + 1.188 + 8.938 + (7.44 + 0.1 - 55)

= 10.502 + 0[*1]

≈ 11[μs]

Because the result that is calculated inside the parenthesis is a negative number, it is regarded as "0" μs.

CJ-series Unit processing time:

The following table gives the I/O refresh time per Unit for the CJ1W-ID211 DC Input Unit, CJ1W-OD211 Transistor Output Unit, CJ1W-AD042 Analog Input Unit, CJ1W-DA021 Analog Output Unit, and CJ1W-SCU42 Serial Communications Unit that are used on the CPU Rack.

Model number	I/O refresh time per Unit [μs]
CJ1W-ID211	1
CJ1W-OD211	1

Model number	I/O refresh time per Unit [μs]
CJ1W-AD042	24
CJ1W-DA021	24
CJ1W-SCU42	2.5

Because the number of Units is all one, use the following formula for the CJ-series Unit processing time.
CJ-series Unit processing time = Σ (I/O refresh time for each CJ-series Unit x Number of Units) - 110
= 1 x 1 + 1 × 1 + 24 × 1 + 24 × 1 + 2.5 × 1 - 110
= -57.5 [μs]

Because the result that is calculated from the above formula is a negative number, the CJ-series Unit processing time is regarded as 0 μs.

The following values of the I/O refresh overhead time, EtherCAT slave processing time, and CJ-series Unit processing time are found by the above calculations.

Item	Value [μs]
I/O refresh overhead time	65
EtherCAT slave processing time	11
CJ-series Unit processing time	0

From these values, the I/O refresh time is calculated by the following formula.
I/O refresh time = I/O refresh overhead time +
(Larger of the EtherCAT slave processing time and the CJ-series Unit processing time)
=65 + 11
=76 [μs]

● User Program Execution Time

The graphs show the following values.

- Standard instruction configuration
  From the graph of the execution time for standard ladder diagram instructions, the user program execution time of 5,000 instructions for the NJ501-□□□□ is 100 μs.
- Arithmetic instructions for LREAL data
  From the graph of the execution time for ladder diagram arithmetic and trigonometric instructions for LREAL data, the user program execution time of 200 instructions for the NJ501-□□□□ is 20 μs.
- Trigonometric instructions for LREAL data
  From the graph of the execution time for ladder diagram arithmetic and trigonometric instructions for LREAL data, the user program execution time of 100 instructions for the NJ501-□□□□ is 27 μs.

Therefore, the user program execution time is the total of the above values, which is given by the following formula.
User program execution time = 100 + 20 + 27
= 147 [μs]

● Motion Control Processing Time

From the graph of the execution time for motion control processing, the execution time of the motion control processing for four axes for the NJ501-□□□□ with a unit version 1.02 is read as 46 μs.

● Common Processing Time

Because the task is the primary periodic task, the common processing time for the NJ501-□□□□ is 150 μs.

Therefore, the average real processing time of the task is given by the following formula.

Average real processing time of task = I/O refresh processing time + User program execution time + Motion control processing time + Common processing time
=76 + 147 + 46 + 150
=419[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 x 1.1

Because the average real processing time of the task that is calculated above is 419 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 depend on the logic operations that are performed in the user program, the presence of communications commands and data links, on whether data tracing is performed, and on other factors. The task execution time for a periodic task depends on whether it is interrupted for the execution of tasks with higher execution priorities.
Use the physical Controller and verify the task execution time with the Task Execution Time Monitor.

A-7 System-defined Variables

System-defined variables are assigned specific functions by the system.
They are registered in the global variable table, or the local variable table for each POU, in advance. These variables cannot be changed. Some of the variables start with an underbar and some start with ”P_”.

Some of the system-defined variables are read-only and some are read/write.

You read and write the variables with the user program, with communications from external devices, with the Sysmac Studio, or with an NS/NA-series PT.
Basically, system-defined variables are classified according to the function modules.
The variables start with the following category names.

Function module	Category name
System-defined variables for the overall NJ/NX-series Controller	None
PLC Function Module	_PLC
	_CJB
	_APB
NX Bus Function Module	_NXB
Motion Control Function Module	_MC, _MC1, and _MC2
EtherCAT Master Function Module	_EC
EtherNet/IP Function Module	_EIP, _EIP1, and _EIP2
X Bus Function Module	_XBU

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 of the Variable name column is the unit version of the CPU Unit when the system-defined variable was added.

Precautions for Correct Use

There are system-defined variables that are not supported or differ in specifications such as the number of arrays. Refer to *A-8 Specifications for Individual System-defined Variables* on page A-163 for details on the specifications for individual system-defined variables.

A-7-1 System-defined Variables for the Overall NJ/NX-series Controller (No Category)

● Functional Classification: Clock

Variable name	Meaning	Function	Data type	Range of values	Reference
_CurrentTime	System Time	Contains the CPU Unit's internal clock data.	DATE_AND_TIME	• NX-series CPU Units DT#1970-01-01-00:00:00 to DT#2069-12-31-23:59:59 • NJ-series CPU Units DT#1970-01-01-00:00:00 to DT#2106-02-06-23:59:59	page A-163

● 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-164
_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-164
_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-164
_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-164

Variable name	Meaning	Function	Data type	Range of values	Reference
_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-165
_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-165
_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-165

● 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-166
_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-166

● Functional Classification: SD Memory Card

Variable name	Meaning	Function	Data type	Range of values	Reference
_Card1Ready	SD Memory Card Ready Flag	TRUE when the SD Memory Card is recognized. FALSE when the SD Memory Card is not recognized. TRUE: Can be used. FALSE: The Card cannot be used.	BOOL	TRUE or FALSE	page A-166
_Card1Protect	SD Memory Card Write Protected Flag	TRUE when the SD Memory Card is write-protected with the LOCK switch. TRUE: Write protected. FALSE: Not write protected.	BOOL	TRUE or FALSE	page A-167
_Card1Err	SD Memory Card Error Flag	TRUE when an unusable SD Memory Card is inserted or a format error occurs. TRUE: There is an error FALSE: There is no error	BOOL	TRUE or FALSE	page A-167
_Card1Access	SD Memory Card Access Flag	TRUE during SD Memory Card access. TRUE: Card is being accessed. FALSE: Card is not being accessed. The system updates the flag every 100 ms. Because of this, access to the SD Memory Card is shown by this flag with a delay of up to 100 ms. We therefore do not recommend the use of this variable in the user program.	BOOL	TRUE or FALSE	page A-167
_Card1Deteriorated	SD Memory Card Life Warning Flag	TRUE when the life of the SD Memory Card is exceeded. TRUE: The life of the Card has been exceeded. FALSE: The Card can still be used.	BOOL	TRUE or FALSE	page A-167
_Card1PowerFail	SD Memory Card Power Interruption Flag	TRUE when the power supply to the CPU Unit was interrupted during access to the SD Memory Card. TRUE: Power was interrupted during SD Memory Card access. FALSE: Normal	BOOL	TRUE or FALSE	page A-168
_Card1Capacity (Ver.1.21/Ver.1.32)	SD Memory Card Storage Capacity	Show the total capacity of the inserted SD Memory Card. The unit of capacity is MiB (1 MiB=1,048,576 bytes). The value is updated every 60 seconds. If the SD PWR indicator is not lit, such as when an SD Memory Card is not inserted, the value is "0".	UDINT	Depends on data type.	page A-168
_Card1Used (Ver.1.21/Ver.1.32)	SD Memory Card Storage Usage	Show the usage of the inserted SD Memory Card. The unit of capacity is MiB (1 MiB=1,048,576 bytes). The value is updated every 60 seconds. If the SD PWR indicator is not lit, such as when an SD Memory Card is not inserted, the value is "0".	UDINT	Depends on data type.	page A-168

Variable name / Member name	Meaning	Function	Data type	Range of values	Reference
_Card1BkupCmd (Ver.1.03)	SD Memory Card Backup Commands		_sBKUP_CMD		page A-168

Variable name Member name	Meaning	Function	Data type	Range of values	Reference
ExecBkup	Execute Backup Flag	Change this variable to TRUE to back up Controller data to the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-168
CancelBkup	Cancel Backup Flag	Change this variable to TRUE to cancel backing up data to the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-169
ExecVefy	Execute Verify Flag	Change this variable to TRUE to compare the Controller data to a backup file in the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-169
CancelVefy	Cancel Verify Flag	Change this variable to TRUE to cancel comparing the Controller data to a backup file in the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	BOOL	TRUE or FALSE	page A-169
DirName	Directory Name	Use this variable to specify the directory name in the SD Memory Card for which to back up data. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.	STRING[64]	Depends on data type.	page A-169
_Card1BkupSta (Ver.1.03)	SD Memory Card Backup Status		_sBKUP_STA		page A-170
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-170

Variable name / Member name	Meaning	Function	Data type	Range of values	Reference
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-170
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-170
_Card1VefySta (Ver.1.03)	SD Memory Card Verify Status		_sVEFY_STA		page A-170
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-170
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-171
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-171
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-171
_Card1PrgTransferCmd (Ver.1.11)	SD Memory Card Program Transfer Command		_sPRGTRANSFER_CMD		page A-171
Exec	Execute Program Transfer Flag	Change this variable to TRUE to transfer the data in a backup file on the SD Memory Card to the Controller by using the function to transfer programs from the SD Memory Card.	BOOL	TRUE or FALSE	page A-171

Variable name / Member name	Meaning	Function	Data type	Range of values	Reference
DirName	Directory Name	Use this variable to specify the directory name in the SD Memory Card in which the backup file to be transferred is stored.	STRING[64]	Depends on data type.	page A-172
Password	Password	Use this variable to specify the password that is used for verification when you start transferring the programs. The password is initialized every time you start transferring programs from the SD Memory Card.	STRING[33]	Depends on data type.	page A-172
TargetUserProgram	User Program and Settings Transfer Flag	Change this variable to TRUE to set a user program or setting as the transfer target. Always set this variable to TRUE for transferring programs from SD Memory Card.	BOOL	TRUE or FALSE	page A-172
TargetIPAdr	IP Address Transfer Flag	Change this variable to TRUE to include the IP address of the built-in EtherNet/IP port as the transfer target. The IP address means setting type, IP address, subnet mask, and default gateway.	BOOL	TRUE or FALSE	page A-172
TargetVariable	Present Values of Variables with the Retain Attribute Transfer Flag	Change this variable to TRUE to set the present values of variables with the Retain attribute as the transfer target.	BOOL	TRUE or FALSE	page A-173
TargetMemory	Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag	Change this variable to TRUE to set the present values of the memory used for CJ-series Units with the Retain attribute as the transfer target.	BOOL	TRUE or FALSE	page A-173
_Card1PrgTransferSta (Ver.1.11)	SD Memory Card Program Transfer Status		_sPRGTRANSFER_STA		page A-173
Done	Done Flag	TRUE when a program transfer is completed.	BOOL	TRUE or FALSE	page A-173
Active	Active Flag	TRUE when a program transfer is in progress.	BOOL	TRUE or FALSE	page A-173
Err	Error Flag	TRUE when a program transfer ended in an error.	BOOL	TRUE or FALSE	page A-174
_Card1RestoreCmd (Ver.1.14)	SD Memory Card Restore Command		_sRESTORE_CMD		page A-174
Exec	Execute Restore Flag	Change this variable to TRUE to restore the data in a backup file on the SD Memory Card to the Controller.	BOOL	TRUE or FALSE	page A-174
DirName	Directory Name	Use this variable to specify the directory name in the SD Memory Card in which the backup file to be restored by the system-defined variable is stored.	STRING[64]	Depends on data type.	page A-174
Password	Password	Use this variable to specify the password that is used for verification when you start the restore by the system-defined variable. The password is initialized every time when the restore by the system-defined variable is started.	STRING[33]	Depends on data type.	page A-174

Variable name / Member name	Meaning	Function	Data type	Range of values	Reference
_Card1RestoreSta (Ver.1.14)	SD Memory Card Restore Status		_sRES-TORE_STA		page A-175
Done	Done Flag	TRUE when a restore operation is completed.	BOOL	TRUE or FALSE	page A-175
Active	Active Flag	TRUE when a restore operation is in progress.	BOOL	TRUE or FALSE	page A-175
Err	Error Flag	TRUE when a restore operation ended in an error.	BOOL	TRUE or FALSE	page A-175

Variable name	Meaning	Function	Data type	Range of values	Reference
_Card1RestoreCmd-TargetUserProgram (Ver.1.14)	User Program and Settings Transfer Flag	Change this variable to TRUE to set a user program or setting for the restore by the system-defined variable as the transfer target. Always set this variable to TRUE for the restore by the system-defined variable.	BOOL	TRUE or FALSE	page A-175
_Card1RestoreCmd-TargetIPAdr (Ver.1.14)	IP Address Transfer Flag	Change this variable to TRUE to include the IP address of the built-in EtherNet/IP port for the restore by the system-defined variable as the transfer target. The IP address means setting type, IP address, subnet mask, and default gateway.	BOOL	TRUE or FALSE	page A-176
_Card1RestoreCmd-TargetVariable (Ver.1.14)	Present Values of Variables with the Retain Attribute Transfer Flag	Change this variable to TRUE to set the present values of variables with the Retain attribute for the restore by the system-defined variable as the transfer target.	BOOL	TRUE or FALSE	page A-176
_Card1RestoreCmd-TargetMemory (Ver.1.14)	Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag	Change this variable to TRUE to set the present values of the memory used for CJ-series Units with the Retain attribute for the restore by the system-defined variable as the transfer target.	BOOL	TRUE or FALSE	page A-176
_Card1RestoreCmd-TargetUnitConfig (Ver.1.14)	Unit and Slave Parameters Transfer Flag	Change this variable to TRUE to set the Unit and slave settings for the restore by the system-defined variable as the transfer target.	BOOL	TRUE or FALSE	page A-176
_Card1RestoreCmd-TargetAbsEncoder (Ver.1.14)	Absolute Encoder Home Offset Transfer Flag	Change this variable to TRUE to set the absolute encoder home offset for the restore by the system-defined variable as the transfer target.	BOOL	TRUE or FALSE	page A-177
_Card1RestoreCmd-TargetOpcuaSPF (Ver.1.17)	OPCUA Security Profile Transfer Flag	Change this variable to TRUE to set the OPC UA security profile for the restore by the system-defined variable as the transfer target.	BOOL	TRUE or FALSE	page A-177
_Card1RestoreCmd-TargetXBusUnitConfig (Ver.1.60) [NX502]	X Bus Unit Settings Transfer Flag	Change this variable to TRUE to set the X Bus Unit Settings for the restore by the system-defined variable as the transfer target.	BOOL	TRUE or FALSE	page A-177
_Card1RestoreCmd-TargetXBusUnitIPAdr (Ver.1.60) [NX502]	IP Address Settings Transfer Flag of EtherNet/IP Port on X Bus Unit	Change this variable to TRUE to include the IP address of the EtherNet/IP port on the X Bus Unit for the restore by the system-defined variable as the transfer target.	BOOL	TRUE or FALSE	page A-177

● Functional Classification: Backup

Variable name	Meaning	Function	Data type	Range of values	Reference
_BackupBusy (Ver.1.03)	Backup Function Busy Flag	TRUE when a backup, restoration, or verification is in progress.	BOOL	TRUE or FALSE	page A-178

● Functional Classification: Power Supply

Variable name	Meaning	Function	Data type	Range of values	Reference
_PowerOnHour	Total Power ON Time	Contains the total time that the power has been ON. Contains the total time that the CPU Unit has been ON in 1-hour increments. To reset this value, overwrite the current value with 0. The value is not updated after it reaches 4294967295. This variable is not initialized at startup.	UDINT	0 to 4294967295	page A-178
_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 to the CPU Unit 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-178
_RetainFail	Retention Failure Flag	TRUE at the following time (failure of retention during power interruptions). • When an error is detected in the battery-backup memory check at startup. FALSE at the following times (no failure of retention during power interruptions). • When no error is detected in the battery-backup memory check at startup. • When the user program is downloaded. • When the Clear All Memory operation is performed. **Note** When the absolute 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-179

● 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-179
P_Off	Always FALSE Flag	This flag is always FALSE.	BOOL	FALSE	page A-179
P_CY	Carry Flag	This flag is updated by some instructions.	BOOL	TRUE or FALSE	page A-179

Variable name	Meaning	Function	Data type	Range of values	Reference
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 initialization when the Controller begins operation. **Note** You cannot use this system-defined variable inside functions.	BOOL	TRUE or FALSE	page A-179
P_First_Run (Ver.1.08)	First Program Period Flag	This flag is TRUE for one task period after execution of the program starts. Use this flag to perform initial processing when execution of a program starts. **Note** You cannot use this system-defined variable inside functions.	BOOL	TRUE or FALSE	page A-180
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-180

● Functional Classification: Communications

Variable name	Meaning	Function	Data type	Range of values	Reference
_Port_numUsingPort	Number of Used Ports	Gives the number of internal logical ports that are currently used. You can use this variable when you debug the user program. **Note** You can use this system-defined variable only for NJ-series CPU Units.	USINT	0 to 32	page A-180
_Port_isAvailable	Network Communications Instruction Enabled Flag	Indicates whether there is an available internal logical port. TRUE when an internal logical port is available. Otherwise FALSE. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-180
_FINSTCPConnSta	FINS/TCP Connection Status	Gives the FINS/TCP connection status. **Note** You can use this system-defined variable only for NJ-series CPU Units.	WORD	16#0000 to 16#FFFF	page A-181

● Functional Classification: Version

Variable name	Meaning	Function	Data type	Range of values	Reference
_UnitVersion (Ver.1.08)	Unit Version	Contains the unit version of the CPU Unit. 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-181
_HardwareRevision (Ver.1.11)	Hardware Revision	Contains the hardware revision of the CPU Unit. Contains - if the hardware revision is in blank, and A to Z for other cases.	STRING[2]	- or A to Z	page A-181
_ProjectUnitVersion (Ver.1.40)	Project Unit Version	Contains the project unit version of the project in the CPU Unit. The integer part of the project unit version is stored in element number 0. The fractional part of the project unit version is stored in element number 1. Example 1) If the project unit version is 1.09, “1” is stored in element number 0 and “9” is stored in element number 1. Example 2) If the project unit version is 1.40, “1” is stored in element number 0 and “40” is stored in element number 1.	ARRAY[0..1] OF USINT	0 to 99	page A-181

● Functional Classification: Self-diagnosis

Variable name	Meaning	Function	Data type	Range of values	Reference
_SelfTest_HighTemperature (Ver.1.10)	CPU Unit High Temperature Flag	TRUE when the internal temperature of the CPU Unit is too high. **Note** Always FALSE for the NX102 CPU Unit and NX1P2 CPU Unit.	BOOL	TRUE or FALSE	page A-182
_SelfTest_LowBattery (Ver.1.10)	Low Battery Flag	TRUE when the battery is disconnected or the battery voltage is dropped.	BOOL	TRUE or FALSE	page A-182
_SelfTest_LowFanRevolution (Ver.1.10)	Low FAN Revolution Flag	TRUE when the fan is disconnected or the rotation speed of a fan is decreased. **Note** Always FALSE for the NX102 CPU Unit, NX1P2 CPU Unit, and NJ-series CPU Unit.	BOOL	TRUE or FALSE	page A-182
_StorageDeteriorated (Ver.1.60) [NJ-series, NX502, NX102, NX1P2] (Ver.1.32) [NX701]	Non-volatile Memory Life Warning Flag	TRUE when the number of deletions for non-volatile memory or the number of bad blocks in memory exceeds the warning value.	BOOL	TRUE or FALSE	page A-182

● Functional Classification: PLC Built-in

Variable name	Meaning	Function	Data type	Range of values	Reference
_DeviceOutHoldCfg (Ver.1.13)	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	16#0000 to 16#FFFF	page A-183
_DeviceOutHoldStatus (Ver.1.13)	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-183

A-7-2 PLC Function Module, Category Name: _PLC

● Functional Classification: Debugging

Variable name Member name	Meaning	Function	Data type	Range of values	Reference
_PLC_TraceSta[0..3]			_sTRACE_STA		page A-183
.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-183
.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-184
.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-184
.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-184

● 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-184

● Functional Classification: Option Boards

Variable name Member name	Meaning	Function	Data type	Range of values	Reference
_PLC_OptBoardSta (Ver. 1.13)	Option Board Status	Contains the status of Option Boards. This variable is commonly used regardless of the models of Option Boards. The array element 1 corresponds to the option board slot 1 and array element 2 corresponds to the option board slot 2. **Note** You can use this system-defined variable only for NX1P2 CPU Units.	ARRAY[1..2] OF _sOPTBOARD_STA		page A-185
isDetect	Option Board Mounted	Indicates an Option Board is mounted to the option board slot. TRUE: Mounted. FALSE: Not mounted. **Note** You can use this system-defined variable only for NX1P2 CPU Units.	BOOL	TRUE or FALSE	page A-185
Run	Option Board Normal Operation	Indicates whether an Option Board is normally operating. To use device variables or communications instructions for an Option Board, program this member as an interlock condition in the user program. TRUE: Normally operating. FALSE: Initializing, changing settings, or an error occurred. **Note** You can use this system-defined variable only for NX1P2 CPU Units.	BOOL	TRUE or FALSE	page A-185
Error	Option Board Error	Indicates whether an Option Board error occurred. TRUE: An error occurred. FALSE: An error not occurred. **Note** You can use this system-defined variable only for NX1P2 CPU Units.	BOOL	TRUE or FALSE	page A-185

Variable name / Member name	Meaning	Function	Data type	Range of values	Reference
_PLC_OptSerialErr-Sta (Ver.1.13)	Serial Option Board Error Status	Contains the error status of a transmission error for the Serial Communications Option Board. When the Serial communications mode of a Serial Communications Option Board is only set to Host Link (FINS), the value of each member is updated. Other than the above setting, the values of all members are FALSE. The array element 1 corresponds to the option board slot 1 and array element 2 corresponds to the option board slot 2. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.	ARRAY[1..2] OF _sOPT-SERIAL-ERR_STA		page A-186
Error	Transmission Error	Indicates whether a transmission error occurred. TRUE: A parity error, framing error, or an overrun error occurred. FALSE: FALSE due to one of the following causes. • A parity error, framing error, or an overrun error not occurred. • A port is restarted. • The Serial communications mode is not set to Host Link (FINS). • The Option Board that is mounted is not a Serial Communications Option Board. • The Option Board is not mounted. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.	BOOL	TRUE or FALSE	page A-186

Variable name Member name	Meaning	Function	Data type	Range of values	Reference
ParityErr	Parity Error	Indicates whether a parity error occurred. If this error occurs, it means that the serial communications settings may not apply to the remote device to connect. TRUE: A parity error occurred. FALSE: FALSE due to one of the following causes. • A parity error not occurred. • A port is restarted. • The Serial communications mode is not set to Host Link (FINS). • The Option Board that is mounted is not a Serial Communications Option Board. • The Option Board is not mounted. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.	BOOL	TRUE or FALSE	page A-187
FramingErr	Framing Error	Indicates whether a framing error occurred. If this error occurs, it means that the serial communications settings may not apply to the remote device to connect. TRUE: A framing error occurred. FALSE: FALSE due to one of the following causes. • A framing error not occurred. • A port is restarted. • The Serial communications mode is not set to Host Link (FINS). • The Option Board that is mounted is not a Serial Communications Option Board. • The Option Board is not mounted. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.	BOOL	TRUE or FALSE	page A-187

Variable name Member name	Meaning	Function	Data type	Range of values	Reference
OverRun	Overrun Error	Indicates whether an overrun error occurred. If this error occurs, it means that the baud rate of an Option Board may be too large. TRUE: An overrun error occurred. FALSE: FALSE due to one of the following causes. • An overrun error not occurred. • A port is restarted. • The Serial communications mode is not set to Host Link (FINS). • The Option Board that is mounted is not a Serial Communications Option Board. • The Option Board is not mounted. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.	BOOL	TRUE or FALSE	page A-188

● Functional Classification: Safety Data Logging

Variable name Member name	Meaning	Function	Data type	Range of values	Reference
_PLC_SFLogSta (Ver.1.31)	Safety Data Logging Status	Stores the status of safety data logging. Element number 0 corresponds to Logging Setting Number 1. Element number 1 corresponds to Logging Setting Number 2. **Note** You can use this system-defined variable only for the NX502 CPU Units and NX102 CPU Units.	ARRAY[0..1] OF _sSFLOG_STA		page A-188
.IsStart	Safety Data Logging Busy Flag	TRUE when safety data logging starts. **Note** You can use this system-defined variable only for the NX502 CPU Units and NX102 CPU Units.	BOOL	TRUE or FALSE	
.IsComplete	Safety Data Logging Completed Flag	TRUE when logging stops. FALSE when the next logging starts. When this flag is TRUE, it means that the logging has completed. **Note** You can use this system-defined variable only for the NX502 CPU Units and NX102 CPU Units.	BOOL	TRUE or FALSE	
.IsOutput	Log File Output Completed Flag	TRUE when the log file is output. FALSE when the next logging starts. **Note** You can use this system-defined variable only for the NX502 CPU Units and NX102 CPU Units.	BOOL	TRUE or FALSE	

A-7-3 PLC Function Module, Category Name: _CJB

● Functional Classification: I/O Bus Status

Variable name	Meaning	Function	Data type	Range of values	Reference
_CJB_MaxRackNo	Largest Rack Number	Contains the largest rack number of the Expansion Racks that are detected by the Controller. **Note** You can use this system-defined variable only for NJ-series CPU Units.	UINT	0 to 3 0: Only CPU Rack.	page A-189
_CJB_MaxSlotNo	Largest Slot Number	Contains one higher than the largest slot number with a CJ-series Unit on each of the Racks that are detected by the Controller. **Note** You can use this system-defined variable only for NJ-series CPU Units.	ARRAY [0..3] OF UINT	0 to 10 0: No CJ-series Unit mounted.	page A-189

● Functional Classification: I/O Bus Errors

Variable name	Meaning	Function	Data type	Range of values	Reference
_CJB_ErrSta	I/O Bus Error Status	Gives the I/O bus error status. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NJ-series CPU Units.	WORD	16#0000 to 16#C0F0	page A-190
_CJB_MstrErrSta	I/O Bus Master Error Status	Gives the I/O bus master error status. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NJ-series CPU Units.	WORD	16#0000 to 16#00F0	page A-190
_CJB_UnitErrSta	I/O Bus Unit Error Status	Gives the error status of the I/O Bus Unit. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device. Refer to information on the meanings of the error status bits at the end of this appendix for details. **Note** You can use this system-defined variable only for NJ-series CPU Units.	ARRAY [0..3, 0..9] OF WORD	16#0000 to 16#80F0	page A-190
_CJB_InRespTm	Basic Input Unit Input Response Times	Contains the response times of the Basic Input Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.	ARRAY [0..3, 0..9] OF UINT	0 to 320	page A-191

● Functional Classification: Auxiliary Area Bits for CJ-series Units

Variable name	Meaning	Function	Data type	Range of values	Reference
_CJB_IOUnitInfo	Basic I/O Unit Information	Shows the status of the Basic I/O Unit alarm output (load short-circuit protection). TRUE: Load short-circuit FALSE: No load short-circuit **Note** You can use this system-defined variable only for NJ-series CPU Units.	ARRAY [0..3, 0..9, 0..7] OF BOOL	TRUE or FALSE	page A-191
_CJB_CBU00InitSta to _CJB_CBU15InitSta	CPU Bus Unit Initializing Flags	The corresponding variable is TRUE during initialization of the CPU Bus Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-191
_CJB_SIO00InitSta to _CJB_SIO95InitSta	Special I/O Unit Initializing Flags	The corresponding variable is TRUE during initialization of the Special I/O Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-192
_CJB_CBU00Restart to _CJB_CBU15Restart	CPU Bus Unit Restart Bits	The CPU Bus Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the CPU Bus Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Bit to TRUE with an instruction, the restart process begins from refresh processing in the next task period. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-192
_CJB_SIO00Restart to _CJB_SIO95Restart	Special I/O Unit Restart Bits	The Special I/O Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the Special I/O Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Bit to TRUE with an instruction, the restart process begins from refresh processing in the next task period. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-192

Variable name	Meaning	Function	Data type	Range of values	Reference
_CJB_SCU00P1ChgSta _CJB_SCU00P2ChgSta to	Serial Communications Unit 0, Port 1/2 Settings Changing Flags	TRUE when the parameters of the specified port are being changed. FALSE after the parameters are changed. It is also possible for the user to indicate a change in serial port settings by turning ON the corresponding flag through the execution of an instruction or a user operation. **Note** You can use this system-defined variable only for NJ-series CPU Units.	BOOL	TRUE or FALSE	page A-193
_CJB_SCU15P1ChgSta _CJB_SCU15P2ChgSta	Serial Communications Units 1 to 15, Port 1/2 Settings Changing Flags		BOOL	TRUE or FALSE	page A-193

A-7-4 PLC Function Module, Category Name: _APB

● Functional Classification: Automation Playback

Variable name Member name	Meaning	Function	Data type	Range of values	Reference
_APB_Status (Ver.1.63)	APB Service Status	Indicates the settings and status of the APB service. **Note** You can use this system-defined variable only for the NX502 CPU Unit.	_sAPB_STATUS		page A-193
.Status	APB Service Operating Status	Indicates the operating status of the APB service.	UINT	0 to 4	page A-193
.LatestLogName	Latest Variable Log File Name	Indicates the file name of the most recent variable log that was saved to storage.	STRING[32]	Depends on data type.	page A-193
.LatestLogTime	Latest Variable Log Output Date and Time	Indicates the most recent date and time when the variable log was output and saved to storage. Note that the value of this member may not match the date and time included in the file name.	DATE_AND_TIME	Depends on data type.	page A-194
.ActiveSampling-SettingName	APB Project Backup Finished	Indicates the status of backing up the project with automation playback.	BOOL	TRUE or FALSE	page A-194
.ActiveSampling-SettingNumber	Active Sampling Setting Number	Indicates the active collection setting number in automation playback.	USINT	1 or 2	page A-194
.ActiveSampling-SettingName	Active Sampling Setting Name	Indicates the name of active collection setting in automation playback.	STRING[32]	Depends on data type.	page A-194
.LatestErrorID	APB Instruction Error Code	Indicates the error code of the last automation playback-related instruction detected.	WORD	Depends on data type.	page A-194
.LatestEventCode	APB Service Event Code	Indicates the last automation playback-related event code detected.	DWORD	Depends on data type.	page A-195
_APB_LogStatus (Ver.1.63)	APS Log Output Status	Indicates the variable log output settings and status. This variable is a two-dimensional array variable of structure type. The primary subscript of the array represents the number of collection setting, and the secondary subscript of the array corresponds to the number of the variable log output settings. **Note** You can use this system-defined variable only for the NX502 CPU Unit.	ARRAY[1..2] [1..2] OF _sAPB_LOG_STATUS		page A-195

Variable name Member name	Meaning	Function	Data type	Range of values	Reference
.Status	Variable Log Output Setting Status	Indicates the operating status of the variable log output settings.	UINT	0 to 6	page A-195
.SettingName	Variable Log Output Setting Name	Indicates the setting name of the variable log output settings.	STRING[32]	Depends on data type.	page A-195
.SettingType	Variable Log Output Setting Type	Indicates the method of variable log output settings.	UINT	0 or 1	page A-195

A-7-5 NX Bus Function Module, Category Name: _NXB

● Functional Classification: NX Bus Function Module Status

Variable name	Meaning	Function	Data type	Range of values	Reference
_NXB_MaxUnitNo (Ver.1.13)	Largest Unit Number	Contains the largest NX Unit number of the NX Units on the CPU Unit that are detected by the NX Bus Function Module. If the Unit configuration information is registered by the Sysmac Studio, the value will be the largest NX Unit number of the registered Unit configuration. Units that are set as unmounted Units are also included. If the Unit configuration information is not registered by the Sysmac Studio, the value will be the largest Unit number of an actual Unit configuration. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.	UINT	0 to 32*1 0: No NX Unit mounted.	page A-196
_NXB_UnitIOActiveTbl (Ver.1.13)	NX Unit I/O Data Active Status	Indicates whether the I/O data in the NX Units on the CPU Unit is valid. This status is given as an array of BOOL data. The subscript of the array corresponds to the NX Unit number. A subscript of 0 indicates the NX Bus Function Module and it is always TRUE. TRUE: The I/O data in the NX Unit is valid. FALSE: The I/O data in the NX Unit is invalid. The status is FALSE for NX Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.	ARRAY [0..32] OF BOOL*2	TRUE or FALSE	page A-196

Variable name	Meaning	Function	Data type	Range of values	Reference
_NXB_UnitMsgActiveTbl (Ver.1.13)	NX Unit Message Enabled Status	Indicates whether the NX Units on the CPU Unit can process message communications. This status is given as an array of BOOL data. The subscript of the array corresponds to the NX Unit number. A subscript of 0 indicates the NX Bus Function Module and it is always TRUE. TRUE: Message communications possible. FALSE: Message communications not possible. The status is FALSE for NX Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.	ARRAY [0..32] OF BOOL*2	TRUE or FALSE	page A-197
_NXB_UnitRegTbl (Ver.1.13)	NX Unit Registration Status	Indicates whether the NX Units on the CPU Unit are registered in the Unit configuration. This status is given as an array of BOOL data. The subscript of the array corresponds to the NX Unit number. A subscript of 0 indicates the NX Bus Function Module. TRUE: Registered. FALSE: Not registered. If the Unit configuration information is not registered by the Sysmac Studio, the status is FALSE for all Units. The status is TRUE for NX Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.	ARRAY [0..32] OF BOOL*2	TRUE or FALSE	page A-197

*1. For the NX1P2 CPU Units, the range of values is 0 to 8.
For the NX502 CPU Units, the range of values is 0 to 63.

*2. For the NX1P2 CPU Units, the data type is ARRAY [0..8] OF BOOL.
For the NX502 CPU Units, the data type is ARRAY [0..63] OF BOOL.

● Functional Classification: NX Bus Function Module Errors

Variable name	Meaning	Function	Data type	Range of values	Reference
_NXB_ErrSta (Ver.1.13)	NX Bus Function Module Error Status	Gives the NX Bus Function Module error status. This system-defined variable provides the collective status of the NX Bus Function Module Master Error Status and NX Bus Function Module Unit Error Status for all NX Units. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.	WORD	16#0000 to 16#40F2	page A-198
_NXB_MstrErrSta (Ver.1.13)	NX Bus Function Module Master Error Status	Gives the status of errors that are detected in the NX Bus Function Module of the CPU Unit. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.	WORD	16#0000 to 16#40F2	page A-198

Variable name	Meaning	Function	Data type	Range of values	Reference
_NXB_UnitErrStaTbl (Ver.1.13)	NX Bus Function Module Unit Error Status	Gives the status of errors that are detected in the NX Bus Function Module of the CPU Unit. This status is given as an array of WORD data. The subscript of the array corresponds to the NX Unit number. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.	ARRAY [1..32] OF WORD*1	16#0000 to 16#40F2	page A-199
_NXB_UnitErr-FlagTbl (Ver.1.13)	NX Unit Error Status	Indicates whether errors occurred in the NX Unit on the CPU Unit. This status is given as an array of BOOL data. The subscript of the array corresponds to the NX Unit number. A subscript of "0" indicates the NX Bus Function Module and whether an event occurred that is detected by the NX Bus Function Module. TRUE: Error. FALSE: No error. The status is "FALSE" for NX Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.	ARRAY [0..32] OF BOOL*2	TRUE or FALSE	page A-199

*1. For the NX1P2 CPU Units, the data type is ARRAY [1..8] OF WORD.
For the NX502 CPU Units, the data type is ARRAY [1..63] OF WORD.

*2. For the NX1P2 CPU Units, the data type is ARRAY [0..8] OF BOOL.
For the NX502 CPU Units, the data type is ARRAY [0..63] OF BOOL.

A-7-6 X Bus Function Module, Category Name: _XBU

● Functional Classification: X Bus Function Module Status

Variable name	Meaning	Function	Data Type	Range of values	Reference
_XBU_RegTbl	X Bus Unit Registration Status	This status tells whether the X Bus Units are registered in the Unit configuration. This status is given as an array of BOOL data. The suffix of the array represents the X Bus Unit number. A suffix of 0 is given to the X Bus Function Module. TRUE: Registered. FALSE: Not registered. If the Unit configuration information is not registered from the Sysmac Studio, the status is FALSE for all Units. The status is TRUE for X Bus Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units.	ARRAY [0..4] OF BOOL	TRUE or FALSE	page A-200

● Functional Classification: X Bus Function Module Errors

Variable name	Meaning	Function	Data type	Range of values	Reference
_XBU_ErrSta	X Bus Function Module Error Status	Gives the status of errors that are detected in the X Bus Function Module. Gives the collective value of the *X Bus Function Module Master Error Status* and *X Bus Function Module Unit Error Status* for all X Bus Units. **Note 1.** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note 2.** You can use this system-defined variable only for the NX502 CPU Units.	WORD	16#0000 to 16#40F2	page A-200

Variable name	Meaning	Function	Data type	Range of values	Reference
_XBU_MstrErr	X Bus Function Module Master Error Status	Gives the status of errors that are detected in the X Bus master. **Note 1.** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note 2.** You can use this system-defined variable only for the NX502 CPU Units.	WORD	16#0000 to 16#40F2	page A-200
_XBU_UnitErr	X Bus Function Module Unit Error Status	Gives the collective value of all *X Bus Function Module Unit Error Status Table*. **Note 1.** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note 2.** You can use this system-defined variable only for the NX502 CPU Units.	WORD	16#0000 to 16#40F2	page A-200
_XBU_UnitErrTbl	X Bus Function Module Unit Error Status Table	Gives the status of errors that are detected in the X Bus Unit. The suffix of the array represents the X Bus Unit number. The status is "16#0000" for X Bus Units that are set as unmounted Units. **Note 1.** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note 2.** You can use this system-defined variable only for the NX502 CPU Units.	ARRAY [1..4] OF WORD	16#0000 to 16#40F2	page A-201

A-7-7 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 *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#40F0	page A-201
_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 *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#00F0	page A-201
_MC_AX_ErrSta	Axis Error Status	Shows the error status for each axis. The status of up to 256 axes*1 is shown. You can use this variable directly in the user program. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits.	Array [0..255] OF WORD*1	16#0000 to 16#00F0	page A-202
_MC_GRP_ErrSta	Axes Group Error Status	Shows the error status for each axes group. The error status for up to 64 axes groups*2 is shown. You can use this variable directly in the user program. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits.	Array [0..63] OF WORD*2	16#0000 to 16#00F0	page A-202
_MC_COM	Common Variable	Shows the status that is common to the Motion Control Function Module. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members.	_sCOMMON_REF	---	page A-202
_MC_GRP	Axes Group Variables	NX701 CPU Units: Used to specify axes groups and shows multi-axes coordinated control status, and multi-axes coordinated control settings for motion control instructions used for motion control 1. NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units: Used to specify axes groups and shows multi-axes coordinated control status, and multi-axes coordinated control settings for motion control instructions. When you create an *axes group* on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members.	ARRAY[0..63] OF _sGROUP_REF*3	---	page A-203

Variable name	Meaning	Function	Data type	Range of values	Reference
_MC1_GRP	Axes Group Variables	Used to specify axes groups and shows multi-axes coordinated control status, and multi-axes coordinated control settings for motion control instructions used for motion control 1. When you create an *axes group* on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members. **Note** You can use this system-defined variable only for NX701 CPU Units. You can access the same values of *_MC1_GRP* and *_MC_GRP* if the array element numbers of them are the same.	ARRAY[0..63] OF _sGROUP_REF	---	page A-203
_MC2_GRP	Axes Group Variables	Used to specify axes groups and shows multi-axes coordinated control status, and multi-axes coordinated control settings for motion control instructions used for motion control 2. When you create an *axes group* on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members. **Note** You can use this system-defined variable only for NX701 CPU Units.	ARRAY[0..63] OF _sGROUP_REF	---	page A-203
_MC_AX	Axis Variables	NX701 CPU Units: Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 1. NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units: Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions. When you create an *axis* on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members.	ARRAY[0..255] OF _sAXIS_REF*4	---	page A-204

Variable name	Meaning	Function	Data type	Range of values	Reference
_MC1_AX	Axis Variables	Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 1. When you create an *axis* on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members. **Note** You can use this system-defined variable only for NX701 CPU Units. You can access the same values of *_MC1_AX* and *_MC_AX* if the array element numbers of them are the same.	ARRAY[0..255] OF _sAXIS_REF	---	page A-204
_MC2_AX	Axis Variables	Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 2. When you create an *axis* on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members. **Note** You can use this system-defined variable only for NX701 CPU Units.	ARRAY[0..255] OF _sAXIS_REF	---	page A-204

*1. For NX502 CPU Units, the error status of up to 128 axes is shown and the data type is ARRAY [0..127] OF WORD.
For the NX102 CPU Units and NX1P2 CPU Units, the error status of up to 16 axes is shown and the data type is ARRAY [0..15] OF WORD.
For NJ-series CPU Units, the error status of up to 64 axes is shown and the data type is ARRAY [0..63] OF WORD.

*2. For NX502 CPU Units, the error status of up to 32 axes groups is shown and the data type is ARRAY [0..31] OF WORD.
For the NX102 CPU Units and NX1P2 CPU Units, the error status of up to 8 axes groups is shown and the data type is ARRAY [0..7] OF WORD.
For NJ-series CPU Units, the error status of up to 32 axes groups is shown and the data type is ARRAY [0..31] OF WORD.

*3. For NX502 CPU Units, the error status of up to 32 axes groups is shown and the data type is ARRAY [0..31] OF _sGROUP_REF.
For the NX102 CPU Units and NX1P2 CPU Units, the error status of up to 8 axes groups is shown and the data type is ARRAY [0..7] OF _sGROUP_REF.
For NJ-series CPU Units, the error status of up to 32 axes groups is shown and the data type is ARRAY [0..31] OF _sGROUP_REF.

*4. For NX502 CPU Units, the error status of up to 64 axes is shown and the data type is ARRAY [0..127] OF _sAXIS_REF.
For the NX102 CPU Units and NX1P2 CPU Units, the error status of up to 16 axes is shown and the data type is ARRAY [0..15] OF _sAXIS_REF.
For NJ-series CPU Units, the error status of up to 64 axes is shown and the data type is ARRAY [0..63] OF _sAXIS_REF.

A-7-8 EtherCAT Master Function Module, Category Name: _EC

● Functional Classification: EtherCAT Communications Errors

Variable name	Meaning	Function	Data type	Range of values	Reference
_EC_ErrSta	EtherCAT Error	This system-defined variable provides the collective status of errors in the EtherCAT Master Function Module. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#40F0	page A-205
_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 *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#00F0	page A-205
_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 *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#00F0	page A-205
_EC_SlavErr	Slave Error	This system-defined variable provides the collective status of all the error status for EtherCAT slaves. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#00F0	page A-205.
_EC_SlavErrTbl	Slave Error Table	This system-defined variable gives the error status for each EtherCAT slave. This error status shows status of error in each slave in the actual system configuration. This variable array indicates slaves in which there are errors. Status is provided for each EtherCAT slave node address (1 to 512)*1. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits.	Array [1..512] OF WORD*1	16#0000 to 16#00F0	page A-206
_EC_MacAdrErr	MAC Address Error	TRUE if the MAC Address Error (14400000 hex) event occurred.	BOOL	TRUE or FALSE	page A-206
_EC_LanHwErr	Communications Controller Error	TRUE if the Communications Controller Error (047C0000 hex) event occurred.	BOOL	TRUE or FALSE	page A-206
_EC_LinkOffErr	Link OFF Error	TRUE if the Link OFF Error (84200000 hex) event occurred.	BOOL	TRUE or FALSE	page A-206
_EC_NetCfgErr	Network Configuration Information Error	TRUE if the Network Configuration Information Error (34400000 hex) event occurred.	BOOL	TRUE or FALSE	page A-206

Variable name	Meaning	Function	Data type	Range of values	Reference
_EC_NetCfgCmpErr	Network Configuration Verification Error	TRUE if one of the following events occurred. • Network Configuration Verification Error (84220000 hex) • Network Configuration Verification Error (Slave Unconnected) (84380000 hex) • Network Configuration Verification Error (Unnecessary Slave Connected) (84320003 hex) • Network Configuration Verification Error (Mismatched Slave) (84330004 hex) • Network Configuration Verification Error (Incorrect Ring Wiring) (843A0000 hex)	BOOL	TRUE or FALSE	page A-207
_EC_NetTopologyErr	Network Configuration Error	TRUE if one of the following events occurred. • Network Configuration Error (84210000 hex) • Incorrect Wiring Detected (843C0000 hex)	BOOL	TRUE or FALSE	page A-207
_EC_PDCommErr	Process Data Communications Error	TRUE if one of the following events occurred. • Process Data Communications Error (842C0000 hex) • Illegal Slave Disconnection Detected (84310002 hex) • Slave PDI WDT Error Detected (84340000 hex)	BOOL	TRUE or FALSE	page A-207
_EC_PDTimeoutErr	Process Data Reception Timeout Error	TRUE if the Process Data Reception Timeout (842B0000 hex) event occurred.	BOOL	TRUE or FALSE	page A-207
_EC_PDSendErr	Process Data Transmission Error	TRUE if the Process Data Transmission Error (84290000 hex) event occurred.	BOOL	TRUE or FALSE	page A-207
_EC_SlavAdrDupErr	Slave Node Address Duplicated Error	TRUE if the Slave Node Address Duplicated (24200000 hex) event occurred.	BOOL	TRUE or FALSE	page A-208
_EC_SlavInitErr	Slave Initialization Error	TRUE if one of the following events occurred. • Slave Initialization Error (84230000 hex) • Slave State Transition Failed (84300001 hex)	BOOL	TRUE or FALSE	page A-208
_EC_SlavAppErr	Slave Application Error	TRUE if one of the following events occurred. • Slave Application Error (84280000 hex) • Slave AL Status Error Detected (84360000 hex)	BOOL	TRUE or FALSE	page A-208
_EC_MsgErr	EtherCAT Message Error	TRUE if one of the following events occurred. • EtherCAT Message Error (842D0000 hex) • Illegal Mailbox Received (84350000 hex)	BOOL	TRUE or FALSE	page A-208
_EC_SlavEmergErr	Emergency Message Detected	TRUE if the Emergency Message Detected (64200000 hex) event occurred.	BOOL	TRUE or FALSE	page A-208

Variable name	Meaning	Function	Data type	Range of values	Reference
_EC_IndataInvalidErr (Ver.1.13)	Input Process Data Invalid Error	TRUE if the Input Process Data Invalid Error (842F0000 hex) event occurred.	BOOL	TRUE or FALSE	page A-209
_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*1	TRUE or FALSE	page A-209
_EC_CycleExceeded	EtherCAT Communications Cycle Exceeded	TRUE if the EtherCAT Communications Cycle Exceeded (34410000 hex) event occurred. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-209

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the node address is 1 to 192 and the data type is ARRAY [1..192] OF WORD.
For the NX502 CPU Units, the node address is 1 to 256 and the data type is Array [1..256] OF WORD.

Additional Information

Typical Relationships for the EtherCAT Error Flags

Variable name	Meaning	Variable name	Meaning	Variable name	Meaning	Event Levels
_EC_ErrSta	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*1	TRUE or FALSE	page A-209
_EC_EntrySlavTbl	Network Connected Slave Table	This table indicates which slaves are connected to the network. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave has entered the network.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-210
_EC_MBXSlavTbl	Message Communications Enabled Slave Table	This table indicates the slaves that can perform message communications. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if message communications are enabled for it (pre-operational, safe-operation, or operational state). **Note** Use this variable to confirm that message communications are possible for the relevant slave before you execute message communications with an EtherCAT slave.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-210
_EC_PDSlavTbl	Process Data Communicating Slave Table	This table indicates the slaves that are performing process data communications. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if process data of the corresponding slave is enabled (operational) for both slave inputs and outputs. **Note** Use this variable to confirm that the data for the relevant slave is valid before controlling an EtherCAT slave.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-210
_EC_DisconnSlavTbl	Disconnected Slave Table	Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave was disconnected.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-211
_EC_DisableSlavTbl	Disabled Slave Table	Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave is disabled.	Array [1..512] OF BOOL*1	TRUE or FALSE	page A-211
_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-211
_EC_PktMonStop	Packet Monitoring Stopped	TRUE when packet monitoring is stopped.	BOOL	TRUE or FALSE	page A-211
_EC_LinkStatus	Link Status	TRUE if the Communications Controller link status is Link ON.	BOOL	TRUE or FALSE	page A-212

Variable name	Meaning	Function	Data type	Range of values	Reference
_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-212
_EC_InDataInvalid	Input Data Invalid	TRUE when process data communications established in the primary periodic task are not normal and the input data is not valid.	BOOL	TRUE or FALSE	page A-212
_EC_InData1Invalid	Input Data1 Invalid	TRUE when process data communications established in the primary periodic task are not normal and the input data is not valid. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-212
_EC_InData2Invalid	Input Data2 Invalid	TRUE when process data communications established in the priority-5 periodic task are not normal and the input data is not valid. **Note** You can use this system-defined variable only for NX-series CPU Units. **Note** This variable is always TRUE for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.	BOOL	TRUE or FALSE	page A-213
_EC_RingBreaking (Ver.1.40)	Ring Disconnection	TRUE when all slaves in the ring topology except for disabled slaves are connected and if there is only one point of disconnection in the communications cables in the ring topology.	BOOL	TRUE or FALSE	page A-213
_EC_RingBreakNodeAdr (Ver.1.40)	Slave Node Address Before Ring Disconnection	When the *_EC_RingBreaking* (Ring Disconnection) system-defined variable is TRUE, the slave node address before point of disconnection is stored. When the *_EC_RingBreaking* (Ring Disconnection) system-defined variable is FALSE, “0” is stored.	UINT	0 to maximum number of node addresses	page A-213

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the data type is ARRAY [1..192] OF BOOL.
For the NX502 CPU Units, the data type is Array [1..256] OF BOOL.

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 (Ver.1.11)	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-213

Variable name	Meaning	Function	Data type	Range of values	Reference
_EC_StatisticsLog-CycleSec (Ver.1.11)	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-214
_EC_Statistic-sLogBusy (Ver.1.11)	Diagnosis/Statistics Log Busy	TRUE while the diagnosis/statistics log operation is in progress.	BOOL	TRUE or FALSE	page A-214
_EC_StatisticsLogErr (Ver.1.11)	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 SD Memory Card is fully used. • The SD Memory Card is write-protected. • There is no 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-214

A-7-9 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	EtherNet/IP Error	This is the error status variable for the built-in EtherNet/IP port. NX-series CPU Units: Represents the collective status of the following error flags. • _EIP1_PortErr (Communications Port1 Error) • _EIP2_PortErr (Communications Port2 Error) • _EIP1_CipErr (CIP Communications1 Error) • _EIP2_CipErr (CIP Communications2 Error) • _EIP_TcpAppErr (TCP Application Communications Error) NJ-series CPU Units: Represents the collective status of the following error flags. • _EIP_PortErr (Communications Port Error) • _EIP_CipErr (CIP Communications Error) • _EIP_TcpAppErr (TCP Application Communications Error) **Note** Refer to *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#00F0	page A-215

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_PortErr	Communications Port Error	This is the error status variable for the communications port. NX-series CPU Units: Represents the collective status of the following error flags. • _EIP1_MacAdrErr (Port1 MAC Address Error) • _EIP1_LanHwErr (Port1 Communications Controller Error) • _EIP1_EtnCfgErr (Port1 Basic Ethernet Setting Error) • _EIP1_IPAdrCfgErr (Port1 IP Address Setting Error) • _EIP1_IPAdrDupErr (Port1 IP Address Duplication Error) • _EIP1_BootpErr (Port1 BOOTP Server Error) • _EIP1_DhcpErr (Port1 DHCP Server Error) • _EIP_DNSCfgErr (DNS Setting Error) • _EIP_DNSSrvErr (DNS Server Connection Error) • _EIP_IPRTblErr (IP Route Table Error) NJ-series CPU Units: Represents the collective status of the following error flags. • _EIP_MacAdrErr (MAC Address Error) • _EIP_LanHwErr (Communications Controller Error) • _EIP_EtnCfgErr (Basic Ethernet Setting Error) • _EIP_IPAdrCfgErr (IP Address Setting Error) • _EIP_IPAdrDupErr (IP Address Duplication Error) • _EIP_BootpErr (BOOTP Server Error) • _EIP_IPRTblErr (IP Route Table Error) **Note** If a *Link OFF Detected* or *EtherNet/IP Error* occurs, it is recorded in the event log and then the corresponding bit turns ON. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#00F0	page A-215

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 1. 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) • _EIP1_DhcpErr (Port1 DHCP 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 *EtherNet/IP Error* occurs, it is recorded in the event log and then the corresponding bit turns ON. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits. **Note** You can use this system-defined variable only for NX-series CPU Units.	WORD	16#0000 to 16#00F0	page A-216

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_PortErr	Communications Port2 Error	This is the error status variable for the communications port 2. Represents the collective status of the following error flags. • _EIP2_MacAdrErr (Port2 MAC Address Error) • _EIP2_LanHwErr (Port2 Communications Controller Error) • _EIP2_EtnCfgErr (Port2 Basic Ethernet Setting Error) • _EIP2_IPAdrCfgErr (Port2 IP Address Setting Error) • _EIP2_IPAdrDupErr (Port2 IP Address Duplication Error) • _EIP2_BootpErr (Port2 BOOTP Server Error) • _EIP2_DhcpErr (Port2 DHCP 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 *EtherNet/IP Error* occurs, it is recorded in the event log and then the corresponding bit turns ON. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	WORD	16#0000 to 16#00F0	page A-216

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_CipErr	CIP Communications Error	This is the error status variable for CIP communications. NX-series CPU Units: Represents the collective status of the following error flags. • _EIP1_IdentityErr (CIP Communications1 Identity Error) • _EIP1_TDLinkCfgErr (CIP Communications1 Tag Data Link Setting Error) • _EIP1_TDLinkOpnErr (CIP Communications1 Tag Data Link Connection Failed) • _EIP1_TDLinkErr (CIP Communications1 Tag Data Link Communications Error) • _EIP1_TagAdrErr (CIP Communications1 Tag Name Resolution Error) • _EIP1_MultiSwONErr (CIP Communications1 Multiple Switches ON Error) NJ-series CPU Units: Represents the collective status of the following error flags. • _EIP_IdentityErr (Identity Error) • _EIP_TDLinkCfgErr (Tag Data Link Setting Error) • _EIP_TDLinkOpnErr (Tag Data Link Connection Failed) • _EIP_TDLinkErr (Tag Data Link Communications Error) • _EIP_TagAdrErr (Tag Name Resolution Error) • _EIP_MultiSwOnErr (Multiple Switches ON Error) **Note** If a *Tag Name Resolution Error* occurs, it is recorded in the event log and this variable changes to TRUE. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#00F0	page A-217

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_CipErr	CIP Communications1 Error	This is the error status variable for CIP communications 1. Represents the collective status of the following error flags. • _EIP1_IdentityErr (CIP Communications1 Identity Error) • _EIP1_TDLinkCfgErr (CIP Communications1 Tag Data Link Setting Error) • _EIP1_TDLinkOpnErr (CIP Communications1 Tag Data Link Connection Failed) • _EIP1_TDLinkErr (CIP Communications1 Tag Data Link Communications Error) • _EIP1_TagAdrErr (CIP Communications1 Tag Name Resolution Error) • _EIP1_MultiSwONErr (CIP Communications1 Multiple Switches ON Error) **Note** If a *Tag Name Resolution Error* occurs, it is recorded in the event log and this variable changes to TRUE. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits. **Note** You can use this system-defined variable only for NX-series CPU Units.	WORD	16#0000 to 16#00F0	page A-217

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_CipErr	CIP Communications2 Error	This is the error status variable for CIP communications 2. Represents the collective status of the following error flags. • _EIP2_IdentityErr (CIP Communications2 Identity Error) • _EIP2_TDLinkCfgErr (CIP Communications2 Tag Data Link Setting Error) • _EIP2_TDLinkOpnErr (CIP Communications2 Tag Data Link Connection Failed) • _EIP2_TDLinkErr (CIP Communications2 Tag Data Link Communications Error) • _EIP2_TagAdrErr (CIP Communications2 Tag Name Resolution Error) • _EIP2_MultiSwONErr (CIP Communications2 Multiple Switches ON Error) **Note** If a *Tag Name Resolution Error* occurs, it is recorded in the event log and this variable changes to TRUE. Refer to *Meanings of Error Status Bits* for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	WORD	16#0000 to 16#00F0	page A-218
_EIP_TcpAppErr	TCP Application Communications Error	This is the error status variable for TCP application communications. Represents the collective status of the following error flags. • _EIP_TcpAppCfgErr (TCP Application Setting Error) • _EIP_NTPSrvErr (NTP Server Connection Error) **Note** Refer to *Meanings of Error Status Bits* for the meanings of the error status bits.	WORD	16#0000 to 16#00F0	page A-218
_EIP_MacAdrErr	MAC Address Error	NX-series CPU Units: Indicates that an error occurred when the MAC address was read on the communications port 1 at startup. TRUE: Error FALSE: Normal NJ-series CPU Units: Indicates that an error occurred when the MAC address was read at startup. TRUE: Error FALSE: Normal	BOOL	TRUE or FALSE	page A-218

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_MacAdrErr	Port1 MAC Address Error	Indicates that an error occurred when the MAC address was read on the communications port 1 at startup. TRUE: Error FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-219
_EIP2_MacAdrErr	Port2 MAC Address Error	Indicates that an error occurred when the MAC address was read on the communications port 2 at startup. TRUE: Error FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-219
_EIP_LanHwErr	Communications Controller Error	NX-series CPU Units: Indicates that a Communications Controller failure occurred on the communications port 1. TRUE: Failure FALSE: Normal NJ-series CPU Units: Indicates that a Communications Controller failure occurred. TRUE: Failure FALSE: Normal	BOOL	TRUE or FALSE	page A-219
_EIP1_LanHwErr	Port1 Communications Controller Error	Indicates that a Communications Controller failure occurred on the communications port 1. TRUE: Failure FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-219
_EIP2_LanHwErr	Port2 Communications Controller Error	Indicates that a Communications Controller failure occurred on the communications port 2. TRUE: Failure FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-220

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_EtnCfgErr	Basic Ethernet Setting Error	NX-series CPU Units: Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 1 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal NJ-series CPU Units: Indicates that the Ethernet communications speed setting (Speed/Duplex) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal	BOOL	TRUE or FALSE	page A-220
_EIP1_EtnCfgErr	Port1 Basic Ethernet Setting Error	Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 1 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-220
_EIP2_EtnCfgErr	Port2 Basic Ethernet Setting Error	Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 2 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-220
_EIP_IPAdrCfgErr	IP Address Setting Error	NX-series CPU Units: Indicates the IP address setting errors for the communications port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The IP address obtained from the DHCP server is inconsistent. FALSE: Normal NJ-series CPU Units: Indicates the IP address setting errors. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The default gateway settings are not correct. FALSE: Normal	BOOL	TRUE or FALSE	page A-221

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_IPAdrCfgErr	Port1 IP Address Setting Error	Indicates the IP address setting errors for the communications port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The IP address obtained from the DHCP server is inconsistent. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-221
_EIP2_IPAdrCfgErr	Port2 IP Address Setting Error	Indicates the IP address setting errors for the communications port 2. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The IP address obtained from the DHCP server is inconsistent. FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-221
_EIP_IPAdrDupErr	IP Address Duplication Error	NX-series CPU Units: Indicates that the same IP address is assigned to more than one node for the communications port 1. TRUE: Duplication occurred. FALSE: Other than the above. NJ-series CPU Units: Indicates that the same IP address is assigned to more than one node. TRUE: Duplication occurred. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-222
_EIP1_IPAdrDupErr	Port1 IP Address Duplication Error	Indicates that the same IP address is assigned to more than one node for the communications port 1. TRUE: Duplication occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-222
_EIP2_IPAdrDupErr	Port2 IP Address Duplication Error	Indicates that the same IP address is assigned to more than one node for the communications port 2. TRUE: Duplication occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-222

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_DNSCfgErr*1	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-222
_EIP_BootpErr	BOOTP Server Error	NX-series CPU Units: Indicates that a BOOTP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. NJ-series CPU Units: Indicates that a BOOTP server connection failure occurred. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server.	BOOL	TRUE or FALSE	page A-223
_EIP1_BootpErr	Port1 BOOTP Server Error	Indicates that a BOOTP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-223
_EIP2_BootpErr	Port2 BOOTP Server Error	Indicates that a BOOTP server connection failure occurred on the communications port 2. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-223

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_DhcpErr	DHCP Server Error	Indicates that a DHCP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the DHCP server (timeout). FALSE: The DHCP is not enabled, or DHCP is enabled and an IP address was normally obtained from the DHCP server. **Note** You can use this system-defined variable only for the NX502 CPU Units.	BOOL	TRUE or FALSE	page A-223
_EIP1_DhcpErr	Port1 DHCP Server Error	Indicates that a DHCP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the DHCP server (timeout). FALSE: The DHCP is not enabled, or DHCP is enabled and an IP address was normally obtained from the DHCP server. **Note** You can use this system-defined variable only for the NX502 CPU Units.	BOOL	TRUE or FALSE	page A-223
_EIP2_DhcpErr	Port2 DHCP Server Error	Indicates that a DHCP server connection failure occurred on the communications port 2. TRUE: There was a failure to connect to the DHCP server (timeout). FALSE: The DHCP is not enabled, or DHCP is enabled and an IP address was normally obtained from the DHCP server. **Note** You can use this system-defined variable only for the NX502 CPU Units.	BOOL	TRUE or FALSE	page A-223
_EIP_IPRTblErr	IP Route Table Error	NX-series CPU Units: Indicates that the default gateway settings or IP router table settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal NJ-series CPU Units: Indicates that the IP router table or hosts settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal	BOOL	TRUE or FALSE	page A-224
_EIP_IdentityErr	Identity Error	NX-series CPU Units: Indicates that the identity information for CIP communications 1 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal NJ-series CPU Units: Indicates that the identity information (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal	BOOL	TRUE or FALSE	page A-224

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_IdentityErr	CIP Communications1 Identity Error	Indicates that the identity information for CIP communications 1 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-225
_EIP2_IdentityErr	CIP Communications2 Identity Error	Indicates that the identity information for CIP communications 2 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-225
_EIP_TDLinkCfgErr	Tag Data Link Setting Error	NX-series CPU Units: Indicates that the tag data link settings for CIP communications 1 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal NJ-series CPU Units: Indicates that the tag data link settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal	BOOL	TRUE or FALSE	page A-225
_EIP1_TDLinkCfgErr	CIP Communications1 Tag Data Link Setting Error	Indicates that the tag data link settings for CIP communications 1 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-225
_EIP2_TDLinkCfgErr	CIP Communications2 Tag Data Link Setting Error	Indicates that the tag data link settings for CIP communications 2 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-226

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TDLinkOpnErr	Tag Data Link Connection Failed	NX-series CPU Units: Indicates that establishing a tag data link connection for CIP communications 1 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above. NJ-series CPU Units: Indicates that establishing a tag data link connection failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-226
_EIP1_TDLinkOpnErr	CIP Communications1 Tag Data Link Connection Failed	Indicates that establishing a tag data link connection for CIP communications 1 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-226

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_TDLinkOpnErr	CIP Communications2 Tag Data Link Connection Failed	Indicates that establishing a tag data link connection for CIP communications 2 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-227
_EIP_TDLinkErr	Tag Data Link Communications Error	NX-series CPU Units: Indicates that a timeout occurred in a tag data link connection for CIP communications 1. TRUE: A timeout occurred. FALSE: Other than the above. NJ-series CPU Units: Indicates that a timeout occurred in a tag data link connection. TRUE: A timeout occurred. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-227
_EIP1_TDLinkErr	CIP Communications1 Tag Data Link Communications Error	Indicates that a timeout occurred in a tag data link connection for CIP communications 1. TRUE: A timeout occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-227
_EIP2_TDLinkErr	CIP Communications2 Tag Data Link Communications Error	Indicates that a timeout occurred in a tag data link connection for CIP communications 2. TRUE: A timeout occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-227

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TagAdrErr	Tag Name Resolution Error	NX-series CPU Units: Indicates that the tag resolution for CIP communications 1 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. NJ-series CPU Units: Indicates that tag name resolution failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-228
_EIP1_TagAdrErr	CIP Communications1 Tag Name Resolution Error	Indicates that the tag resolution for CIP communications 1 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-228

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_TagAdrErr	CIP Communications2 Tag Name Resolution Error	Indicates that the tag resolution for CIP communications 2 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-229
_EIP_MultiSwONErr	Multiple Switches ON Error	NX-series CPU Units: Indicates that more than one switch turned ON at the same time in CIP communications 1. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. NJ-series CPU Units: Indicates that more than one switch turned ON at the same time. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-229
_EIP1_MultiSwONErr	CIP Communications1 Multiple Switches ON Error	Indicates that more than one switch turned ON at the same time in CIP communications 1. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-229

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_MultiSwO-NErr	CIP Communications2 Multiple Switches ON Error	Indicates that more than one switch turned ON at the same time in CIP communications 2. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-230
_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-230
_EIP_NTPSrvErr	NTP Server Connection Error	TRUE: The NTP client failed to connect to the server (timeout). FALSE: NTP is not set. Or, NTP is set and the connection was successful.	BOOL	TRUE or FALSE	page A-230
_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-230

*1. With the NJ-series CPU Unit, this variable can be used with the unit version 1.11 or later.

Hierarchical Relationship of System-defined Variables Related to EtherNet/IP Errors in the NJ-series CPU Unit

The system-defined variables that are related to EtherNet/IP errors have the following hierarchical relationship. For example, if the value of any of the _EIP_PortErr, _EIP_CipErr, or _EIP_TcpAppErr variables in the second level is TRUE, then the _EIP_ErrSta variable in the first level also changes to TRUE. Therefore, you can check the values of system-defined variables in a higher level to see if an error has occurred for a variable in a lower level.

Level 1		Level 2		Level 3	
Variable	Name	Variable	Name	Variable	Name
_EIP_ErrSta	EtherNet/IP Error	_EIP_PortErr	Communications Port Error	_EIP_MacAdrErr	MAC Address Error
				_EIP_LanHwErr	Communications Controller Error
				_EIP_EtnCfgErr	Basic Ethernet Setting Error
				_EIP_IPAdrCfgErr	IP Address Setting Error
				_EIP_IPAdrDupErr	IP Address Duplication Error
				_EIP_BootpErr	BOOTP Server Error
				_EIP_DNSSrvErr	DNS Server Connection Error
				_EIP_IPRTblErr	IP Route Table Error
		_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

Hierarchical Relationship of System-defined Variables Related to EtherNet/IP Errors in the NX-series CPU Unit

The system-defined variables that are related to EtherNet/IP errors have the following hierarchical relationship. For example, if the value of any of the _EIP1_PortErr, _EIP2_PortErr, EIP1_CipErr, _EIP2_CipErr, and _EIP_TcpAppErr variables in the second level is TRUE, then the _EIP_ErrSta variable in the first level also changes to TRUE. Therefore, you can check the values of system-defined variables in a higher level to see if an error has occurred for a variable in a lower level.

Level 1		Level 2		Level 3	
Variable	Name	Variable	Name	Variable	Name
_EIP_ErrSta	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
				_EIP1_DhcpErr	Port1 DHCP Server Error
				_EIP_DNSCfgErr	DNS Setting Error
				_EIP_DNSSrvErr	DNS Server Connection Error
				_EIP_IPRTblErr	IP Route Table Error
		_EIP2_PortErr	Communications Port2 Error	_EIP2_MacAdrErr	Port2 MAC Address Error
				_EIP2_LanHwErr	Port2 Communications Controller Error

Level 1		Level 2		Level 3	
Variable	Name	Variable	Name	Variable	Name
				_EIP2_EtnCfgErr	Port2 Basic Ethernet Setting Error
				_EIP2_IPAdrCfgErr	Port2 IP Address Setting Error
				_EIP2_IPAdrDupErr	Port2 IP Address Duplication Error
				_EIP2_BootpErr	Port2 BOOTP Server Error
				_EIP2_DhcpErr	Port2 DHCP Server Error
				_EIP_DNSCfgErr	DNS Setting Error
				_EIP_DNSSrvErr	DNS Server Connection Error
				_EIP_IPRTblErr	IP Route Table Error
		_EIP1_CipErr	CIP Communications1 Error	_EIP1_IdentityErr	CIP Communications1 Identity Error
				_EIP1_TDLinkCfgErr	CIP Communications1 Tag Data Link Setting Error
				_EIP1_TDLinkOpnErr	CIP Communications1 Tag Data Link Connection Failed
				_EIP1_TDLinkErr	CIP Communications1 Tag Data Link Communications Error
				_EIP1_TagAdrErr	CIP Communications1 Tag Name Resolution Error
				_EIP1_MultiSwONErr	CIP Communications1 Multiple Switches ON Error
		_EIP2_CipErr	CIP Communications2 Error	_EIP2_IdentityErr	CIP Communications2 Identity Error
				_EIP2_TDLinkCfgErr	CIP Communications2 Tag Data Link Setting Error
				_EIP2_TDLinkOpnErr	CIP Communications2 Tag Data Link Connection Failed
				_EIP2_TDLinkErr	CIP Communications2 Tag Data Link Communications Error
				_EIP2_TagAdrErr	CIP Communications2 Tag Name Resolution Error
				_EIP2_MultiSwONErr	CIP Communications2 Multiple Switches ON Error
		_EIP_TcpAppErr	TCP Application Communications Error	_EIP_TcpAppCfgErr _EIP_NTPSrvErr	TCP Application Setting Error NTP Server Connection Error

Note 1. 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 (Communications Port Error).

Note 2. You can use the system-defined variables whose variable names with *_EIP2* only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.

● Functional Classification: EtherNet/IP Communications Status

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_EtnOnlineSta	Online	NX-series CPU Units: Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 1 (that is, the link is ON, IP address is defined, and there are no errors.) TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. NJ-series CPU Units: Indicates that the built-in EtherNet/IP port's communications can be used via the communications port (that is, the link is ON and 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-231
_EIP1_EtnOnlineSta	Port1 Online	Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 1 (that is, the link is ON, IP address is defined, and there are no errors.) TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-231
_EIP2_EtnOnlineSta	Port2 Online	Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 2 (that is, the link is ON, IP address is defined, and there are no errors.) TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-231

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TDLinkRunSta	Tag Data Link Communications Status	NX-series CPU Units: Indicates that at least one connection is in normal operation in CIP communications 1. TRUE: Normal operation FALSE: Other than the above. NJ-series CPU Units: Indicates that at least one connection is in normal operation. TRUE: Normal operation FALSE: Other than the above.	BOOL	TRUE or FALSE	page A-232
_EIP1_TDLinkRunSta	CIP Communications1 Tag Data Link Communications Status	Indicates that at least one connection is in normal operation in CIP communications 1. TRUE: Normal operation FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-232
_EIP2_TDLinkRunSta	CIP Communications2 Tag Data Link Communications Status	Indicates that at least one connection is in normal operation in CIP communications 2. TRUE: Normal operation FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-232
_EIP_TDLinkAllRunSta	All Tag Data Link Communications Status	NX-series CPU Units: Indicates that all tag data links are communicating in CIP communications 1. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. NJ-series CPU Units: Indicates that all tag data links are communicating. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection.	BOOL	TRUE or FALSE	page A-232
_EIP1_TDLinkAllRunSta	CIP Communications1 All Tag Data Link Communications Status	Indicates that all tag data links are communicating in CIP communications 1. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-233

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_TDLinkAll-RunSta	CIP Communications2 All Tag Data Link Communications Status	Indicates that all tag data links are communicating in CIP communications 2. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-233
_EIP_RegTarget-Sta[255]	Registered Target Node Information	NX-series CPU Units: Gives a list of nodes for which EtherNet/IP connections are registered for CIP communications 1. This variable is valid only when the EtherNet/IP port is the originator. Array[x] is TRUE: The connection to the node with a target node ID of x is registered. Array[x] is FALSE: The connection to the node with a target node ID of x is not registered. NJ-series CPU Units: Gives a list of nodes for which EtherNet/IP connections are registered. This variable is valid only when the 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-233
_EIP1_RegTarget-Sta[255]	CIP Communications1 Registered Target Node Information	Gives a list of nodes for which EtherNet/IP connections are registered for CIP communications 1. This variable is valid only when the EtherNet/IP port is the originator. Array[x] is TRUE: The connection to the node with a target node ID of x is registered. Array[x] is FALSE: The connection to the node with a target node ID of x is not registered. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-233

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_RegTarget-Sta[255]	CIP Communications2 Registered Target Node Information	Gives a list of nodes for which EtherNet/IP connections are registered for CIP communications 2. This variable is valid only when the EtherNet/IP port is the originator. Array[x] is TRUE: The connection to the node with a target node ID of x is registered. Array[x] is FALSE: The connection to the node with a target node ID of x is not registered. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-234
_EIP_EstbTarget-Sta[255]	Normal Target Node Information	NX-series CPU Units: Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 1. Array[x] is TRUE: The connection to the node with a target node ID of x was established normally. Array[x] is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. NJ-series CPU Units: Gives a list of nodes that have normally established EtherNet/IP connections. Array[x] is TRUE: The connection to the node with a target node ID of x was established normally. Array[x] is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-234
_EIP1_EstbTarget-Sta[255]	CIP Communications1 Normal Target Node Information	Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 1. Array[x] is TRUE: The connection to the node with a target node ID of x was established normally. Array[x] is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-234

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_EstbTarget-Sta[255]	CIP Communica-tions2 Normal Target Node Information	Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 2. Array[x] is TRUE: The connection to the node with a target node ID of x was established normally. Array[x] is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. **Note** You can use this system-de-fined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-235
_EIP_TargetPLCMo-deSta[255]	Target PLC Operat-ing Mode	NX-series CPU Units: Shows the operat-ing status of the target node Controllers that are connected for CIP communica-tions 1, with the 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, it indicates the previous operat-ing status. Array[x] is TRUE: This is the operating state of the target Controller with a node address of x. Array[x] is FALSE: Other than the above. NJ-series CPU Units: Shows the operat-ing status of the target node Controllers that are connected with the 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, it indicates the previous operat-ing 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-235

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_TargetPLC-ModeSta[255]	CIP Communications1 Target PLC Operating Mode	Shows the operating status of the target node Controllers that are connected for CIP communications 1, with the 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, it indicates the previous operating status. Array[x] is TRUE: This is the operating state of the target Controller with a node address of x. Array[x] is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-235
_EIP2_TargetPLC-ModeSta[255]	CIP Communications2 Target PLC Operating Mode	Shows the operating status of the target node Controllers that are connected for CIP communications 2, with the 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, it indicates the previous operating status. Array[x] is TRUE: This is the operating state of the target Controller with a node address of x. Array[x] is FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-236

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TargetPL-CErr[255]	Target PLC Error Information	NX-series CPU Units: Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 1, with the EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. Array[x] is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. Array[x] is FALSE: Other than the above. NJ-series CPU Units: Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected with the 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-236
_EIP1_TargetPL-CErr[255]	CIP Communications1 Target PLC Error Information	Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 1, with the EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. Array[x] is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. Array[x] is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-236

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_TargetPLCErr[255]	CIP Communications2 Target PLC Error Information	Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 2, with the EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. Array[x] is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. Array[x] is FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-237

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TargetNodeErr[255]	Target Node Error Information	NX-series CPU Units: Indicates that the connection for the Registered Target Node Information for CIP communications 1 was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE). NJ-series CPU Units: Indicates that the connection for the Registered Target Node Information was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE).	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-237

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_TargetNodeErr[255]	CIP Communications1 Target Node Error Information	Indicates that the connection for the Registered Target Node Information for CIP communications 1 was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE). **Note** You can use this system-defined variable only for NX-series CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-238

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP2_TargetNodeErr[255]	CIP Communications2 Target Node Error Information	Indicates that the connection for the Registered Target Node Information for CIP communications 2 was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE). **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	ARRAY [0..255] OF BOOL	TRUE or FALSE	page A-238
_EIP_NTPResult	NTP Operation Information	Use the GetNTPStatus instruction to read the NTP operation information from the user program. Direct access is not possible.	_sNTP_RESULT		page A-238

Variable name	Meaning	Function	Data type	Range of values	Reference
.ExecTime	NTP Last Operation Time	Gives the last time that NTP processing ended normally. The time that was obtained from the NTP server is stored when the time is obtained normally. The time is not stored if it is not obtained from the NTP server normally. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device.	DATE_AND_TIME	Depends on data type.	page A-239
.ExecNormal	NTP Operation Result	TRUE: Indicates an NTP normal end. FALSE: Indicates that NTP operation ended in an error or has not been executed even once. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device.	BOOL	TRUE or FALSE	page A-239

Precautions for Correct Use

Communications Status with Target Node

The communications status with the target node of an NJ/NX-series Controller is shown by the combination of the values of four system-defined variables.

- _EIP_RegTargetSta (Registered Target Node Information)
- _EIP_EstbTargetSta (Normal Target Node Information)
- _EIP_TargetPLCErr (Target PLC Error Information)
- _EIP_TargetNodeErr (Target Node Error Information)

Value of *_EIP_RegTargetSta*	Value of *_EIP_EstbTargetSta*	Value of *_EIP_TargetPLCErr*	Value of *_EIP_TargetNodeErr*	Communications status with target node
TRUE	TRUE	FALSE	FALSE	A connection with the target node was established normally and there is no error in the target PLC.
		TRUE	TRUE	A connection with the target node was established but there is an error in the target PLC.
	FALSE	---	TRUE	A connection with the target node was not established normally.
FALSE	---	---	---	The information is not valid because the target node is not registered.

For the NX-series Controller, the communications status of CIP communications 1 and CIP communications 2 is shown by the combination of the values of four system-defined variables in the same way as shown in the above table.

- CIP Communications 1
  - _EIP1_RegTargetSta (CIP Communications1 Registered Target Node Information)
  - _EIP1_EstbTargetSta (CIP Communications1 Normal Target Node Information)
  - _EIP1_TargetPLCErr (CIP Communications1 Target PLC Error Information)
  - _EIP1_TargetNodeErr (CIP Communications1 Target Node Error Information)
- CIP Communications 2
  - _EIP2_RegTargetSta (CIP Communications2 Registered Target Node Information)
  - _EIP2_EstbTargetSta (CIP Communications2 Normal Target Node Information)
  - _EIP2_TargetPLCErr (CIP Communications2 Target PLC Error Information)
  - _EIP2_TargetNodeErr (CIP Communications2 Target Node Error Information)

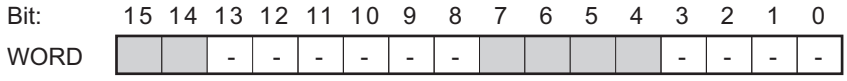
● Functional Classification: EtherNet/IP Communications Switches

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP_TDLink-StartCmd	Tag Data Link Communications Start Switch	NX-series CPU Units: Change this variable to TRUE to start tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation starts. NJ-series CPU Units: Change this variable to TRUE to start tag data links. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically.	BOOL	TRUE or FALSE	page A-239
_EIP1_TDLink-StartCmd	CIP Communications1 Tag Data Link Communications Start Switch	Change this variable to TRUE to start tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-239
_EIP2_TDLink-StartCmd	CIP Communications2 Tag Data Link Communications Start Switch	Change this variable to TRUE to start tag data links for CIP communications 2. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-240
_EIP_TDLink-StopCmd	Tag Data Link Communications Stop Switch	NX-series CPU Units: Change this variable to TRUE to stop tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation stops. NJ-series CPU Units: Change this variable to TRUE to stop tag data links. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically.	BOOL	TRUE or FALSE	page A-240

Variable name	Meaning	Function	Data type	Range of values	Reference
_EIP1_TDLink-StopCmd	CIP Communications1 Tag Data Link Communications Stop Switch	Change this variable to TRUE to stop tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.	BOOL	TRUE or FALSE	page A-240
_EIP2_TDLink-StopCmd	CIP Communications2 Tag Data Link Communications Stop Switch	Change this variable to TRUE to stop tag data links for CIP communications 2. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.	BOOL	TRUE or FALSE	page A-240

A-7-10 Meanings of Error Status Bits

The meanings of the individual bits in the error status are shown in the following table.



Bit	Description
15	Master-detected error: This bit indicates whether the master detected a Controller error in the Unit/slave for the error status of the Controller error. TRUE: The master detected a Controller error. FALSE: The master has not detected a Controller error.
14	Collective slave error status: This bit indicates if a Controller error is 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.
13 to 8	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.

Bit	Description
4	This bit indicates whether an observation level Controller error has occurred. TRUE: An observation level Controller error has occurred. TRUE: An observation level Controller error has not occurred.
3 to 0	Reserved.

A list of variables for error status is given below. The following table shows whether bit14 and bit15 of each variable are valid or invalid and whether they can be used in the user program.

Variable name	Valid or invalid for bit 15	Valid or invalid for bit 14	Usage in user program
_ErrSta (Controller Error Status)	Valid	Valid	Not possible*1
_PLC_ErrSta (PLC Function Module Error Status)	Invalid	Invalid	Possible
_CJB_ErrSta (I/O Bus Error Status)	Valid	Valid	Not possible*2
_CJB_MstrErrSta (I/O Bus Master Error Status)	Invalid	Invalid	
_CJB_UnitErrSta (I/O Bus Unit Error Status)	Valid	Invalid	
_NXB_ErrSta (NX Bus Function Module Error Status)	Invalid	Valid	Not recommended*3
_NXB_MstrErrSta (NX Bus Function Module Master Error Status)	Invalid	Valid	
_NXB_UnitErrStaTbl (NX Bus Function Module Unit Error Status)	Invalid	Valid	
_MC_ErrSta (MC Error Status)	Invalid	Valid	Possible
_MC_ComErrSta (MC Common Error Status)	Invalid	Invalid	
_MC_AX_ErrSta (Axis Error Status)	Invalid	Invalid	
_MC_GRP_ErrSta (Axes Group Error Status)	Invalid	Invalid	
_EC_ErrSta (EtherCAT Error)	Invalid	Valid	Possible
_EC_PortErr (Communications Port Error)	Invalid	Invalid	
_EC_MstrErr (Master Error)	Invalid	Invalid	
_EC_SlavErr (Slave Error)	Invalid	Invalid	
_EC_SlavErrTbl (Slave Error Table)	Invalid	Invalid	
_EIP_ErrSta (EtherNet/IP Error)	Invalid	Invalid	Possible
_EIP_PortErr (Communications Port Error), *_EIP1_PortErr* (Communications Port1 Error), *_EIP2_PortErr* (Communications Port2 Error)	Invalid	Invalid	
_EIP_CipErr (CIP Communications Error), *_EIP1_CipErr* (CIP Communications1 Error), *_EIP2_CipErr* (CIP Communications2 Error)	Invalid	Invalid	
_EIP_TcpAppErr (TCP Application Communications Error)	Invalid	Invalid	
_XBU_ErrSta (X Bus Function Module Error Status)	Invalid	Valid	Not recommended*3
_XBU_MstrErr (X Bus Function Module Master Error Status)	Invalid	Invalid	
_XBU_UnitErr (X Bus Function Module Unit Error Status)	Invalid	Valid	
_XBU_UnitErrTbl (X Bus Function Module Unit Error Status Table)	Invalid	Invalid	

*1. 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.

*2. Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI.

*3. We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI.

A-8 Specifications for Individual System-defined Variables

The specifications for each system-defined variable are given as described below.

Variable name	This is the system-defined variable name. The prefix gives the category name.			**Members (for structures)**	The member names are given for structure variables only.
Meaning	This is the meaning of the variable.			**Global/local**	Global: Global variable, Local: Local variable
Function	The function of the variable is described.				
Data type	The data type of the variable is given.			**Range of values**	The range of values that the variable can take is given.
R/W access	R: Read only, RW: Read/write	**Retained**	The Retain attribute of the variable is given.	**Network Publish**	The Network Publish attribute of the variable is given.
Usage in user program	Whether you can use the variable directly in the user program is specified.	**Related instructions**	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.		

A-8-1 System-defined Variables for the Overall NJ/NX-series Controller (No Category)

● Functional Classification: Clock

Variable name	_CurrentTime				
Meaning	System Time			**Global/local**	Global
Function	Contains the CPU Unit's internal clock data.				
Data type	DATE_AND_TIME			**Range of values**	• NX-series CPU Units DT#1970-01-01-00:00:00 to DT#2069-12-31-23:59:59 • NJ-series CPU Units DT#1970-01-01-00:00:00 to DT#2106-02-06-23:59:59
R/W access	R	**Retained**	Not retained.	**Network Publish**	Published.
Usage in user program	Possible.	**Related instructions**	Clock instructions		

● Functional Classification: Tasks

Variable name	_TaskName_Active				
Meaning	Task Active Flag			Global/local	Global
Function	TRUE during task execution. FALSE when task execution is not in progress. **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instructions	• ActEventTask You can access this variable from the user program only with the following instruction. • Task_IsActive		

Variable name	_TaskName_LastExecTime				
Meaning	Last Task Execution Time			Global/local	Global
Function	Contains the task execution time the last time the task was executed (unit: 0.1 μs). **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.				
Data type	TIME			Range of values	Depends on data type.
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instructions	You can access this variable from the user program only with the following instruction. • GetMyTaskStatus		

Variable name	_TaskName_MaxExecTime				
Meaning	Maximum Task Execution Time			Global/local	Global
Function	Contains the maximum value of the task execution time (unit: 0.1 μs). **Note** You cannot use this system-defined variable in the user program. It is used only to access task status for data tracing from the Sysmac Studio.				
Data type	TIME			Range of values	Depends on data type.
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	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 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		

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 error status of the function module. Use this variable only to access status through communications from an external device. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#C0F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	• ResetPLCError • ResetCJBError • ResetECError • ResetMCError • MC_Reset • MC_GroupReset You can access this variable from the user program only with the following instruction. • GetPLCError • GetCJBError • GetECError • GetMCError • GetEIPError		

Variable name	_AlarmFlag				
Meaning	User-defined Error Status			Global/local	Global
Function	The bit corresponding to the event level is TRUE while there is a user-defined error. Bits 00 to 07 correspond to user fault levels 1 to 8. This variable contains 0000 hex when there is no user-defined error.				
Data type	WORD			Range of values	16#0000 to 16#00FF
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• SetAlarm • ResetAlarm • GetAlarm		

● Functional Classification: SD Memory Card

Variable name	_Card1Ready				
Meaning	SD Memory Card Ready Flag			Global/local	Global
Function	TRUE when the SD Memory Card is recognized. FALSE when the SD Memory Card is not recognized. TRUE: The Card can be used. FALSE: The Card cannot be used.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1Protect				
Meaning	SD Memory Card Write Protected Flag			Global/local	Global
Function	TRUE when the SD Memory Card is write-protected with the LOCK switch. TRUE: Write protected. FALSE: Not write protected.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1Err				
Meaning	SD Memory Card Error Flag			Global/local	Global
Function	TRUE when an unusable SD Memory Card is inserted or a format error occurs. TRUE: There is an error FALSE: There is no error				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1Access				
Meaning	SD Memory Card Access Flag			Global/local	Global
Function	TRUE during SD Memory Card access. TRUE: Card is being accessed. FALSE: Card is not being accessed. The system updates the flag every 100 ms. Because of this, access to the SD Memory Card is shown by this flag with a delay of up to 100 ms. We therefore do not recommend the use of this variable in the user program.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1Deteriorated				
Meaning	SD Memory Card Life Warning Flag			Global/local	Global
Function	TRUE when the life of the SD Memory Card is exceeded. If this variable changed to TRUE, replace the SD Memory Card. Read/write operation may fail if the SD Memory Card is not replaced. TRUE: The life of the Card has been exceeded. FALSE: The Card can still be used.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_Card1PowerFail				
Meaning	SD Memory Card Power Interruption Flag			Global/local	Global
Function	TRUE when the power supply to the CPU Unit was interrupted during access to the SD Memory Card. TRUE: Power was interrupted during SD Memory Card access. FALSE: Normal				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*1	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1Capacity*1				
Meaning	SD Memory Card Storage Capacity			Global/local	Global
Function	Show the total capacity of the inserted SD Memory Card. The unit of capacity is MiB (1 MiB=1,048,576 bytes). The value is updated every 60 seconds. If the SD PWR indicator is not lit, such as when an SD Memory Card is not inserted, the value is “0””0".				
Data type	UDINT			Range of values	Depends on data type.
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 1.21 or 1.32 of the CPU Unit.

Variable name	_Card1Used*1				
Meaning	SD Memory Card Storage Usage			Global/local	Global
Function	Show the usage of the inserted SD Memory Card. The unit of capacity is MiB (1 MiB=1,048,576 bytes). The value is updated every 60 seconds. If the SD PWR indicator is not lit, such as when an SD Memory Card is not inserted, the value is “0””0".				
Data type	UDINT			Range of values	Depends on data type.
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 1.21 or 1.32 of the CPU Unit.

Variable name	_Card1BkupCmd*1			Member name	.ExecBkup
Meaning	Execute Backup Flag			Global/local	Global
Function	Change this variable to TRUE to back up Controller data to an SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupCmd*1			Member name	.CancelBkup
Meaning	Cancel Backup Flag			Global/local	Global
Function	Change this variable to TRUE to cancel backing up data to an SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupCmd*1			Member name	.ExecVefy
Meaning	Execute Verify Flag			Global/local	Global
Function	Change this variable to TRUE to compare the Controller data to a backup file in the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupCmd*1			Member name	.CancelVefy
Meaning	Cancel Verify Flag			Global/local	Global
Function	Change this variable to TRUE to cancel comparing the Controller data to a backup file in the SD Memory Card. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupCmd*1			Member name	.DirName
Meaning	Directory Name			Global/local	Global
Function	Use this variable to specify the directory name in the SD Memory Card for which to back up data. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications when sending a command from an HMI or host computer.				
Data type	Structure: _sBKUP_CMD, Member: STRING[64]			Range of values	Depends on data type.
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupSta*1			Member name	.Done
Meaning	Done Flag			Global/local	Global
Function	TRUE when a backup is completed. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sBKUP_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupSta*1			Member name	.Active
Meaning	Active Flag			Global/local	Global
Function	TRUE when a backup is in progress. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sBKUP_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1BkupSta*1			Member name	.Err
Meaning	Error Flag			Global/local	Global
Function	TRUE when processing a backup ended in an error. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sBKUP_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1VefySta*1			Member name	.Done
Meaning	Done Flag			Global/local	Global
Function	TRUE when a verification is completed. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sVEFY_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1VefySta*1			Member name	.Active
Meaning	Active Flag			Global/local	Global
Function	TRUE when a verification is in progress. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sVEFY_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1VefySta*1			Member name	.VefyRslt
Meaning	Verify Result Flag			Global/local	Global
Function	TRUE if the data was the same. FALSE if differences were found. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sVEFY_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1VefySta*1			Member name	.Err
Meaning	Error Flag			Global/local	Global
Function	TRUE when processing a verification ended in an error. **Note** You cannot use this system-defined variable in the user program. Use it in CIP message communications to read the status from an HMI or host computer.				
Data type	Structure: _sVEFY_STA, Member: BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

Variable name	_Card1PrgTransferCmd*1			Member name	.Exec
Meaning	Execute Program Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to transfer the data in a backup file on the SD Memory Card to the Controller by using the function to transfer programs from the SD Memory Card. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.

*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1PrgTransferCmd*1			Member name	.DirName
Meaning	Directory Name			Global/local	Global
Function	Use this variable to specify the directory name in the SD Memory Card in which the backup file to be transferred is stored. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_CMD, Member: STRING[64]			Range of values	Depends on data type.
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1PrgTransferCmd*1			Member name	.Password
Meaning	Password			Global/local	Global
Function	Use this variable to specify the password that is used for verification when you start transferring the programs. The password is initialized every time you start transferring programs from the SD Memory Card. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_CMD, Member: STRING[33]			Range of values	Depends on data type.
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.

Variable name	_Card1PrgTransferCmd*1			Member name	.TargetUserProgram
Meaning	User Program and Settings Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set a user program or setting as the transfer target. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1PrgTransferCmd*1			Member name	.TargetIPAdr
Meaning	IP Address Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to include the IP address of the built-in EtherNet/IP port as the transfer target. The IP address means setting type, IP address, subnet mask, and default gateway. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1PrgTransferCmd*1			Member name	.TargetVariable
Meaning	Present Values of Variables with the Retain Attribute Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set the present values of variables with the Retain attribute as the transfer target. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.

*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1PrgTransferCmd*1			Member name	.TargetMemory
Meaning	Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set the present values of the memory used for CJ-series Units with the Retain attribute as the transfer target. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.

*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1PrgTransferSta*1			Member name	.Done
Meaning	Done Flag			Global/local	Global
Function	TRUE when a program transfer is completed. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_STA, Member: 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 1.11 of the CPU Unit.

Variable name	_Card1PrgTransferSta*1			Member name	.Active
Meaning	Active Flag			Global/local	Global
Function	TRUE when a program transfer is in progress. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_STA, Member: 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 1.11 of the CPU Unit.

Variable name	_Card1PrgTransferSta*1			Member name	.Err
Meaning	Error Flag			Global/local	Global
Function	TRUE when a program transfer ended in an error. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sPRGTRANSFER_STA, Member: 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 1.11 of the CPU Unit.

Variable name	_Card1RestoreCmd*1			Member name	.Exec
Meaning	Execute Restore Flag			Global/local	Global
Function	Change this variable to TRUE to restore the data in a backup file on the SD Memory Card to the Controller. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sRESTORE_CMD, Member: BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.14 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmd*1			Member name	.DirName
Meaning	Directory Name			Global/local	Global
Function	Use this variable to specify the directory name in the SD Memory Card in which the backup file to be restored by the system-defined variable is stored. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sRESTORE_CMD, Member: STRING[64]			Range of values	Depends on data type.
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.14 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmd*1			Member name	.Password
Meaning	Password			Global/local	Global
Function	Use this variable to specify the password that is used for verification when you start the restore by the system-defined variable. The password is initialized every time when the restore by the system-defined variable is started. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sRESTORE_CMD, Member: STRING[33]			Range of values	Depends on data type.
R/W access	RW	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 CPU Unit.

Variable name	_Card1RestoreSta *1			Member name	.Done
Meaning	Done Flag			Global/local	Global
Function	TRUE when a restore operation is completed. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sRESTORE_STA, Member: 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 CPU Unit.

Variable name	_Card1RestoreSta*1			Member name	.Active
Meaning	Active Flag			Global/local	Global
Function	TRUE when a restore operation is in progress. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sRESTORE_STA, Member: 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 CPU Unit.

Variable name	_Card1RestoreSta*1			Member name	.Err
Meaning	Error Flag			Global/local	Global
Function	TRUE when a restore operation ended in an error. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	Structure: _sRESTORE_STA, Member: 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 CPU Unit.

Variable name	_Card1RestoreCmdTargetUserProgram*1				
Meaning	User Program and Settings Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set a user program or setting for the restore by the system-defined variable as the transfer target. Always set this variable to TRUE for the restore by the system-defined variable. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.14 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmdTargetIPAdr*1				
Meaning	IP Address Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to include the IP address of the built-in EtherNet/IP port for the restore by the system-defined variable as the transfer target. The IP address means setting type, IP address, subnet mask, and default gateway. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.14 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmdTargetVariable*1				
Meaning	Present Values of Variables with the Retain Attribute Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set the present values of variables with the Retain attribute for the restore by the system-defined variable as the transfer target. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.14 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmdTargetMemory*1				
Meaning	Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set the present values of the memory used for CJ-series Units with the Retain attribute for the restore by the system-defined variable as the transfer target. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.14 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmdTargetUnitConfig*1				
Meaning	Unit and Slave Parameters Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set the Unit and slave settings for the restore by the system-defined variable as the transfer target. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.14 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmdTargetAbsEncoder*1				
Meaning	Absolute Encoder Home Offset Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set the absolute encoder home offset for the restore by the system-defined variable as the transfer target. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained.*2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.14 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmdTargetOpcuaSPF*1				
Meaning	OPCUA Security Profile Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set the OPC UA security profile for the restore by the system-defined variable as the transfer target. **Note** You can use this system-defined variable only for NJ/NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained. *2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.17 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmdTargetXBusUnitConfig*1				
Meaning	X Bus Unit Settings Transfer Flag			Global/local	Global
Function	Change this variable to TRUE to set the X Bus Unit Settings for the restore by the system-defined variable as the transfer target. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained. *2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.60 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_Card1RestoreCmdTargetXBusUnitIPAdr*1				
Meaning	IP Address Settings Transfer Flag of EtherNet/IP Port on X Bus Unit			Global/local	Global
Function	Change this variable to TRUE to include the IP address of the EtherNet/IP port on the X Bus Unit for the restore by the system-defined variable as the transfer target. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Retained. *2	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.60 of the CPU Unit.
*2. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

● Functional Classification: Backup

Variable name	_BackupBusy*1				
Meaning	Backup Function Busy Flag			Global/local	Global
Function	TRUE when a backup, restoration, or verification is in progress.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.03 of the CPU Unit.

● Functional Classification: Power Supply

Variable name	_PowerOnHour				
Meaning	Total Power ON Time			Global/local	Global
Function	Contains the total time that the power has been ON. Contains the total time that the CPU Unit has been ON in 1-hour increments. To reset this value, overwrite the current value with 0. The value is not updated after it reaches 4294967295. This variable is not initialized at startup.				
Data type	UDINT			Range of values	0 to 4294967295
R/W access	RW	Retained	Retained.*1	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable is not applicable for the data backup function even with a Retain attribute.

Variable name	_PowerOnCount				
Meaning	Power Interruption Count			Global/local	Global
Function	Contains the number of times that the power supply has been interrupted. The value is incremented by 1 each time the power supply is interrupted after the first time that the power to the CPU Unit 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	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	_RetainFail				
Meaning	Retention Failure Flag			Global/local	Global
Function	TRUE at the following time (failure of retention during power interruptions). • When an error is detected in the battery-backup memory check at startup. FALSE at the following times (no failure of retention during power interruptions). • When no error is detected in the battery-backup memory check at startup. • When the user program is downloaded. • When the Clear All Memory operation is performed. **Note** When the absolute encoder home offset data is not retained, the status is given in the error status of the axis variable, and not in this flag.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: Programming

Variable name	P_On				
Meaning	Always TRUE Flag			Global/local	Global
Function	This flag is always TRUE.				
Data type	BOOL			Range of values	TRUE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

Variable name	P_Off				
Meaning	Always FALSE Flag			Global/local	Global
Function	This flag is always FALSE.				
Data type	BOOL			Range of values	FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

Variable name	P_CY				
Meaning	Carry Flag			Global/local	Local
Function	This flag is updated by some instructions.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

Variable name	P_First_RunMode				
Meaning	First RUN Period Flag			Global/local	Local
Function	This flag is TRUE for only one task period after the operating mode of the CPU Unit is changed from PROGRAM mode to RUN mode if execution of the program is in progress. This flag remains FALSE if execution of the program is not in progress. Use this flag to perform initialization when the CPU Unit begins operation. **Note** You cannot use this system-defined variable inside functions.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

Variable name	P_First_Run*1				
Meaning	First Program Period Flag			Global/local	Local
Function	This flag is TRUE for one task period after execution of the program starts. Use this flag to perform initial processing when execution of a program starts. **Note** You cannot use this system-defined variable inside functions.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.08 of the CPU Unit.

Variable name	P_PRGER				
Meaning	Instruction Error Flag			Global/local	Local
Function	This flag changes to and remains TRUE when an instruction error occurs in the program or in a function/function block called from the program. After this flag changes to TRUE, it stays TRUE until the user program changes it back to FALSE.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: Communications

Variable name	_Port_numUsingPort				
Meaning	Number of Used Ports			Global/local	Global
Function	Gives the number of internal logical ports that are currently used. You can use this variable when you debug the user program. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	USINT			Range of values	0 to 32
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	Communications instructions (ExecPMCR, SerialSend, SerialRcv, Send, Rcv, and SendCmd)		

Variable name	_Port_isAvailable				
Meaning	Network Communications Instruction Enabled Flag			Global/local	Global
Function	Indicates whether there is an available internal logical port. TRUE when an internal logical port is available. Otherwise FALSE. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Possible.	Related instructions	Communications instructions (ExecPMCR, SerialSend, SerialRcv, Send, Rcv, and SendCmd)		

Variable name	_FINSTCPConnSta				
Meaning	FINS/TCP Connection Status			Global/local	Global
Function	Gives the FINS/TCP connection status. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#FFFF
R/W access	R	Retained	Not retained.	Network Publish	Published.*1
Usage in user program	Possible.	Related instructions	---		

*1. The network for CPU Units with unit version 1.07 or earlier is not published.

● Functional Classification: Version

Variable name	_UnitVersion*1				
Meaning	Unit Version			Global/local	Global
Function	Contains the unit version of the CPU Unit. The integer part of the unit version is stored in element number 0. The fractional part of the unit version is stored in element number 1. Example 1) If the unit version is 1.08, “1” is stored in element number 0 and “8” is stored in element number 1. Example 2) If the unit version is 1.10, “1” is stored in element number 0 and “10” is stored in element number 1.				
Data type	ARRAY[0..1] OF USINT			Range of values	0 to 99
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.08 of the CPU Unit.

Variable name	_HardwareRevision*1				
Meaning	Hardware Revision			Global/local	Global
Function	Contains the hardware revision of the CPU Unit. 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	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.

Variable name	_ProjectUnitVersion*1				
Meaning	Project Unit Version			Global/local	Global
Function	Contains the project unit version of the project in the CPU Unit. The integer part of the project unit version is stored in element number 0. The fractional part of the project unit version is stored in element number 1. Example 1) If the project unit version is 1.09, “1” is stored in element number 0 and “9” is stored in element number 1. Example 2) If the project unit version is 1.40, “1” is stored in element number 0 and “40” is stored in element number 1.				
Data type	ARRAY[0..1] OF USINT			Range of values	0 to 99
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.40 of the CPU Unit.

● Functional Classification: Self-diagnosis

Variable name	_SelfTest_HighTemperature*1				
Meaning	CPU Unit High Temperature Flag			Global/local	Global
Function	TRUE when the internal temperature of the CPU Unit is too high. **Note** Always FALSE for the NX102 CPU Unit and NX1P2 CPU Unit.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.10 of the CPU Unit.

Variable name	_SelfTest_LowBattery*1				
Meaning	Low Battery Flag			Global/local	Global
Function	TRUE when the battery is disconnected or the battery voltage is dropped.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.10 of the CPU Unit.

Variable name	_SelfTest_LowFanRevolution*1				
Meaning	Low FAN Revolution Flag			Global/local	Global
Function	TRUE when the fan is disconnected or the rotation speed of a fan is decreased. **Note** Always FALSE for the NX102 CPU Unit, NX1P2 CPU Unit, and NJ-series CPU Unit.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.10 of the CPU Unit.

Variable name	_StorageDeteriorated*1				
Meaning	Non-volatile Memory Life Warning Flag			Global/local	Global
Function	TRUE when the number of deletions for non-volatile memory or the number of bad blocks in memory exceeds the warning value.				
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 the following CPU Units.
- NJ-series, NX502, NX102, NX1P2 CPU Unit: Version 1.60
- NX701 CPU Unit: Version 1.32

● 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 1.13 of the CPU Unit.

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 1.13 of the CPU Unit.

A-8-2 PLC Function Module, Category Name: _PLC

● Functional Classification: Debugging

Variable name	_PLC_TraceSta[0..3]			Member name	.IsStart
Meaning	Trace Busy Flag			Global/local	Global
Function	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.				
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]			Member name	.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]			Member name	.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 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	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]			Member name	.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 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	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-7-10 Meanings of Error Status Bits* on page A-161 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		

● Functional Classification: Option Boards

Variable name	_PLC_OptBoardSta*1				
Meaning	Option Board Status			Global/local	Global
Function	Contains the status of Option Boards. This variable is commonly used regardless of the models of Option Boards. The array element 1 corresponds to the option board slot 1 and array element 2 corresponds to the option board slot 2. **Note** You can use this system-defined variable only for NX1P2 CPU Units.				
Data type	ARRAY[1..2] OF _sOPTBOARD_STA			Range of values	---
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 1.13 of the CPU Unit.

Variable name	_PLC_OptBoardSta*1			Member name	IsDetect
Meaning	Option Board Mounted			Global/local	Global
Function	Indicates an Option Board is mounted to the option board slot. TRUE: Mounted. FALSE: Not mounted. **Note** You can use this system-defined variable only for NX1P2 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_PLC_OptBoardSta*1			Member name	Run
Meaning	Option Board Normal Operation			Global/local	Global
Function	Indicates whether an Option Board is normally operating. To use device variables or communications instructions for an Option Board, program this member as an interlock condition in the user program. TRUE: Normally operating. FALSE: Initializing, changing settings, or an error occurred. **Note** You can use this system-defined variable only for NX1P2 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_PLC_OptBoardSta*1			Member name	Error
Meaning	Option Board Error			Global/local	Global
Function	Indicates whether an Option Board error occurred. TRUE: An error occurred. FALSE: An error not occurred. **Note** You can use this system-defined variable only for NX1P2 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_PLC_OptSerialErrSta*1				
Meaning	Serial Option Board Error Status			Global/local	Global
Function	Contains the error status of a transmission error for the Serial Communications Option Board. When the Serial communications mode of a Serial Communications Option Board is only set to Host Link (FINS), the value of each member is updated. Other than the above setting, the values of all members are FALSE. The array element 1 corresponds to the option board slot 1 and array element 2 corresponds to the option board slot 2. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.				
Data type	ARRAY[1..2] OF _sOPTSERIALERR_STA			Range of values	---
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_PLC_OptSerialErrSta*1			Member name	Error
Meaning	Transmission Error			Global/local	Global
Function	Indicates whether a transmission error occurred. TRUE: A parity error, framing error, or an overrun error occurred. FALSE: FALSE due to one of the following causes. • A parity error, framing error, or an overrun error not occurred. • A port is restarted. • The Serial communications mode is not set to Host Link (FINS). • The Option Board that is mounted is not a Serial Communications Option Board. • The Option Board is not mounted. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_PLC_OptSerialErrSta*1			Member name	ParityErr
Meaning	Parity Error			Global/local	Global
Function	Indicates whether a parity error occurred. If this error occurs, it means that the serial communications settings may not apply to the remote device to connect. TRUE: A parity error occurred. FALSE: FALSE due to one of the following causes. • A parity error not occurred. • A port is restarted. • The Serial communications mode is not set to Host Link (FINS). • The Option Board that is mounted is not a Serial Communications Option Board. • The Option Board is not mounted. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_PLC_OptSerialErrSta*1			Member name	FramingErr
Meaning	Framing Error			Global/local	Global
Function	Indicates whether a framing error occurred. If this error occurs, it means that the serial communications settings may not apply to the remote device to connect. TRUE: A framing error occurred. FALSE: FALSE due to one of the following causes. • A framing error not occurred. • A port is restarted. • The Serial communications mode is not set to Host Link (FINS). • The Option Board that is mounted is not a Serial Communications Option Board. • The Option Board is not mounted. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_PLC_OptSerialErrSta*1			Member name	OverRun
Meaning	Overrun Error			Global/local	Global
Function	Indicates whether an overrun error occurred. If this error occurs, it means that the baud rate of an Option Board may be too large. TRUE: An overrun error occurred. FALSE: FALSE due to one of the following causes. • An overrun error not occurred. • A port is restarted. • The Serial communications mode is not set to Host Link (FINS). • The Option Board that is mounted is not a Serial Communications Option Board. • The Option Board is not mounted. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. **Note** You can use this system-defined variable only for NX1P2 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

● Functional Classification: Safety Data Logging

Variable name	_PLC_SFLogSta*1				
Meaning	Safety Data Logging Status			Global/local	Global
Function	Stores the status of safety data logging. Element number 0 corresponds to Logging Setting Number 1. Element number 1 corresponds to Logging Setting Number 2. **Note** You can use this system-defined variable only for the NX502 CPU Units and NX102 CPU Units.				
Data type	ARRAY[0..1] OF _sSFLOG_STA			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Not published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.31 of the CPU Unit.

Variable name	_PLC_SFLogSta*1			Member name	.IsStart
Meaning	Safety Data Logging Busy Flag			Global/local	Global
Function	TRUE when safety data logging starts. **Note** You can use this system-defined variable only for the NX502 CPU Units and NX102 CPU Units.				
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	---		

*1. This system-defined variable was added for unit version 1.31 of the CPU Unit.

Variable name	_PLC_SFLogSta*1			Member name	.IsComplete
Meaning	Safety Data Logging Completed Flag			Global/local	Global
Function	TRUE when logging stops. FALSE when the next logging starts. When this flag is TRUE, it means that the logging has completed. **Note** You can use this system-defined variable only for the NX502 CPU Units and NX102 CPU Units.				
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	---		

*1. This system-defined variable was added for unit version 1.31 of the CPU Unit.

Variable name	_PLC_SFLogSta*1			Member name	.IsOutput
Meaning	Log File Output Completed Flag			Global/local	Global
Function	TRUE when the log file is output. FALSE when the next logging starts. **Note** You can use this system-defined variable only for the NX502 CPU Units and NX102 CPU Units.				
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	---		

*1. This system-defined variable was added for unit version 1.31 of the CPU Unit.

A-8-3 PLC Function Module, Category Name: _CJB

● Functional Classification: I/O Bus Status

Variable name	_CJB_MaxRackNo				
Meaning	Largest Rack Number			Global/local	Global
Function	Contains the largest rack number of the Expansion Racks that are detected by the Controller. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	UINT			Range of values	0 to 3 "0" means there are no Expansion Racks.
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_CJB_MaxSlotNo				
Meaning	Largest Slot Number			Global/local	Global
Function	Contains one higher than the largest slot number with a CJ-series Unit on each of the Racks that are detected by the Controller. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	ARRAY [0..3] OF UINT			Range of values	0 to 10 0: No CJ-series Unit mounted.
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: I/O Bus Errors

Variable name	_CJB_ErrSta				
Meaning	I/O Bus Error Status			Global/local	Global
Function	Gives the I/O bus error status. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#C0F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instrucitons	You can access this variable from the user program only with the following instruction. • GetCJBError You can use the following instruction to clear this variable. • ResetCJBError		

Variable name	_CJB_MstrErrSta				
Meaning	I/O Bus Master Error Status			Global/local	Global
Function	Gives the I/O bus master error status. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	You can access this variable from the user program only with the following instruction. • GetCJBError You can use the following instruction to clear this variable. • ResetCJBError		

Variable name	_CJB_UnitErrSta				
Meaning	I/O Bus Unit Error Status			Global/local	Global
Function	Gives the error status of the I/O Bus Unit. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	ARRAY [0..3, 0..9] OF WORD			Range of values	16#0000 to 16#80F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	You can access this variable from the user program only with the following instruction. • GetCJBError You can use the following instruction to clear this variable. • ResetCJBError		

Variable name	_CJB_InRespTm				
Meaning	Basic Input Unit Input Response Times			Global/local	Global
Function	Contains the response times of the Basic Input Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type	ARRAY [0..3, 0..9] OF UINT			Range of values	0 to 320
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: Auxiliary Area Bits for CJ-series Units

Variable name		_CJB_IOUnitInfo				
Meaning		Basic I/O Unit Information			Global/local	Global
Function		Shows the status of the Basic I/O Unit alarm output (load short-circuit protection). TRUE: Load short-circuit FALSE: No load short-circuit **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type		ARRAY [0..3, 0..9, 0..7] OF BOOL			Range of values	TRUE or FALSE
R/W access		R	Retained	Not retained.	Network Publish	Not published.
Usage in user program		Possible.	Related instructions	---		
Auxiliary Area Addresses	Words	A50 to A69				
	Bits	A50.00 to A69.15				

Variable name		_CJB_CBU00InitSta to _CJB_CBU15InitSta				
Meaning		CPU Bus Unit Initializing Flags			Global/local	Global
Function		The corresponding variable is TRUE during initialization of the CPU Bus Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type		BOOL			Range of values	TRUE or FALSE
R/W access		R	Retained	Not retained.	Network Publish	Published.
Usage in user program		Possible.	Related instructions	• ResetUnit		
Auxiliary Area Addresses	Words	A302				
	Bits	A302.00 to A302.15				

Variable name		_CJB_SIO00InitSta to _CJB_SIO95InitSta				
Meaning		Special I/O Unit Initializing Flags			**Global/local**	Global
Function		The corresponding variable is TRUE during initialization of the Special I/O Unit. The corresponding variable changes to FALSE when the initialization is completed. The numbers in the variables indicate the unit numbers of the applicable Units. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type		BOOL			**Range of values**	TRUE or FALSE
R/W access		R	**Retained**	Not retained.	**Network Publish**	Published.
Usage in user program		Possible.	**Related instructions**	• ResetUnit		
Auxiliary Area Addresses	**Words**	A330 to A335				
	Bits	A330.00 to A335.15				

Variable name		_CJB_CBU00Restart to _CJB_CBU15Restart				
Meaning		CPU Bus Unit Restart Bits			**Global/local**	Global
Function		The CPU Bus Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the CPU Bus Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Bit to TRUE with an instruction, the restart process begins from refresh processing in the next task period. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type		BOOL			**Range of values**	TRUE or FALSE
R/W access		RW	**Retained**	Not retained.	**Network Publish**	Published.
Usage in user program		Possible.	**Related instructions**	• ResetUnit		
Auxiliary Area Addresses	**Words**	A501				
	Bits	A501.00 to A501.15				

Variable name		_CJB_SIO00Restart to _CJB_SIO95Restart				
Meaning		Special I/O Unit Restart Bits			**Global/local**	Global
Function		The Special I/O Unit is restarted when the corresponding variable changes to TRUE. (It is changed to FALSE by the system after the Special I/O Unit is restarted.) The numbers in the variables indicate the unit numbers of the applicable Units. If you change the Restart Bit to TRUE with an instruction, the restart process begins from refresh processing in the next task period. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type		BOOL			**Range of values**	TRUE or FALSE
R/W access		RW	**Retained**	Not retained.	**Network Publish**	Published.
Usage in user program		Possible.	**Related instructions**	• ResetUnit		
Auxiliary Area Addresses	**Words**	A502 to A507				
	Bits	A502.00 to A507.15				

Variable name		_CJB_SCU00P1ChgSta _CJB_SCU00P2ChgSta to _CJB_SCU15P1ChgSta _CJB_SCU15P2ChgSta				
Meaning		Serial Communications Unit 0, Port 1/2 Settings Changing Flags to Serial Communications Units 1 to 15, Port 1/2 Settings Changing Flags			Global/local	Global
Function		TRUE when the parameters of the specified port are being changed. FALSE after the parameters are changed. It is also possible for the user to indicate a change in serial port settings by turning ON the corresponding flag through the execution of an instruction or a user operation. **Note** You can use this system-defined variable only for NJ-series CPU Units.				
Data type		BOOL			Range of values	TRUE or FALSE
R/W access		RW	Retained	Not retained.	Network Publish	Published.
Usage in user program		Possible.	Related instructions	---		
Auxiliary Area Addresses	Words	Port on Serial Communications Unit with unit number 0: A620 Ports on Serial Communications Unit with unit numbers 1 to 15: A621 to A635				
	Bits	Port on Serial Communications Unit with unit number 0: A620.01 to A620.02 Ports on Serial Communications Unit with unit numbers 1 to 15: A621.01 to A635.02				

A-8-4 PLC Function Module, Category Name: _APB

● Functional Classification: Automation Playback

Variable name	_APB_Status*1			Member name	.Status
Meaning	APB Service Operating Status			Global/local	Global
Function	Indicates the operating status of the APB service. 0: Disabled 1: Initializing 2: Idle 3: Run 4: Error **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	UINT			Range of values	0 to 4
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 1.63 of the CPU Unit.

Variable name	_APB_Status*1			Member name	.LatestLogName
Meaning	Latest Variable Log File Name			Global/local	Global
Function	Indicates the file name of the most recent variable log that was saved to storage. **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	STRING[32]			Range of values	Depends on data type.
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 1.63 of the CPU Unit.

Variable name	_APB_Status*1			Member name	.LatestLogTime
Meaning	Latest Variable Log Output Date and Time			Global/local	Global
Function	Indicates the most recent date and time when the variable log was output and saved to storage. Note that the value of this member may not match the date and time included in the file name. **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	DATE_AND_TIME			Range of values	Depends on data type.
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 1.63 of the CPU Unit.

Variable name	_APB_Status*1			Member name	.IsProjectBackupFinished
Meaning	APB Project Backup Finished			Global/local	Global
Function	Indicates the status of backing up the project with automation playback. **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.63 of the CPU Unit.

Variable name	_APB_Status*1			Member name	.ActiveSamplingSettingNumber
Meaning	Active Sampling Setting Number			Global/local	Global
Function	Indicates the active collection setting number in automation playback. **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	USINT			Range of values	1 or 2
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	APB_ChangeSamplingSettings		

*1. This system-defined variable was added for unit version 1.63 of the CPU Unit.

Variable name	_APB_Status*1			Member name	.ActiveSamplingSettingName
Meaning	Active Sampling Setting Name			Global/local	Global
Function	Indicates the name of active collection setting in automation playback. **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	STRING[32]			Range of values	Depends on data type.
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	APB_ChangeSamplingSettings		

*1. This system-defined variable was added for unit version 1.63 of the CPU Unit.

Variable name	_APB_Status*1			Member name	.LatestErrorID
Meaning	APB Instruction Error Code			Global/local	Global
Function	Indicates the error code of the last automation playback-related instruction detected. **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	WORD			Range of values	Depends on data type.
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 1.63 of the CPU Unit.

Variable name	_APB_Status*1			Member name	.LatestEventCode
Meaning	APB Service Event Code			Global/local	Global
Function	Indicates the last automation playback-related event code detected. **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	DWORD			Range of values	Depends on data type.
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 1.63 of the CPU Unit.

Variable name	_APB_LogStatus*1			Member name	.Status
Meaning	Variable Log Output Setting Status			Global/local	Global
Function	Indicates the operating status of the variable log output settings. 0: Disabled 1: Stopped 2: Sampling 3: Triggered 4: Sampled 5: Storing 6: Error **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	UINT			Range of values	0 to 6
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 1.63 of the CPU Unit.

Variable name	_APB_LogStatus*1			Member name	.SettingName
Meaning	Variable Log Output Setting Name			Global/local	Global
Function	Indicates the setting name of the variable log output settings. **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	STRING[32]			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	---		

*1. This system-defined variable was added for unit version 1.63 of the CPU Unit.

Variable name	_APB_LogStatus*1			Member name	.SettingType
Meaning	Variable Log Output Setting Type			Global/local	Global
Function	Indicates the method of variable log output settings. 0: Pre/Post trigger method 1: Start/Save trigger method **Note** You can use this system-defined variable only for the NX502 CPU Unit.				
Data type	UINT			Range of values	0 or 1
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Not possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.63 of the CPU Unit.

A-8-5 NX Bus Function Module, Category Name: _NXB

● Functional Classification: NX Bus Function Module Status

Variable name	_NXB_MaxUnitNo*1				
Meaning	Largest Unit Number			Global/local	Global
Function	Contains the largest NX Unit number of the NX Units on the CPU Unit that are detected by the NX Bus Function Module. If the Unit configuration information is registered by the Sysmac Studio, the value will be largest NX Unit number of the registered Unit configuration. Units that are set as unmounted Units are also included. If the Unit configuration information is not registered by the Sysmac Studio, the value will be the largest Unit number of an actual Unit configuration. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.				
Data type	UINT			Range of values	0 to 32*2 0: No NX Unit mounted.
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 1.13 of the CPU Unit.
*2. For the NX502 CPU Units, the range of values is 0 to 63.
For the NX1P2 CPU Units, the range of values is 0 to 8.

Variable name	_NXB_UnitIOActiveTbl*1				
Meaning	NX Unit I/O Data Active Status			Global/local	Global
Function	Indicates whether the I/O data in the NX Units on the CPU Unit is valid. This status is given as an array of BOOL data. The subscript of the array corresponds to the NX Unit number. A subscript of 0 indicates the NX Bus Function Module and it is always TRUE. TRUE: The I/O data in the NX Unit is valid. FALSE: The I/O data in the NX Unit is invalid. The status is FALSE for NX Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.				
Data type	ARRAY [0..32] OF BOOL*2			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• ResetNXBError • RestartNXUnit		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.
*2. For the NX502 CPU Units, the data type is ARRAY [0..63] OF BOOL.
For the NX1P2 CPU Units, the data type is ARRAY [0..8] OF BOOL.

Variable name	_NXB_UnitMsgActiveTbl*1				
Meaning	NX Unit Message Enabled Status			Global/local	Global
Function	Indicates whether the NX Units on the CPU Unit can process message communications. This status is given as an array of BOOL data. The subscript of the array corresponds to the NX Unit number. A subscript of 0 indicates the NX Bus Function Module and it is always TRUE. TRUE: Message communications possible. FALSE: Message communications not possible. The status is FALSE for NX Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.				
Data type	ARRAY [0..32] OF BOOL*2			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• ResetNXBError • RestartNXUnit		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

*2. For the NX502 CPU Units, the data type is ARRAY [0..63] OF BOOL.
For the NX1P2 CPU Units, the data type is ARRAY [0..8] OF BOOL.

Variable name	_NXB_UnitRegTbl*1				
Meaning	NX Unit Registration Status			Global/local	Global
Function	Indicates whether the NX Units on the CPU Unit are registered in the Unit configuration. This status is given as an array of BOOL data. The subscript of the array corresponds to the NX Unit number. A subscript of 0 indicates the NX Bus Function Module. TRUE: Registered. FALSE: Not registered. If the Unit configuration information is not registered by the Sysmac Studio, the status is FALSE for all Units. The status is TRUE for NX Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.				
Data type	ARRAY [0..32] OF BOOL*2			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 1.13 of the CPU Unit.

*2. For the NX502 CPU Units, the data type is ARRAY [0..63] OF BOOL.
For the NX1P2 CPU Units, the data type is ARRAY [0..8] OF BOOL.

● Functional Classification: NX Bus Function Module Errors

Variable name	_NXB_ErrSta*1				
Meaning	NX Bus Function Module Error Status			Global/local	Global
Function	Gives the NX Bus Function Module error status. This system-defined variable provides the collective status of the NX Bus Function Module Master Error Status and NX Bus Function Module Unit Error Status for all NX Units. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#40F2
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 only with the following instruction. • GetNXBError You can use the following instructions to clear this variable. • ResetNXBError • RestartNXUnit		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_NXB_MstrErrSta*1				
Meaning	NX Bus Function Module Master Error Status			Global/local	Global
Function	Gives the status of errors that are detected in the NX Bus Function Module of the CPU Unit. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#40F2
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 only with the following instruction. • GetNXBError You can use the following instructions to clear this variable. • ResetNXBError • RestartNXUnit		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_NXB_UnitErrStaTbl*1				
Meaning	NX Bus Function Module Unit Error Status			Global/local	Global
Function	Gives the status of errors that are detected in the NX Bus Function Module of the CPU Unit. This status is given as an array of WORD data. The subscript of the array corresponds to the NX Unit number. **Note** We do not recommend the use of this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device such as an HMI. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.				
Data type	ARRAY [1..32] OF WORD*2			Range of values	16#0000 to 16#40F2
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 only with the following instruction. • GetNXBError You can use the following instructions to clear this variable. • ResetNXBError • RestartNXUnit		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.
*2. For the NX502 CPU Units, the data type is ARRAY [0..63] OF BOOL.
For the NX1P2 CPU Units, the data type is ARRAY [0..8] OF WORD.

Variable name	_NXB_UnitErrFlagTbl*1				
Meaning	NX Unit Error Status			Global/local	Global
Function	Indicates whether errors occurred in the NX Unit on the CPU Unit. This status is given as an array of BOOL data. The subscript of the array corresponds to the NX Unit number. A subscript of "0" indicates the NX Bus Function Module and whether an event occurred that is detected by the NX Bus Function Module. TRUE: Error. FALSE: No error. The status is "FALSE" for NX Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.				
Data type	ARRAY [0..32] OF BOOL*2			Range of values	TRUE or FALSE
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 only with the following instruction. • GetNXBError You can use the following instructions to clear this variable. • ResetNXBError • RestartNXUnit		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.
*2. For the NX502 CPU Units, the data type is ARRAY [0..63] OF BOOL.
For the NX1P2 CPU Units, the data type is ARRAY [0..8] OF BOOL.

A-8-6 X Bus Function Module, Category Name: _XBU

● Functional Classification: X Bus Function Module Status

Variable name	_XBU_RegTbl				
Meaning	X Bus Unit Registration Status			Global/local	Global
Function	This status tells whether the X Bus Units are registered in the Unit configuration. This status is given as an array of BOOL data. The suffix of the array represents the X Bus Unit number. A suffix of 0 is given to the X Bus Function Module. TRUE: Registered. FALSE: Not registered. If the Unit configuration information is not registered from the Sysmac Studio, the status is FALSE for all Units. The status is TRUE for X Bus Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	ARRAY [0..4] 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	---		

● Functional Classification: X Bus Function Module Errors

Variable name	_XBU_ErrSta				
Meaning	X Bus Function Module Error Status			Global/local	Global
Function	Gives the status of errors that are detected in the X Bus Function Module. Gives the collective value of the *X Bus Function Module Master Error Status* and *X Bus Function Module Unit Error Status* for all X Bus Units. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#40F2
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 only with the following instruction. • GetXBUError You can use the following instructions to clear this variable. • ResetXBUError		

Variable name	_XBU_MstrErr				
Meaning	X Bus Function Module Master Error Status			Global/local	Global
Function	Gives the status of errors that are detected in the X Bus master. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#40F2
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 only with the following instruction. • GetXBUError You can use the following instructions to clear this variable. • ResetXBUError		

Variable name	_XBU_UnitErr				
Meaning	X Bus Function Module Unit Error Status			Global/local	Global
Function	Gives the collective value of all *X Bus Function Module Unit Error Status Table*. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#40F2
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 only with the following instruction. • GetXBUError You can use the following instructions to clear this variable. • ResetXBUError		

Variable name	_XBU_UnitErrTbl				
Meaning	X Bus Function Module Unit Error Status Table			Global/local	Global
Function	Gives the status of errors that are detected in the X Bus Unit. The suffix of the array represents the X Bus Unit number. The status is "16#0000" for X Bus Units that are set as unmounted Units. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	ARRAY [1..4] OF WORD			Range of values	16#0000 to 16#40F2
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 only with the following instruction. • GetXBUError You can use the following instructions to clear this variable. • ResetXBUError		

A-8-7 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-7-10 Meanings of Error Status Bits* on page A-161 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-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• GetMCError • ResetMCError		

Variable name	_MC_AX_ErrSta				
Meaning	Axis Error Status			Global/local	Global
Function	Shows the error status for each axis. The status of up to 256 axes*1 is shown. You can use this variable directly in the user program. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits.				
Data type	ARRAY [0..255] OF WORD*1			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• GetMCError • ResetMCError • MC_Reset		

*1. For NX502 CPU Units, the error status of up to 128 axes is shown and the data type is ARRAY [0..127] OF WORD.
For the NX102 CPU Units and NX1P2 CPU Units, the error status of up to 16 axes is shown and the data type is ARRAY [0..15] OF WORD.
For NJ-series CPU Units, the error status of up to 64 axes is shown and the data type is ARRAY [0..63] OF WORD.

Variable name	_MC_GRP_ErrSta				
Meaning	Axes Group Error Status			Global/local	Global
Function	Shows the error status for each axes group. The error status for up to 64 axes groups *1 is shown. You can use this variable directly in the user program. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits.				
Data type	ARRAY [0..63] OF WORD*1			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	• GetMCError • ResetMCError • MC_GroupReset		

*1. For NX502 CPU Units, the error status of up to 32 axes groups is shown and the data type is ARRAY [0..31] OF WORD.
For the NX102 CPU Units and NX1P2 CPU Units, the error status of up to 8 axes groups is shown and the data type is ARRAY [0..7] OF WORD.
For NJ-series CPU Units, the error status of up to 32 axes groups is shown and the data type is ARRAY [0..31] OF WORD.

Variable name	_MC_COM				
Meaning	Common Variable			Global/local	Global
Function	Shows the status that is common to the Motion Control Function Module. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members.				
Data type	_sCOMMON_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_MC_GRP				
Meaning	Axes Group Variables			Global/local	Global
Function	NX701 CPU Units: Used to specify axes groups and shows multi-axes coordinated control status, and multi-axes coordinated control settings for motion control instructions used for motion control 1. NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units: Used to specify axes groups and shows multi-axes coordinated control status, and multi-axes coordinated control settings for motion control instructions. When you create an axes group on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members.				
Data type	ARRAY[0..63] OF _sGROUP_REF*1			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. For NX502 CPU Units, the error status of up to 32 axes groups is shown and the data type is ARRAY [0..31] OF _sGROUP_REF.
For the NX102 CPU Units and NX1P2 CPU Units, the data type is ARRAY [0..7] OF _sGROUP_REF.
For NJ-series CPU Units, the data type is ARRAY[0..31] OF _sGROUP_REF.

Variable name	_MC1_GRP				
Meaning	Axes Group Variables			Global/local	Global
Function	Used to specify axes groups and shows multi-axes coordinated control status, and multi-axes coordinated control settings for motion control instructions used for motion control 1. When you create an axes group on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members. **Note** You can use this system-defined variable only for the NX701 CPU Units. You can access the same values of *_MC1_GRP* and *_MC_GRP* if the array element numbers of them are the same.				
Data type	ARRAY[0..63] OF _sGROUP_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_MC2_GRP				
Meaning	Axes Group Variables			Global/local	Global
Function	Used to specify axes groups and shows multi-axes coordinated control status, and multi-axes coordinated control settings for motion control instructions used for motion control 2. When you create an axes group on the System Studio, a user-defined axes group variable with a different name is created. Normally, you use an Axes Group Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members. **Note** You can use this system-defined variable only for the NX701 CPU Units.				
Data type	ARRAY[0..63] OF _sGROUP_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_MC_AX				
Meaning	Axis Variables			Global/local	Global
Function	NX701 CPU Units: Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 1. NX502 CPU Units, NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units: Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions. When you create an axis on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members.				
Data type	ARRAY[0..255] OF _sAXIS_REF*1			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. For NX502 CPU Units, the error status of up to 128 axes is shown and the data type is ARRAY [0..127] OF _sAXIS_REF.
For the NX102 CPU Units and NX1P2 CPU Units, the data type is ARRAY [0..15] OF _sAXIS_REF.
For NJ-series CPU Units, the data type is ARRAY[0..63] OF _sAXIS_REF.

Variable name	_MC1_AX				
Meaning	Axis Variables			Global/local	Global
Function	Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 1. When you create an axis on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members. **Note** You can use this system-defined variable only for the NX701 CPU Units. You can access the same values of *_MC1_AX* and *_MC_AX* if the array element numbers of them are the same.				
Data type	ARRAY[0..255] OF _sAXIS_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_MC2_AX				
Meaning	Axis Variables			Global/local	Global
Function	Used to specify axes and shows single-axis control status, and single-axis control settings for motion control instructions used for motion control 2. When you create an axis on the System Studio, a user-defined axis variable with a different name is created. Normally, you use an Axis Variable with a different name. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on structure members. **Note** You can use this system-defined variable only for the NX701 CPU Units.				
Data type	ARRAY[0..255] OF _sAXIS_REF			Range of values	---
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

A-8-8 EtherCAT Master Function Module, Category Name: _EC

● Functional Classification: EtherCAT Communications Errors

Variable name	_EC_ErrSta				
Meaning	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-7-10 Meanings of Error Status Bits* on page A-161 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 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-7-10 Meanings of Error Status Bits* on page A-161 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 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-7-10 Meanings of Error Status Bits* on page A-161 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 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-7-10 Meanings of Error Status Bits* on page A-161 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 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. This error status shows status of error in each slave in the actual system configuration. This variable array indicates slaves in which there are errors. Status is provided for each EtherCAT slave node address (1 to 512)*1. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits.				
Data type	Array [1..512] OF WORD*1			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Get EtherCAT Error Status • GetECError Reset EtherCAT Error • ResetECError		

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the node address is 1 to 192 and the data type is ARRAY [1..192] OF WORD.
For the NX502 CPU Units, the node address is 1 to 256 and the data type is Array [1..256] OF WORD.

Variable name	_EC_MacAdrErr				
Meaning	MAC Address Error			Global/local	Global
Function	TRUE if the MAC Address Error (14400000 hex) event occurred.				
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_LanHwErr				
Meaning	Communications Controller Error			Global/local	Global
Function	TRUE if the Communications Controller Error (047C0000 hex) event occurred.				
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_LinkOffErr				
Meaning	Link OFF Error			Global/local	Global
Function	TRUE if the Link OFF Error (84200000 hex) event occurred.				
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 Error • ResetECError		

Variable name	_EC_NetCfgErr				
Meaning	Network Configuration Information Error			Global/local	Global
Function	TRUE if the Network Configuration Information Error (34400000 hex) event occurred.				
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_NetCfgCmpErr				
Meaning	Network Configuration Verification Error			Global/local	Global
Function	TRUE if one of the following events occurred. • Network Configuration Verification Error (84220000 hex) • Network Configuration Verification Error (Slave Unconnected) (84380000 hex) • Network Configuration Verification Error (Unnecessary Slave Connected) (84320003 hex) • Network Configuration Verification Error (Mismatched Slave) (84330004 hex) • Network Configuration Verification Error (Incorrect Ring Wiring) (843A0000 hex)				
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 Error • ResetECError		

Variable name	_EC_NetTopologyErr				
Meaning	Network Configuration Error			Global/local	Global
Function	TRUE if one of the following events occurred. • Network Configuration Error (84210000 hex) • Incorrect Wiring Detected (843C0000 hex)				
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 Error • ResetECError		

Variable name	_EC_PDCommErr				
Meaning	Process Data Communications Error			Global/local	Global
Function	TRUE if one of the following events occurred. • Process Data Communications Error (842C0000 hex) • Illegal Slave Disconnection Detected (84310002 hex) • Slave PDI WDT Error Detected (84340000 hex)				
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 Error • ResetECError		

Variable name	_EC_PDTimeoutErr				
Meaning	Process Data Reception Timeout Error			Global/local	Global
Function	TRUE if the Process Data Reception Timeout (842B0000 hex) event occurred.				
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 Error • ResetECError		

Variable name	_EC_PDSendErr				
Meaning	Process Data Transmission Error			Global/local	Global
Function	TRUE if the Process Data Transmission Error (84290000 hex) event occurred.				
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 Error • ResetECError		

Variable name	_EC_SlavAdrDupErr				
Meaning	Slave Node Address Duplicated Error			Global/local	Global
Function	TRUE if the Slave Node Address Duplicated (24200000 hex) event occurred.				
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 Error • ResetECError		

Variable name	_EC_SlavInitErr				
Meaning	Slave Initialization Error			Global/local	Global
Function	TRUE if one of the following events occurred. • Slave Initialization Error (84230000 hex) • Slave State Transition Failed (84300001 hex)				
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 Error • ResetECError		

Variable name	_EC_SlavAppErr				
Meaning	Slave Application Error			Global/local	Global
Function	TRUE if one of the following events occurred. • Slave Application Error (84280000 hex) • Slave AL Status Error Detected (84360000 hex)				
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 Error • ResetECError		

Variable name	_EC_MsgErr				
Meaning	EtherCAT Message Error			Global/local	Global
Function	TRUE if one of the following events occurred. • EtherCAT Message Error (842D0000 hex) • Illegal Mailbox Received (84350000 hex)				
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 EtherCAT CoE SDO) • EC_CoESDORead CoE messages (Write EtherCAT CoE SDO) • EC_CoESDOWrite		

Variable name	_EC_SlavEmergErr				
Meaning	Emergency Message Detected			Global/local	Global
Function	TRUE if the Emergency Message Detected (64200000 hex) event occurred.				
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 Error • ResetECError		

Variable name	_EC_IndataInvalidErr*1				
Meaning	Input Process Data Invalid Error			Global/local	Global
Function	TRUE if the Input Process Data Invalid Error (842F0000 hex) event occurred.				
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 Error • ResetECError		

*1. This system-defined variable was added for unit version 1.13 of the CPU Unit.

Variable name	_EC_CommErrTbl				
Meaning	Communications Error Slave Table			Global/local	Global
Function	Slaves are given in the table in the order of slave node addresses. The corresponding slave element is TRUE if the master detected an error for the slave.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Reset EtherCAT Error • ResetECError		

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the data type is ARRAY [1..192] OF BOOL.
For the NX502 CPU Units, the data type is Array [1..256] OF BOOL.

Note The values of all system-defined variables that are related to errors in EtherCAT communications do not change until the cause of the error is removed and then the error in the Controller is reset with the troubleshooting functions of the Sysmac Studio or the ResetECError instruction.

Variable name	_EC_CycleExceeded				
Meaning	EtherCAT Communications Cycle Exceeded			Global/local	Global
Function	TRUE if the EtherCAT Communications Cycle Exceeded (34410000 hex) event occurred. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: EtherCAT Communications Status

Variable name	_EC_RegSlavTbl				
Meaning	Registered Slave Table			Global/local	Global
Function	This table indicates the slaves that are registered in the network configuration information. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave is registered.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the data type is ARRAY [1..192] OF BOOL.
For the NX502 CPU Units, the data type is Array [1..256] OF BOOL.

Variable name	_EC_EntrySlavTbl				
Meaning	Network Connected Slave Table			Global/local	Global
Function	This table indicates which slaves are connected to the network. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave has entered the network.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the data type is ARRAY [1..192] OF BOOL.
For the NX502 CPU Units, the data type is Array [1..256] OF BOOL.

Variable name	_EC_MBXSlavTbl				
Meaning	Message Communications Enabled Slave Table			Global/local	Global
Function	This table indicates the slaves that can perform message communications. Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if message communications are enabled for it (pre-operational, safe-operation, or operational state). **Note** Use this variable to confirm that message communications are possible for the relevant slave before you execute message communications with an EtherCAT slave.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Disconnect EtherCAT Slave • EC_DisconnectSlave Connect EtherCAT Slave • EC_ConnectSlave		

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the data type is ARRAY [1..192] OF BOOL.
For the NX502 CPU Units, the data type is Array [1..256] OF BOOL.

Variable name	_EC_PDSlavTbl				
Meaning	Process Data Communicating Slave Table			Global/local	Global
Function	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.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Disconnect EtherCAT Slave • EC_DisconnectSlave Connect EtherCAT Slave • EC_ConnectSlave		

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the data type is ARRAY [1..192] OF BOOL.
For the NX502 CPU Units, the data type is Array [1..256] OF BOOL.

Variable name	_EC_DisconnSlavTbl				
Meaning	Disconnected Slave Table			Global/local	Global
Function	Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave was disconnected.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	Disconnect EtherCAT Slave • EC_DisconnectSlave Connect EtherCAT Slave • EC_ConnectSlave		

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the data type is ARRAY [1..192] OF BOOL.
For the NX502 CPU Units, the data type is Array [1..256] OF BOOL.

Variable name	_EC_DisableSlavTbl				
Meaning	Disabled Slave Table			Global/local	Global
Function	Slaves are given in the table in the order of slave node addresses. The element for a slave is TRUE if the corresponding slave is disabled.				
Data type	Array [1..512] OF BOOL*1			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. For the NX102 CPU Units, NX1P2 CPU Units, and NJ-series CPU Units, the data type is ARRAY [1..192] OF BOOL.
For the NX502 CPU Units, the data type is Array [1..256] OF BOOL.

Variable name	_EC_PDActive				
Meaning	Process Data Communications Status			Global/local	Global
Function	TRUE when process data communications are performed with all slaves*1.				
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		

*1. 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 established 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 1. All system-defined variables that are related to the status of EtherCAT communications give the current status.
Note 2. The variable temporarily changes to TRUE if the EC_DisconnectSlave (Disconnect EtherCAT Slave) instruction, EC_ConnectSlave (Connect EtherCAT Slave) instruction, or EC_ChangeEnableSetting (Enable/Disable EtherCAT Slave) instruction is executed.

Variable name	_EC_InData1Invalid				
Meaning	Input Data1 Invalid			Global/local	Global
Function	TRUE when process data communications established in the primary periodic task are not normal and the input data is not valid. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Note 1. All system-defined variables that are related to the status of EtherCAT communications give the current status.
Note 2. The variable temporarily changes to TRUE if the EC_DisconnectSlave (Disconnect EtherCAT Slave) instruction, EC_ConnectSlave (Connect EtherCAT Slave) instruction, or EC_ChangeEnableSetting (Enable/Disable EtherCAT Slave) instruction is executed.

Variable name	_EC_InData2Invalid				
Meaning	Input Data2 Invalid			Global/local	Global
Function	TRUE when process data communications established in the priority-5 periodic task are not normal and the input data is not valid. **Note** You can use this system-defined variable only for NX-series CPU Units. **Note** This variable is always TRUE for the NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Note 1. All system-defined variables that are related to the status of EtherCAT communications give the current status.

Note 2. The variable temporarily changes to TRUE if the EC_DisconnectSlave (Disconnect EtherCAT Slave) instruction, EC_ConnectSlave (Connect EtherCAT Slave) instruction, or EC_ChangeEnableSetting (Enable/Disable EtherCAT Slave) instruction is executed.

Variable name	_EC_RingBreaking*1				
Meaning	Ring Disconnection			Global/local	Global
Function	TRUE when all slaves in the ring topology except for disabled slaves are connected and if there is only one point of disconnection in the communications cables in the ring topology.				
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 1.40 of the CPU Unit.

Variable name	_EC_RingBreakNodeAdr*1				
Meaning	Slave Node Address Before Ring Disconnection			Global/local	Global
Function	When the *_EC_RingBreaking* (Ring Disconnection) system-defined variable is TRUE, the slave node address before point of disconnection is stored. When the *_EC_RingBreaking* (Ring Disconnection) system-defined variable is FALSE, “0” is stored.				
Data type	UINT			Range of values	0 to maximum number of node addresses
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 1.40 of the CPU Unit.

● Functional Classification: EtherCAT Communications Diagnosis/Statistics Log

Variable name	_EC_StatisticsLogEnable*1				
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	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.

Variable name	_EC_StatisticsLogCycleSec*1				
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	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.

Variable name	_EC_StatisticsLogBusy*1				
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	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.

Variable name	_EC_StatisticsLogErr*1				
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 SD Memory Card is fully used. • The SD Memory Card is write-protected. • There is no 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	---		

*1. This system-defined variable was added for unit version 1.11 of the CPU Unit.

A-8-9 EtherNet/IP Function Module, Category Name: _EIP

● Functional Classification: EtherNet/IP Communications Errors

Variable name	_EIP_ErrSta				
Meaning	EtherNet/IP Error			Global/local	Global
Function	This is the error status variable for the built-in EtherNet/IP port. NX-series CPU Units: Represents the collective status of the following error flags. • _EIP1_PortErr (Communications Port1 Error) • _EIP2_PortErr (Communications Port2 Error) • _EIP1_CipErr (CIP Communications1 Error) • _EIP2_CipErr (CIP Communications2 Error) • _EIP_TcpAppErr (TCP Application Communications Error) NJ-series CPU Units: Represents the collective status of the following error flags. • _EIP_PortErr (Communications Port Error) • _EIP_CipErr (CIP Communications Error) • _EIP_TcpAppErr (TCP Application Communications Error) **Note** Refer to *A-7-10 Meaning of Error Status Bits* on page A-161 for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP_PortErr				
Meaning	Communications Port Error			Global/local	Global
Function	This is the error status variable for the communications port. NX-series CPU Units: Represents the collective status of the following error flags. • _EIP1_MacAdrErr (Port1 MAC Address Error) • _EIP1_LanHwErr (Port1 Communications Controller Error) • _EIP1_EtnCfgErr (Port1 Basic Ethernet Setting Error) • _EIP1_IPAdrCfgErr (Port1 IP Address Setting Error) • _EIP1_IPAdrDupErr (Port1 IP Address Duplication Error) • _EIP1_BootpErr (Port1 BOOTP Server Error) • _EIP1_DhcpErr (Port1 DHCP Server Error) • _EIP_DNSCfgErr (DNS Setting Error) • _EIP_DNSSrvErr (DNS Server Connection Error) • _EIP_IPRTblErr (IP Route Table Error) NJ-series CPU Units: Represents the collective status of the following error flags. • _EIP_MacAdrErr (MAC Address Error) • _EIP_LanHwErr (Communications Controller Error) • _EIP_EtnCfgErr (Basic Ethernet Setting Error) • _EIP_IPAdrCfgErr (IP Address Setting Error) • _EIP_IPAdrDupErr (IP Address Duplication Error) • _EIP_BootpErr (BOOTP Server Error) • _EIP_DNSSrvErr (DNS Server Connection Error) • _EIP_IPRTblErr (IP Route Table Error) **Note** If a *Link OFF Detected* or *EtherNet/IP Error* occurs, it is recorded in the event log and then the corresponding bit turns ON. Refer to *A-7-10 Meaning of Error Status Bits* on page A-161 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) • _EIP1_DhcpErr (Port1 DHCP 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 *EtherNet/IP Error* occurs, it is recorded in the event log and then the corresponding bit turns ON. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP2_PortErr				
Meaning	Communications Port2 Error			Global/local	Global
Function	This is the error status variable for the communications port 2. It represents the collective status of the following error flags. • _EIP2_MacAdrErr (Port2 MAC Address Error) • _EIP2_LanHwErr (Port2 Communications Controller Error) • _EIP2_EtnCfgErr (Port2 Basic Ethernet Setting Error) • _EIP2_IPAdrCfgErr (Port2 IP Address Setting Error) • _EIP2_IPAdrDupErr (Port2 IP Address Duplication Error) • _EIP2_BootpErr (Port2 BOOTP Server Error) • _EIP2_DhcpErr (Port2 DHCP 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 *EtherNet/IP Error* occurs, it is recorded in the event log and then the corresponding bit turns ON. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP_CipErr				
Meaning	CIP Communications Error			Global/local	Global
Function	This is the error status variable for CIP communications. NX-series CPU Units: Represents the collective status of the following error flags. • _EIP1_IdentityErr (CIP Communications1 Identity Error) • _EIP1_TDLinkCfgErr (CIP Communications1 Tag Data Link Setting Error) • _EIP1_TDLinkOpnErr (CIP Communications1 Tag Data Link Connection Failed) • _EIP1_TDLinkErr (CIP Communications1 Tag Data Link Communications Error) • _EIP1_TagAdrErr (CIP Communications1 Tag Name Resolution Error) • _EIP1_MultiSwONErr (CIP Communications1 Multiple Switches ON Error) NJ-series CPU Units: Represents the collective status of the following error flags. • _EIP_IdentityErr (Identity Error) • _EIP_TDLinkCfgErr (Tag Data Link Setting Error) • _EIP_TDLinkOpnErr (Tag Data Link Connection Failed) • _EIP_TDLinkErr (Tag Data Link Communications Error) • _EIP_TagAdrErr (Tag Name Resolution Error) • _EIP_MultiSwOnErr (Multiple Switches ON Error) **Note** If a Tag Name Resolution Error occurs, it is recorded in the event log and this variable changes to TRUE. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 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_CipErr				
Meaning	CIP Communications1 Error			Global/local	Global
Function	This is the error status variable for CIP communications 1. It represents the collective status of the following error flags. • _EIP1_IdentityErr (CIP Communications1 Identity Error) • _EIP1_TDLinkCfgErr (CIP Communications1 Tag Data Link Setting Error) • _EIP1_TDLinkOpnErr (CIP Communications1 Tag Data Link Connection Failed) • _EIP1_TDLinkErr (CIP Communications1 Tag Data Link Communications Error) • _EIP1_TagAdrErr (CIP Communications1 Tag Name Resolution Error) • _EIP1_MultiSwONErr (CIP Communications1 Multiple Switches ON Error) **Note** If a Tag Name Resolution Error occurs, it is recorded in the event log and this variable changes to TRUE. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP2_CipErr				
Meaning	CIP Communications2 Error			Global/local	Global
Function	This is the error status variable for CIP communications 2. It represents the collective status of the following error flags. • _EIP2_IdentityErr (CIP Communications2 Identity Error) • _EIP2_TDLinkCfgErr (CIP Communications2 Tag Data Link Setting Error) • _EIP2_TDLinkOpnErr (CIP Communications2 Tag Data Link Connection Failed) • _EIP2_TDLinkErr (CIP Communications2 Tag Data Link Communications Error) • _EIP2_TagAdrErr (CIP Communications2 Tag Name Resolution Error) • _EIP2_MultiSwONErr (CIP Communications2 Multiple Switches ON Error) **Note** If a Tag Name Resolution Error occurs, it is recorded in the event log and this variable changes to TRUE. Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP_TcpAppErr				
Meaning	TCP Application Communications Error			Global/local	Global
Function	This is the error status variable for TCP application communications. It represents the collective status of the following error flags. • _EIP_TcpAppCfgErr (TCP Application Setting Error) • _EIP_NTPSrvErr (NTP Server Connection Error) **Note** Refer to *A-7-10 Meanings of Error Status Bits* on page A-161 for the meanings of the error status bits.				
Data type	WORD			Range of values	16#0000 to 16#00F0
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	You can access this variable from the user program with the following instruction. • GetEIPError		

Variable name	_EIP_MacAdrErr				
Meaning	MAC Address Error			Global/local	Global
Function	NX-series CPU Units: Indicates that an error occurred when the MAC address was read on the communications port 1 at startup. TRUE: Error FALSE: Normal NJ-series CPU Units: Indicates that an error occurred when the MAC address was read at startup. TRUE: Error FALSE: Normal				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

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 **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_MacAdrErr				
Meaning	Port2 MAC Address Error			Global/local	Global
Function	Indicates that an error occurred when the MAC address was read on the communications port 2 at startup. TRUE: Error FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_LanHwErr				
Meaning	Communications Controller Error			Global/local	Global
Function	NX-series CPU Units: Indicates that a Communications Controller failure occurred on the communications port 1. TRUE: Failure FALSE: Normal NJ-series CPU Units: Indicates that a Communications Controller failure occurred. TRUE: Failure FALSE: Normal				
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 **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TTRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_LanHwErr				
Meaning	Port2 Communications Controller Error			Global/local	Global
Function	Indicates that a Communications Controller failure occurred on the communications port 2. TRUE: Failure FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_EtnCfgErr				
Meaning	Basic Ethernet Setting Error			Global/local	Global
Function	NX-series CPU Units: Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 1 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal NJ-series CPU Units: Indicates that the Ethernet communications speed setting (Speed/Duplex) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_EtnCfgErr				
Meaning	Port1 Basic Ethernet Setting Error			Global/local	Global
Function	Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 1 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_EtnCfgErr				
Meaning	Port2 Basic Ethernet Setting Error			Global/local	Global
Function	Indicates that the Ethernet communications speed setting (Speed/Duplex) for the communications port 2 is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_IPAdrCfgErr				
Meaning	IP Address Setting Error			Global/local	Global
Function	NX-series CPU Units: Indicates the IP address setting errors for the communications port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The IP address obtained from the DHCP server is inconsistent. FALSE: Normal NJ-series CPU Units: Indicates the IP address setting errors. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The default gateway settings are not correct. FALSE: Normal				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_IPAdrCfgErr				
Meaning	Port1 IP Address Setting Error			Global/local	Global
Function	Indicates the IP address setting errors for the communications port 1. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The IP address obtained from the DHCP server is inconsistent. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Related instructions	Related instructions	---		

Variable name	_EIP2_IPAdrCfgErr				
Meaning	Port2 IP Address Setting Error			Global/local	Global
Function	Indicates the IP address setting errors for the communications port 2. TRUE: • There is an illegal IP address setting. • A read operation failed. • The IP address obtained from the BOOTP server is inconsistent. • The IP address obtained from the DHCP server is inconsistent. FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_IPAdrDupErr				
Meaning	IP Address Duplication Error			Global/local	Global
Function	NX-series CPU Units: Indicates that the same IP address is assigned to more than one node for the communications port 1. TRUE: Duplication occurred. FALSE: Other than the above. NJ-series CPU Units: Indicates that the same IP address is assigned to more than one node. TRUE: Duplication occurred. FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_IPAdrDupErr				
Meaning	Port1 IP Address Duplication Error			Global/local	Global
Function	Indicates that the same IP address is assigned to more than one node for the communications port 1. TRUE: Duplication occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_IPAdrDupErr				
Meaning	Port2 IP Address Duplication Error			Global/local	Global
Function	Indicates that the same IP address is assigned to more than one node for the communications port 2. TRUE: Duplication occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_DNSCfgErr*1				
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	---		

*1. With the NJ-series CPU Unit, this variable can be used with the unit version 1.11 or later.

Variable name	_EIP_BootpErr				
Meaning	BOOTP Server Error			Global/local	Global
Function	NX-series CPU Units: Indicates that a BOOTP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. NJ-series CPU Units: Indicates that a BOOTP server connection failure occurred. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_BootpErr				
Meaning	Port1 BOOTP Server Error			Global/local	Global
Function	Indicates that a BOOTP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_BootpErr				
Meaning	Port2 BOOTP Server Error			Global/local	Global
Function	Indicates that a BOOTP server connection failure occurred on the communications port 2. TRUE: There was a failure to connect to the BOOTP server (timeout). FALSE: The BOOTP is not enabled, or BOOTP is enabled and an IP address was normally obtained from the BOOTP server. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_DhcpErr				
Meaning	DHCP Server Error			Global/local	Global
Function	Indicates that a DHCP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the DHCP server (timeout). FALSE: The DHCP is not enabled, or DHCP is enabled and an IP address was normally obtained from the DHCP server. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_DhcpErr				
Meaning	Port1 DHCP Server Error			Global/local	Global
Function	Indicates that a DHCP server connection failure occurred on the communications port 1. TRUE: There was a failure to connect to the DHCP server (timeout). FALSE: The DHCP is not enabled, or DHCP is enabled and an IP address was normally obtained from the DHCP server. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_DhcpErr				
Meaning	Port2 DHCP Server Error			Global/local	Global
Function	Indicates that a DHCP server connection failure occurred on the communications port 2. TRUE: There was a failure to connect to the DHCP server (timeout). FALSE: The DHCP is not enabled, or DHCP is enabled and an IP address was normally obtained from the DHCP server. **Note** You can use this system-defined variable only for the NX502 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_IPRTblErr				
Meaning	IP Route Table Error			Global/local	Global
Function	NX-series CPU Units: Indicates that the default gateway settings or IP router table settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal NJ-series CPU Units: Indicates that the IP router table or hosts settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal				
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	NX-series CPU Units: Indicates that the identity information for CIP communications 1 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal NJ-series CPU Units: Indicates that the identity information (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal				
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_IdentityErr				
Meaning	CIP Communications1 Identity Error			Global/local	Global
Function	Indicates that the identity information for CIP communications 1 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_IdentityErr				
Meaning	CIP Communications2 Identity Error			Global/local	Global
Function	Indicates that the identity information for CIP communications 2 (which you cannot overwrite) is incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkCfgErr				
Meaning	Tag Data Link Setting Error			Global/local	Global
Function	NX-series CPU Units: Indicates that the tag data link settings for CIP communications 1 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal NJ-series CPU Units: Indicates that the tag data link settings are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkCfgErr				
Meaning	CIP Communications1 Tag Data Link Setting Error			Global/local	Global
Function	Indicates that the tag data link settings for CIP communications 1 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkCfgErr				
Meaning	CIP Communications2 Tag Data Link Setting Error			Global/local	Global
Function	Indicates that the tag data link setting for CIP communications 2 are incorrect. Or, a read operation failed. TRUE: Setting incorrect or read failed. FALSE: Normal **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkOpnErr				
Meaning	Tag Data Link Connection Failed			Global/local	Global
Function	NX-series CPU Units: Indicates that establishing a tag data link connection for CIP communications 1 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above. NJ-series CPU Units: Indicates that establishing a tag data link connection failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkOpnErr				
Meaning	CIP Communications1 Tag Data Link Connection Failed			Global/local	Global
Function	Indicates that establishing a tag data link connection for CIP communications 1 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkOpnErr				
Meaning	CIP Communications2 Tag Data Link Connection Failed			Global/local	Global
Function	Indicates that establishing a tag data link connection for CIP communications 2 failed. TRUE: Establishing a tag data link connection failed due to one of the following causes. • The information registered for a target node in the tag data link parameters is different from the actual node information. • There was no response from the remote node. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkErr				
Meaning	Tag Data Link Communications Error			Global/local	Global
Function	NX-series CPU Units: Indicates that a timeout occurred in a tag data link connection for CIP communications 1. TRUE: A timeout occurred. FALSE: Other than the above. NJ-series CPU Units: Indicates that a timeout occurred in a tag data link connection. TRUE: A timeout occurred. FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkErr				
Meaning	CIP Communications1 Tag Data Link Communications Error			Global/local	Global
Function	Indicates that a timeout occurred in a tag data link connection for CIP communications 1. TRUE: A timeout occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkErr				
Meaning	CIP Communications2 Tag Data Link Communications Error			Global/local	Global
Function	Indicates that a timeout occurred in a tag data link connection for CIP communications 2. TRUE: A timeout occurred. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TagAdrErr				
Meaning	Tag Name Resolution Error			Global/local	Global
Function	NX-series CPU Units: Indicates that the tag resolution for CIP communications 1 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. NJ-series CPU Units: Indicates that tag name resolution failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above.				
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_TagAdrErr				
Meaning	CIP Communications1 Tag Name Resolution Error			Global/local	Global
Function	Indicates that the tag resolution for CIP communications 1 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TagAdrErr				
Meaning	CIP Communications2 Tag Name Resolution Error			Global/local	Global
Function	Indicates that the tag resolution for CIP communications 2 failed (i.e., the address could not be identified from the tag name). TRUE: Tag resolution failed (i.e., the address could not be identified from the tag name). The following causes are possible. • The size of the network variable is different from the tag settings. • The I/O direction that is set in the tag data link settings does not agree with the I/O direction of the variable in the CPU Unit. • There is no network variable in the CPU Unit that corresponds to the tag setting. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_MultiSwONErr				
Meaning	Multiple Switches ON Error			Global/local	Global
Function	NX-series CPU Units: Indicates that more than one switch turned ON at the same time in CIP communications 1. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. NJ-series CPU Units: Indicates that more than one switch turned ON at the same time TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_MultiSwONErr				
Meaning	CIP Communications1 Multiple Switches ON Error			Global/local	Global
Function	Indicates that more than one switch turned ON at the same time in CIP communications 1. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_MultiSwONErr				
Meaning	CIP Communications2 Multiple Switches ON Error			Global/local	Global
Function	Indicates that more than one switch turned ON at the same time in CIP communications 2. TRUE: More than one data link start/stop switch changed to TRUE at the same time. FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TcpAppCfgErr				
Meaning	TCP 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	TRUE: The NTP client failed to connect to the server (timeout). FALSE: NTP is not set. Or, NTP is set 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	---		

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	Global
Usage in user program	Possible.	Related instructions	---		

● Functional Classification: EtherNet/IP Communications Status

Variable name	_EIP_EtnOnlineSta				
Meaning	Online			Global/local	Global
Function	NX-series CPU Units: Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 1 (that is, the link is ON, IP address is defined, and there are no errors.) TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. NJ-series CPU Units: Indicates that the built-in EtherNet/IP port's communications can be used via the communications port (that is, the link is ON and IP address is defined, and there are no errors.) TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_EtnOnlineSta				
Meaning	Port1 Online			Global/local	Global
Function	Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 1 (that is, the link is ON, IP address is defined, and there are no errors.) TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_EtnOnlineSta				
Meaning	Port2 Online				Global
Function	Indicates that the built-in EtherNet/IP port's communications can be used via the communications port 2 (that is, the link is ON, IP address is defined, and there are no errors.) TRUE: The built-in EtherNet/IP port's communications can be used. FALSE: The built-in EtherNet/IP port's communications is disabled due to an error in initial processing, restart processing, or link OFF status. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkRunSta				
Meaning	Tag Data Link Communications Status			Global/local	Global
Function	NX-series CPU Units: Indicates that at least one connection is in normal operation in CIP communications 1. TRUE: Normal operation FALSE: Other than the above. NJ-series CPU Units: Indicates that at least one connection is in normal operation. TRUE: Normal operation FALSE: Other than the above.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkRunSta				
Meaning	CIP Communications1 Tag Data Link Communications Status			Global/local	Global
Function	Indicates that at least one connection is in normal operation in CIP communications 1. TRUE: Normal operation FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkRunSta				
Meaning	CIP Communications2 Tag Data Link Communications Status			Global/local	Global
Function	Indicates that at least one connection is in normal operation in CIP communications 2. TRUE: Normal operation FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkAllRunSta				
Meaning	All Tag Data Link Communications Status			Global/local	Global
Function	NX-series CPU Units: Indicates that all tag data links are communicating in CIP communications 1. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. NJ-series CPU Units: Indicates that all tag data links are communicating. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkAllRunSta				
Meaning	CIP Communications1 All Tag Data Link Communications Status			Global/local	Global
Function	Indicates that all tag data links are communicating in CIP communications 1. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkAllRunSta				
Meaning	CIP Communications2 All Tag Data Link Communications Status			Global/local	Global
Function	Indicates that all tag data links are communicating in CIP communications 2. TRUE: Tag data links are communicating in all connections as the originator. FALSE: An error occurred in at least one connection. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_RegTargetSta[255]				
Meaning	Registered Target Node Information			Global/local	Global
Function	NX-series CPU Units: Gives a list of nodes for which EtherNet/IP connections are registered for CIP communications 1. This variable is valid only when the EtherNet/IP port is the originator. Array[x] is TRUE: The connection to the node with a target node ID of x is registered. Array[x] is FALSE: The connection to the node with a target node ID of x is not registered. NJ-series CPU Units: Gives a list of nodes for which EtherNet/IP connections are registered. This variable is valid only when the EtherNet/IP port is the originator. Array[x] is TRUE: The connection to the node with a target node ID of x is registered. Array[x] is FALSE: The connection to the node with a target node ID of x is not registered.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_RegTargetSta[255]				
Meaning	CIP Communications1 Registered Target Node Information			Global/local	Global
Function	Gives a list of nodes for which EtherNet/IP connections are registered for CIP communications 1. This variable is valid only when the EtherNet/IP port is the originator. Array[x] is TRUE: The connection to the node with a target node ID of x is registered. Array[x] is FALSE: The connection to the node with a target node ID of x is not registered. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_RegTargetSta[255]				
Meaning	CIP Communications2 Registered Target Node Information			Global/local	Global
Function	Gives a list of nodes for which EtherNet/IP connections are registered for CIP communications 2. This variable is valid only when the EtherNet/IP port is the originator. Array[x] is TRUE: The connection to the node with a target node ID of x is registered. Array[x] is FALSE: The connection to the node with a target node ID of x is not registered. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_EstbTargetSta[255]				
Meaning	Normal Target Node Information			Global/local	Global
Function	NX-series CPU Units: Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 1. Array[x] is TRUE: The connection to the node with a target node ID of x was established normally. Array[x] is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. NJ-series CPU Units: Gives a list of nodes that have normally established EtherNet/IP connections. Array[x] is TRUE: The connection to the node with a target node ID of x was established normally. Array[x] is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_EstbTargetSta[255]				
Meaning	CIP Communications1 Normal Target Node Information			Global/local	Global
Function	Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 1. Array[x] is TRUE: The connection to the node with a target node ID of x was established normally. Array[x] is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_EstbTargetSta[255]				
Meaning	CIP Communications2 Normal Target Node Information			Global/local	Global
Function	Gives a list of nodes that have normally established EtherNet/IP connections for CIP communications 2. Array[x] is TRUE: The connection to the node with a target node ID of x was established normally. Array[x] is FALSE: The connection to the node with a target node ID of x was not established, or an error occurred. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TargetPLCModeSta[255]				
Meaning	Target PLC Operating Mode			Global/local	Global
Function	NX-series CPU Units: Shows the operating status of the target node Controllers that are connected for CIP communications 1, with the 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, it indicates the previous operating status. Array[x] is TRUE: This is the operating state of the target Controller with a node address of x. Array[x] is FALSE: Other than the above. NJ-series CPU Units: Shows the operating status of the target node Controllers that are connected with the 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, it indicates the previous operating status. Array[x] is TRUE: This is the operating state of the target Controller with a node address of x. Array[x] is FALSE: Other than the above.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TargetPLCModeSta[255]				
Meaning	CIP Communications1 Target PLC Operating Mode			Global/local	Global
Function	Shows the operating status of the target node Controllers that are connected for CIP communications 1, with the 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, it indicates the previous operating status. Array[x] is TRUE: This is the operating state of the target Controller with a node address of x. Array[x] is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TargetPLCModeSta[255]				
Meaning	CIP Communications2 Target PLC Operating Mode			Global/local	Global
Function	Shows the operating status of the target node Controllers that are connected for CIP communications 2, with the 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, it indicates the previous operating status. Array[x] is TRUE: This is the operating state of the target Controller with a node address of x. Array[x] is FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TargetPLCErr[255]				
Meaning	Target PLC Error Information			Global/local	Global
Function	NX-series CPU Units: Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 1, with the EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. Array[x] is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. Array[x] is FALSE: Other than the above. NJ-series CPU Units: Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected with the EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. Array[x] is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. Array[x] is FALSE: Other than the above.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TargetPLCErr[255]				
Meaning	CIP Communications1 Target PLC Error Information			Global/local	Global
Function	Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 1, with the EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. Array[x] is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. Array[x] is FALSE: Other than the above. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TargetPLCErr[255]				
Meaning	CIP Communications2 Target PLC Error Information			Global/local	Global
Function	Shows the error status (logical OR of fatal and non-fatal errors) of the target node Controllers that are connected for CIP communications 2, with the EtherNet/IP ports as the originator. The array elements are valid only when the corresponding Normal Target Node Information is TRUE. The immediately preceding value is retained if this variable is FALSE. Array[x] is TRUE: A fatal or non-fatal error occurred in the target Controller with a target node ID of x. Array[x] is FALSE: Other than the above. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TargetNodeErr[255]				
Meaning	Target Node Error Information			Global/local	Global
Function	NX-series CPU Units: Indicates that the connection for the Registered Target Node Information for CIP communications 1 was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE). NJ-series CPU Units: Indicates that the connection for the Registered Target Node Information was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE).				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TargetNodeErr[255]				
Meaning	CIP Communications1 Target Node Error Information			Global/local	Global
Function	Indicates that the connection for the Registered Target Node Information for CIP communications 1 was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE). **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TargetNodeErr[255]				
Meaning	CIP Communications2 Target Node Error Information			Global/local	Global
Function	Indicates that the connection for the Registered Target Node Information for CIP communications 2 was not established or that an error occurred in the target Controller. The array elements are valid only when the Registered Target Node Information is TRUE. Array[x] is TRUE: A connection was not normally established with the target node for a target node ID of x (the Registered Target Node Information is TRUE and the Normal Target Node Information is FALSE), or a connection was established with the target node but an error occurred in the target Controller. Array[x] is FALSE: The target node is not registered for a target node ID of x (the Registered Target Node Information is FALSE), or a connection was normally established with the target node (the Registered Target Node Information is TRUE and the Normal Target Node Information is TRUE). An error occurred in the target Controller (the Target PLC Error Information is TRUE). **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	ARRAY [0..255] OF BOOL			Range of values	TRUE or FALSE
R/W access	R	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_NTPResult				
Meaning	NTP Operation Information			Global/local	Global
Function	Use the GetNTPStatus instruction to read the NTP operation information from the user program. Direct access is not possible.				
Data type	_sNTP_RESULT			Range of values	---
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	.ExecTime
Meaning	NTP Last Operation Time			Global/local	Global
Function	Gives the last time that NTP processing ended normally. The time that was obtained from the NTP server is stored when the time is obtained normally. The time is not stored if it is not obtained from the NTP server normally. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device.				
Data type	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	This variable shows if the NTP operation ended normally. TRUE: Indicates an NTP normal end. FALSE:Indicates that NTP operation ended in an error or has not been executed even once. **Note** Do not use this variable in the user program. There may be a delay in updating it. Use this variable only to access status through communications from an external device.				
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	NX-series CPU Units: Change this variable to TRUE to start tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation starts. NJ-series CPU Units: Change this variable to TRUE to start tag data links. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkStartCmd				
Meaning	CIP Communications1 Tag Data Link Communications Start Switch			Global/local	Global
Function	Change this variable to TRUE to start tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkStartCmd				
Meaning	CIP Communications2 Tag Data Link Communications Start Switch			Global/local	Global
Function	Change this variable to TRUE to start tag data links for CIP communications 2. It automatically changes back to FALSE after tag data link operation starts. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP_TDLinkStopCmd				
Meaning	Tag Data Link Communications Stop Switch			Global/local	Global
Function	NX-series CPU Units: Change this variable to TRUE to stop tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation stops. NJ-series CPU Units: Change this variable to TRUE to stop tag data links. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP1_TDLinkStopCmd				
Meaning	CIP Communications1 Tag Data Link Communications Stop Switch			Global/local	Global
Function	Change this variable to TRUE to stop tag data links for CIP communications 1. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for NX-series CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	_EIP2_TDLinkStopCmd				
Meaning	CIP Communications2 Tag Data Link Communications Stop Switch			Global/local	Global
Function	Change this variable to TRUE to stop tag data links for CIP communications 2. It automatically changes back to FALSE after tag data link operation stops. **Note** Do not force this switch to change to FALSE from the user program or from the Sysmac Studio. It changes to FALSE automatically. **Note** You can use this system-defined variable only for the NX701 CPU Units, NX502 CPU Units, and NX102 CPU Units.				
Data type	BOOL			Range of values	TRUE or FALSE
R/W access	RW	Retained	Not retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

A-9 Attributes of CPU Unit Data

The following table shows the attributes of the CPU Unit data including the Retain/Non-retain attribute in the following cases: power interruption, power on, operating mode change, and major fault level Controller error.

CPU Unit data			Data retention at power interruptions ○: Retained. ×: Not-retained.	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 ○: Allowed. ×: Not allowed.
					Changing between PROGRAM and RUN Mode	When a major fault level Controller error occurs		Synchronized data ○: Applicable. ×: Not applicable.		
User program	POUs and user program execution ID in user program		○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM/ RUN mode (online editing)	○ Online editing
Task Settings	Task Settings		○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
Variables	Variable tables (not variable values)	Device variable	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	x
		User-defined variables	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM/ RUN mode (online editing)	○ Online editing
Data type		User-defined data types	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM/ RUN mode (online editing)	○ Online editing
Controller name		CPU Unit name	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM/ RUN mode	○
		Built-in EtherNet/IP port name	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Supported.	×	PROGRAM/ RUN mode	○

CPU Unit data			Data retention at power interruptions ○: Retained. ×: Not-retained.	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 ○: Allowed. ×: Not allowed.
					Changing between PROGRAM and RUN Mode	When a major fault level Controller error occurs		Synchronized data ○: Applicable. ×: Not applicable.		
Controller Setup	Operation Settings	Operation Settings Error Settings	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	CPU Unit name: RUN/PROGRAM mode, Other settings: PROGRAM mode	×
	Security Settings	Protection Settings at Startup	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	Write Protection and other settings: PROGRAM mode	○
	Built-in EtherNet/IP Port Settings	TCP/IP Settings LINK Settings SNMP Settings SNMP Trap Settings, NTP Settings FTP Settings CIP Settings*1	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
		Tag data link settings for built-in EtherNet/IP port	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Supported.	×	PROGRAM/RUN mode	×
	FINS Settings	Node Address Settings, FINS/UDP Settings, FINS/TCP Settings, FINS Routing Tables, FINS Write Protection	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×

CPU Unit data			Data retention at power interruptions ○: Retained. ×: Not-retained.	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 ○: Allowed. ×: Not allowed.
					Changing between PROGRAM and RUN Mode	When a major fault level Controller error occurs		Synchronized data ○: Applicable. ×: Not applicable.		
	Built-in I/O Settings		○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
	Option Board Settings		○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
	Memory Settings for CJ-series Units		○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
OPC UA Settings	OPC UA Server Settings	Server Settings Network Settings End Point Settings Execution Log Settings	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
	Client Certification Settings	Certification Trust List Certification Trust List (Expired) Certification Reject List Root Certification/Intermediate Certification List Root Certification/Intermediate Certification List (Expired)	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Supported.	×	PROGRAM/ RUN mode	○
	User Settings	User Authentication Definition File	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Supported.	×	PROGRAM/ RUN mode	○
Motion Control Setup	Axis assignments, axis parameter settings, axes group parameter settings, MC common parameter settings		○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×

CPU Unit data			Data retention at power interruptions ○: Retained. ×: Not-retained.	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 ○: Allowed. ×: Not allowed.
					Changing between PROGRAM and RUN Mode	When a major fault level Controller error occurs		Synchronized data ○: Applicable. ×: Not applicable.		
Cam Data			○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
Event Setting Table	**Event Setting Table**	**User-defined error messages**	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	RUN/ PROGRAM mode	×
Bus configuration	**NX Units**		○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
	X Bus Units		○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
	CJ-series bus configuration	**I/O table**	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	○	PROGRAM mode	×
Special I/O Unit Settings/ CPU Bus Unit Settings	**CJ-series Unit Settings**	**Data in CJ-series Units, such as protocol macros**	○ (Depends on CJ-series Unit)	---	Retained.	Retained.	Supported.	×	Depends on the Unit.	
		Words allocated to CPU Bus Units, Example: Controller Link Data Link Tables.	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained.	Supported.	×	RUN/ PROGRAM mode	○
		Words allocated in DM Area	○ (with Battery)	Same as before power interruption.	Retained.	Retained.	Supported.	○	RUN/ PROGRAM mode	○
EtherCAT Configuration	**EtherCAT Network Configuration**	**Network configuration information**	○ (with non-volatile memory)	Same as before power interruption.	Retained.	Retained	Not supported.	○	PROGRAM mode	×

CPU Unit data			Data retention at power interruptions ○: Retained. ×: Not-retained.	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 ○: Allowed. ×: Not allowed.
					Changing between PROGRAM and RUN Mode	When a major fault level Controller error occurs		Synchronized data ○: Applicable. ×: Not applicable.		
EtherCAT Settings	EtherCAT Settings	Master	○ (with nonvolatile memory)	Same as before power interruption.	Retained.	Retained	Not supported.	○	PROGRAM mode	×
		Settings in Slaves	○ (by slaves)	---	Retained.	Retained.	Supported.	○	RUN/ PROGRAM mode	○
Operation Authority Verification			○ (with nonvolatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	×	RUN/ PROGRAM mode	○
User Authentication*1			○ (with nonvolatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	×	RUN/ PROGRAM mode	○
User program execution ID in CPU Unit			○ (with nonvolatile memory)	Same as before power interruption.	Retained.	Retained.	Not supported.	×	PROGRAM mode	×
Values of Variables	Values of non-retained variables	User-defined variables and device variables	×	Initial values	Initial values	Initial values	Supported.	×	RUN/ PROGRAM mode	○
	Values of retained variables	User-defined variables and device variables	○ (with Battery*2)	Same as before power interruption.	Retained.	Retained.	Supported.	×	RUN/ PROGRAM mode	○
Contents of memory used for CJ-series Units	CIO/WR		×	16#0000	16#0000	16#0000	Supported.	×	RUN/ PROGRAM mode	○
	HR/DM/EM		○ (with Battery*2)	Same as before power interruption.	Retained.	Retained.	Supported.	×	RUN/ PROGRAM mode	○
Event logs	Logs	System log User event log	○ (with Battery*2)	Same as before power interruption.	Retained.	Retained.	Supported.	×		○

CPU Unit data		Data retention at power interruptions ○: Retained. ×: Not-retained.	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 ○: Allowed. ×: Not allowed.
				Changing between PROGRAM and RUN Mode	When a major fault level Controller error occurs		Synchronized data ○: Applicable. ×: Not applicable.		
Internal Clock	**Current time of system clock**	○ (with Battery)	With Battery: Retained (continued), Without Battery: Not predictable (may stop).	Retained (continued).	Retained (continued).	Supported.	×	RUN/ PROGRAM mode	○
Absolute encoder home offsets		○ (with Battery*2)	Same as before power interruption.	Retained (continued).	Retained (continued).	Supported.	×		×

*1. CIP settings and user authentication are available in the following CPU Units of the stated unit versions.
- NJ-series, NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

*2. For the NX502 CPU Units, NX102 CPU Units and NX1P2 CPU Units, data is retained in the non-volatile memory.

A-10 Contents of Memory Used for CJ-series Units

You can specify addresses in the memory used for CJ-series Units for AT specifications for variables. Details on each area are provided below.

Precautions for Correct Use

- You can use the memory used for CJ-series Units only with the NJ-series CPU Units, NX502 CPU Units, NX102 CPU Units, and NX1P2 CPU Units.
- Refer to the *NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual (Cat. No. W579)* for how to use memory for CJ-series Units on the NX1P2 CPU Units.

A-10-1 CIO Area

I/O Bits

● Description

The bits in this area are allocated to input and output terminals on CJ-series Basic I/O Units.
The number of words (16 bits each) that is required for each CJ-series Basic I/O Unit are allocated in order based on the position where the Units are connected (from left to right starting from the Unit that is closest to the CPU Unit).
Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses
Range	CIO 0 to CIO 159	CIO 0.00 or CIO 159.15

Additional Information

You can access this area on NJ-series CPU Units through device variables allocated to I/O ports. We therefore recommend that you do not use AT specifications to access this area.
You should use AT specifications for the CIO Area only when you specify addresses for some of the Special Units.

CPU Bus Unit Area

● Description

The bits in this area are allocated to control and status information for CJ-series CPU Bus Units.
Each Unit is allocated 25 words based on its unit number.
Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses	Words per Unit
Range	CIO 1500 to CIO 1899	CIO 1500.00 to CIO 1899.15	25 words

The words that are allocated are listed in the following table.

Word addresses	Unit Number
CIO 1500 to CIO 1524	0
CIO 1525 to CIO 1549	1
to	to
CIO 1875 to CIO 1899	F

For details on how to use the allocated words, refer to the operation manual for the CJ-series CPU Bus Unit.

Additional Information

You can access the CPU Bus Unit Area in NJ-series CPU Units through the device variables that are allocated to I/O ports. We therefore recommend that you do not use AT specifications to access this area.
You should use AT specifications for the CIO Area only when you specify addresses for some of the Special Units.

Special I/O Unit Area

● Description

The bits in this area are allocated to control and status information for CJ-series Special I/O Units. Each Unit is allocated 10 words based on the unit number for up to a total of 96 Units (unit numbers 0 to 95).
Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses	Words per Unit
Range	CIO 2000 to CIO 2959 (10 words × 96 unit numbers)	CIO 2000.00 to CIO 2959.15	10 words

The words that are allocated are listed in the following table.

Word addresses	Unit Number
CIO 2000 to CIO 2009	0
CIO 2010 to CIO 2019	1
to	to
CIO 2950 to CIO 2959	95

For details on how to use the allocated words, refer to the operation manual for the CJ-series Special I/O Unit.

Additional Information

You can access the Special I/O Unit Area in NJ-series CPU Units through the device variables that are allocated to I/O ports. We therefore recommend that you do not use AT specifications to access this area.

DeviceNet Area

Description

The bits in this area are allocated to the slaves when the remote I/O master function of a DeviceNet Unit is used (fixed allocations only).
Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

Addresses

Addresses	Word addresses	Bit addresses
Range	CIO 3200 to CIO 3799	CIO 3200.00 to CIO 3799.15

Words in this area are allocated to slaves for fixed allocations according to fixed allocation setting 1, 2, or 3 in the software switches in the CIO Area. Select one of these fixed areas.

Addresses	Master to slave output area	Slave to master input area
Fixed allocation area 1	CIO 3200 to CIO 3263	CIO 3300 to CIO 3363
Fixed allocation area 2	CIO 3400 to CIO 3463	CIO 3500 to CIO 3563
Fixed allocation area 3	CIO 3600 to CIO 3663	CIO 3700 to CIO 3763

You can allocate memory in the DeviceNet Area even if you use fixed allocations to use the remote I/O slave function of a DeviceNet Unit.

Addresses	Master to slave output area	Slave to master input area
Fixed allocation area 1	CIO 3370	CIO 3270
Fixed allocation area 2	CIO 3570	CIO 3470
Fixed allocation area 3	CIO 3770	CIO 3670

Refer to the *CJ-series DeviceNet Units Operation Manual for NJ-series CPU Unit* (Cat. No. W497) for details.

CIO Area Work Areas

Description

You use the bits in these areas only in programming. You cannot use them to input or output data through external I/O terminals. If you need work bits, you should normally use bits in this area.
Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses
Range	CIO 1300 to CIO 1499 and CIO 3800 to CIO 6143	CIO 1300.00 to CIO 1499.15 and CIO 3800.00 to CIO 6143.15

A-10-2 Internal I/O Area

● Description

You use the bits in these areas only in programming. You cannot use them to input or output data through external I/O terminals. If you need work bits, you should normally use bits in this area.
Data in this area is cleared when power is cycled or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses
Range	W000 to W511	W000.00 to W511.15

A-10-3 Holding Area

● Description

You use the words and bits in this area only in programming. The status of the words and bits in this area are retained during power interruptions or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses
Range	H0 to H511	H0.00 to H511.15

A-10-4 DM Area

● Description

This is a general-purpose data area used to read and write 16-bit words.
You can also add a bit number to address specify bits.
Data in this area is retained during power interruption or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses
Range	D0 to D32767	D0.00 to D32767.15

DM Area Words for Special Units

● Description

The following words in the DM Area are allocated to initial settings for Special Units.

● Addresses

Addresses	Type of CJ-series Special Unit	Word addresses	Words per Unit
Range	CJ-series Special I/O Units	D20000 to D29599 (100 words × 96 unit numbers)	100 words
	CJ-series CPU Bus Units	D30000 to D31599 (100 words × 16 unit numbers)	100 words

The words that are allocated are listed in the following table.

CJ-series Special I/O Units

Word addresses	Unit Number
D20000 to D20099	0
D20100 to D20199	1
to	to
D29500 to D29599	95

CJ-series CPU Bus Units

Word addresses	Unit Number
D30000 to D30099	0
D30100 to D30199	1
to	to
D31500 to D31599	F

For details on how to use the allocated words, refer to the operation manual for the Special Unit.

Additional Information

You can access the DM Area words that are allocated to Special Units in NJ-series CPU Units through the device variables that are allocated to I/O ports. We therefore recommend that you do not use AT specifications to access this area.

A-10-5 EM Area

● Description

This is a general-purpose data area used to read and write 16-bit words.

You can also add a bit number to address specify bits.

Data in this area is retained during power interruption or when the operating mode is changed between PROGRAM and RUN mode.

● Addresses

Addresses	Word addresses	Bit addresses
Range	NJ501-□□□□: E0_0 to E18_32767 NJ301-□□□□: E0_0 to E3_32767 NJ101-□□□□: E0_0 to E3_32767	NJ501-□□□□: E0_0.00 to E18_32767.15 NJ301-□□□□: E0_0.00 to E3_32767.15 NJ101-□□□□: E0_0.00 to E3_32767.15

Note The number of banks is given in hexadecimal.

A-11 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 or union variables are used in the following type of communications with other devices.

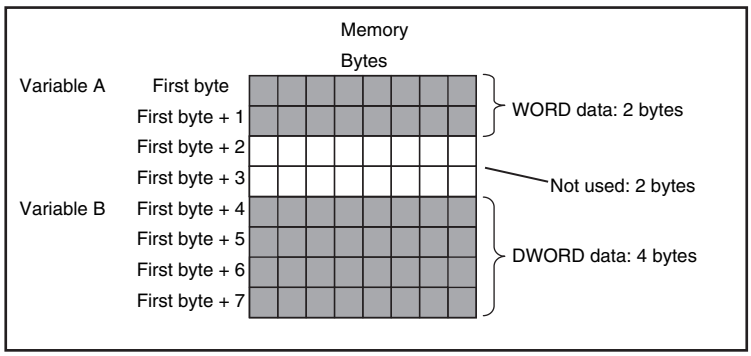
- When using EtherNet/IP tag data links or CIP messages to access variables between NJ/NX-series CPU Units and other CPU Units
- When using structure or union variables to exchange data with devices other than CPU Units, such as ID Tags

A-11-1 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.

Item	Specification
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*1
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

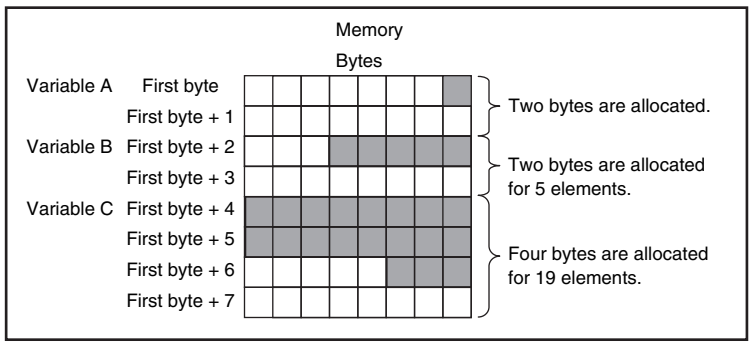
*1. BOOL arrays are an exception. Refer to *Precautions for Correct Use*, below, for the amount of memory that is allocated for BOOL arrays.

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.

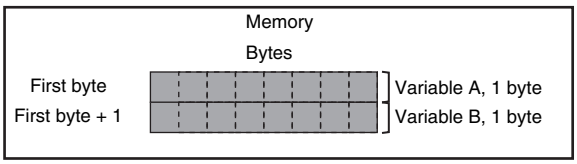
Specific examples of the rules for memory allocation for variables of each data type are given below.

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



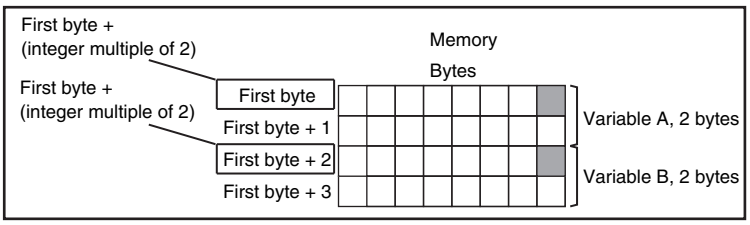
Variable Table

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

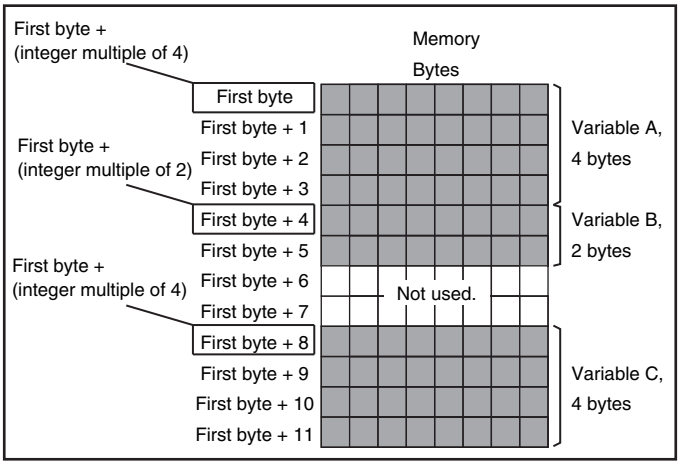
Name	Data type
A	BOOL
B	BOOL

● 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

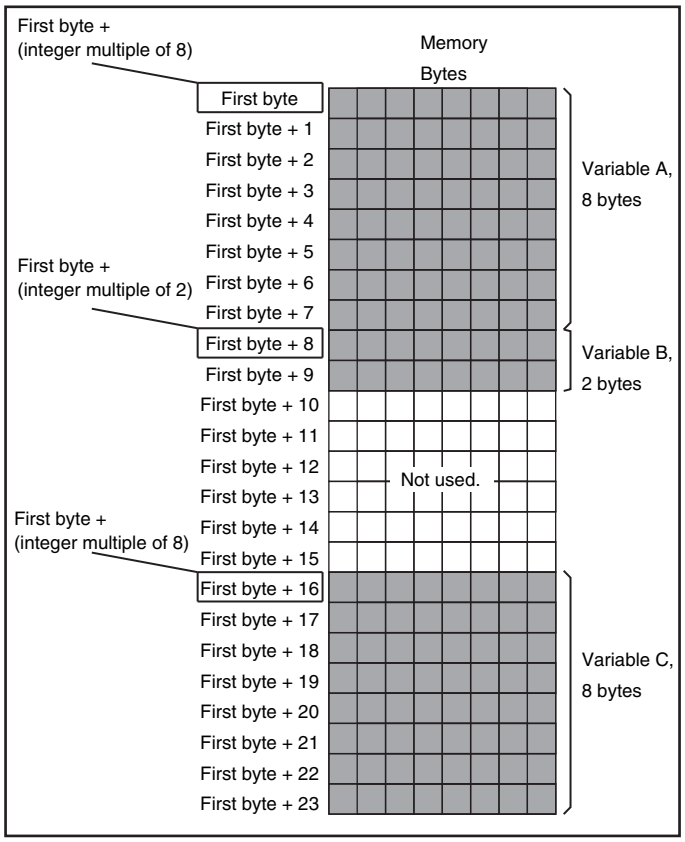
Name	Data type
A	DWORD
B	WORD
C	DWORD

● 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



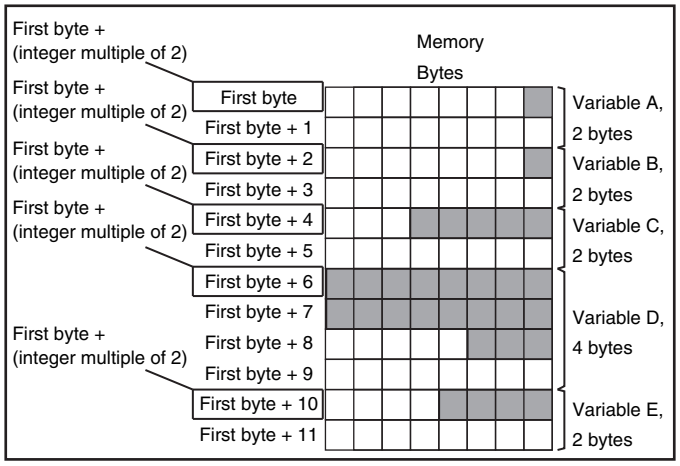
Variable Table

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.

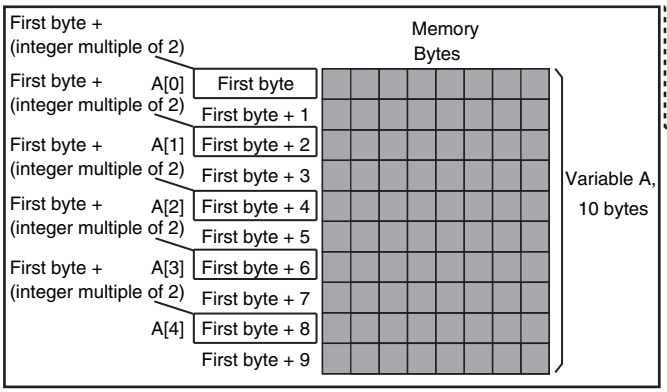
Example: Continuous variables in the following order: two BOOL variable, one BOOL array with five elements, one BOOL array with 19 elements, and one BOOL array with four elements



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

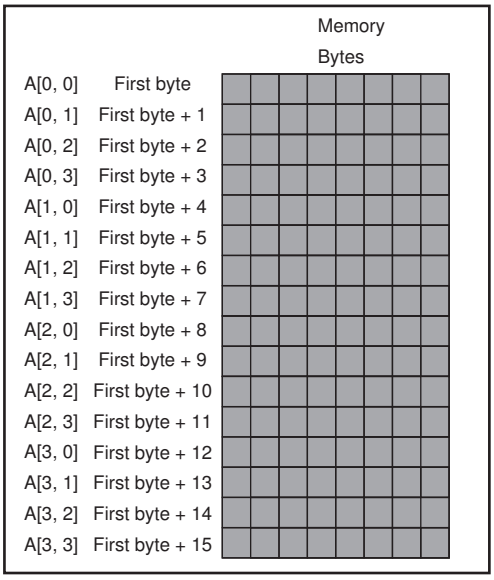
Example: INT array with five elements



Variable Table

Name	Data type
Variable A	ARRAY[0..4] OF INT

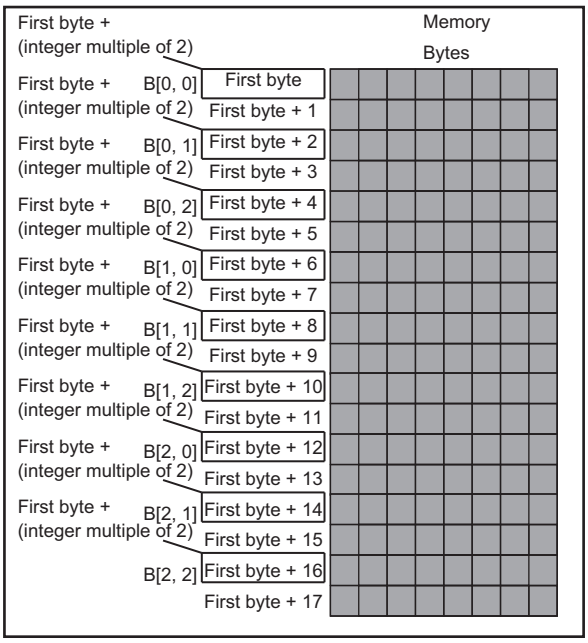
Example: BYTE array with four elements for each dimension with two-dimensional array



Variable Table

Name	Data type
Variable A	ARRAY[0..3, 0..3] OF BYTE

Example: WORD array with three elements for each dimension with two-dimensional array



Variable Table

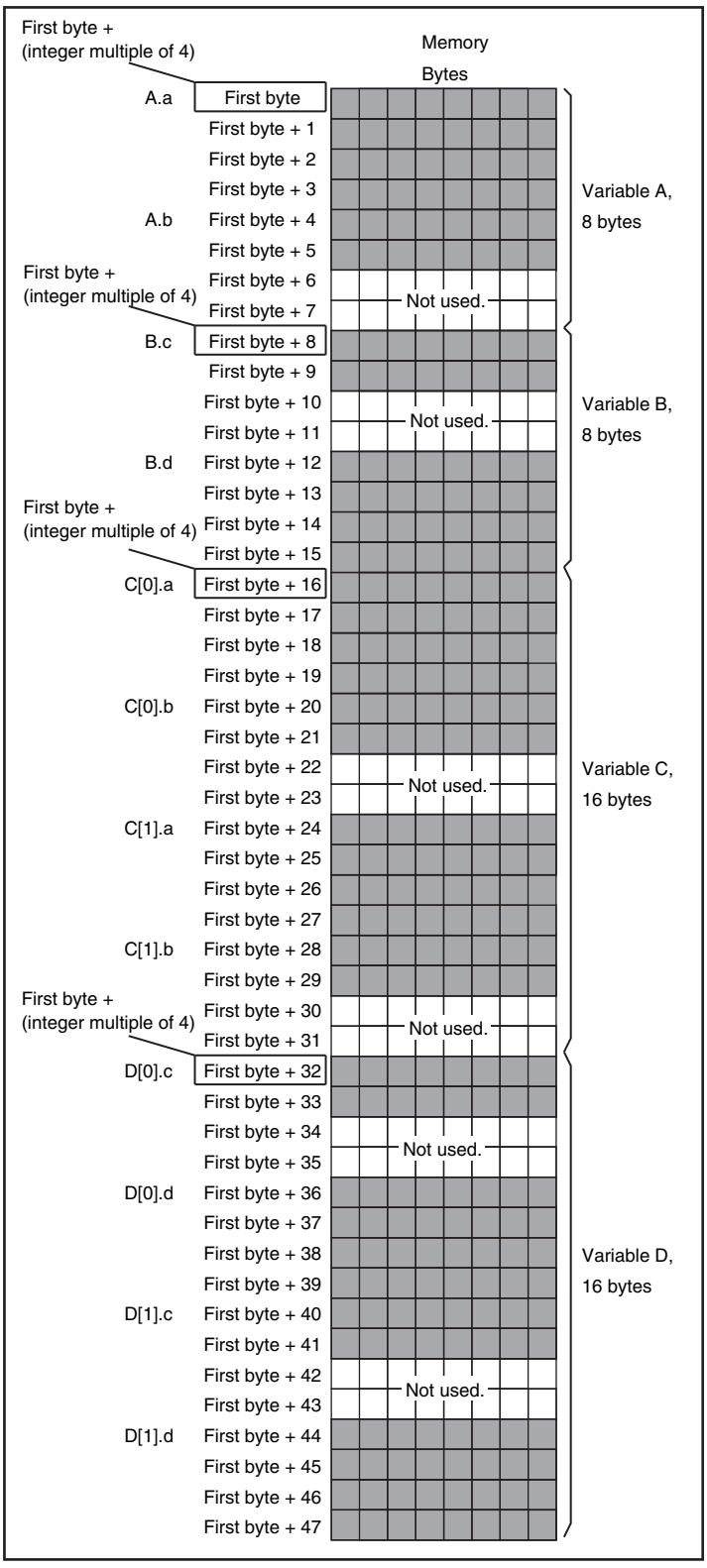
Name	Data type
Variable B	ARRAY[0..2, 0..2] OF WORD

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

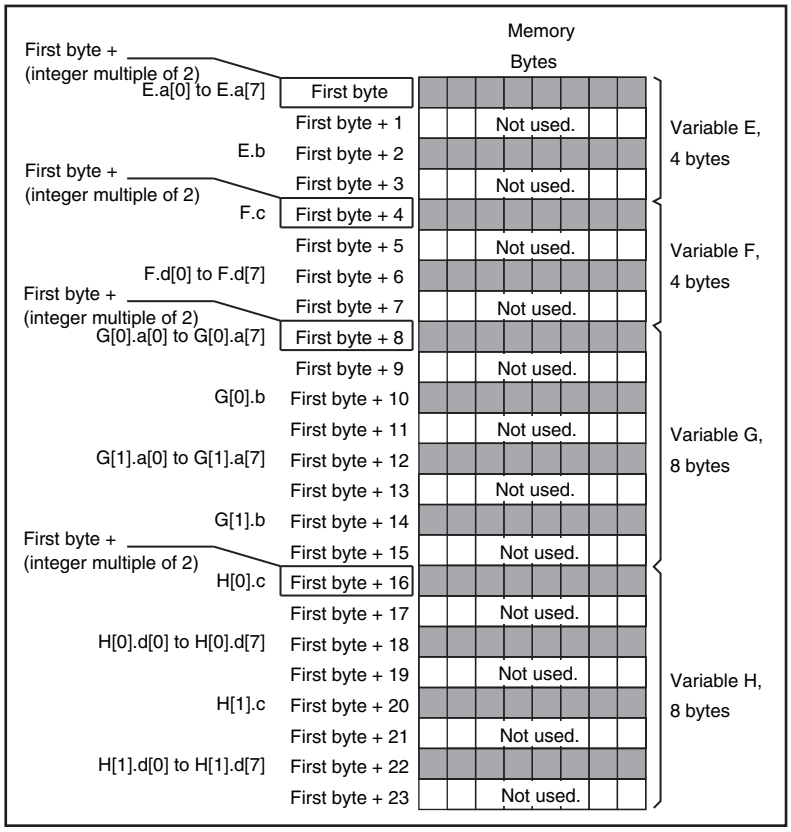
Variable Table

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

Variable	Alignment [bytes]	Amount of memory that is allocated [bytes]
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

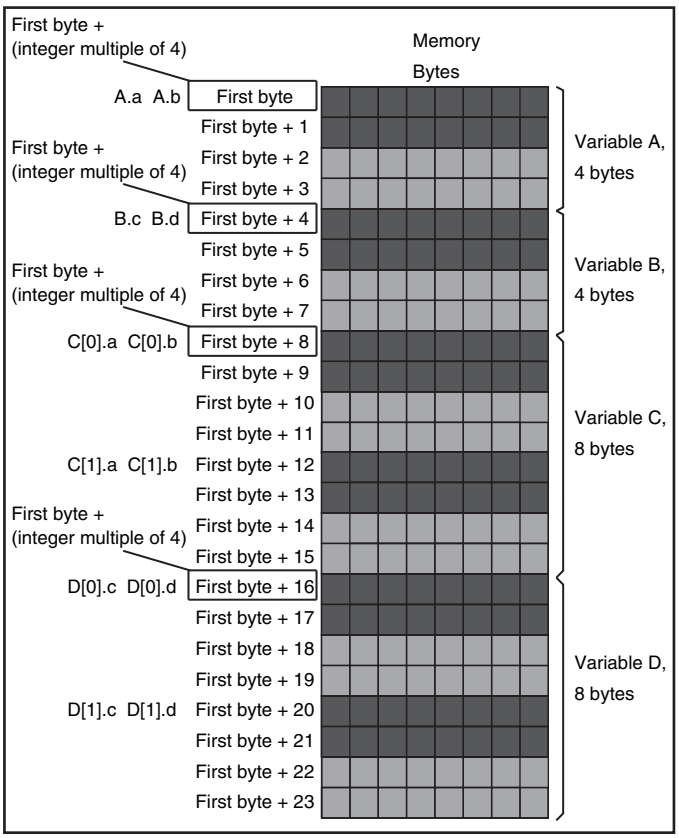
Name	Data type
Variable E	Structure *STR_C*
Variable F	Structure *STR_D*
Variable G	ARRAY[0..1] OF STR_C
Variable H	ARRAY[0..1] OF STR_D

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-11-2 Important Case Examples

When you exchange structure variable data between an NJ/NX-series CPU Unit and a remote device, you must align the memory configuration of the structure variable members with those of the remote device.

This section describes what to do in either the NJ/NX-series CPU Unit or in the remote device.

Additional Information

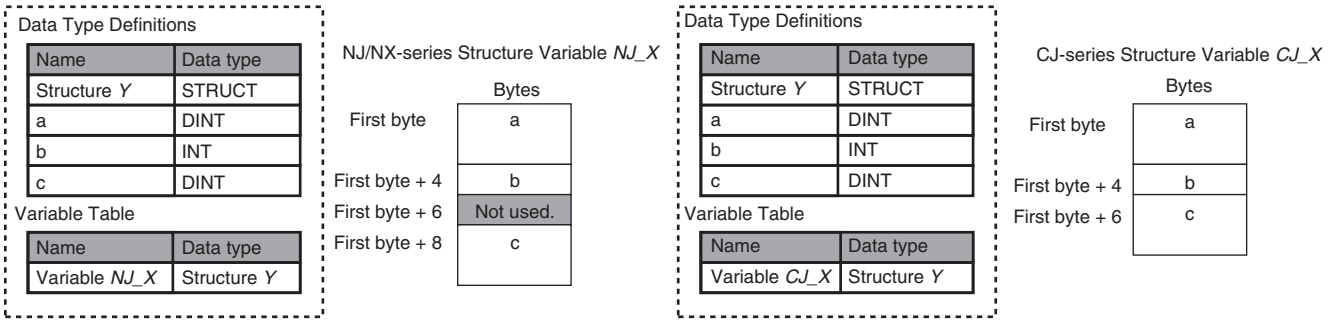
This is not necessary when you exchange data between NJ/NX-series CPU Units.

Aligning the Memory Configuration with a Remote Device

There are two methods that you can use to align the memory configuration with a remote device.

For example, the differences in the memory configuration for structure variables between an NJ/NX-series CPU Unit and a CJ-series CPU Unit are shown below.

This section describes how to align the memory configuration for these Units.



● Method 1: Changing the Memory Configuration of the Structure Variable in the NJ/NX-series CPU Unit

With an NJ/NX-series CPU Unit, you can specify member offsets to change the memory configuration of the members of a structure variable. You can change the memory configuration of the members of a structure variable in the NJ/NX-series CPU Unit so that it is the same as the memory configuration in a remote device that the CPU Unit will communicate with.

Specify the member offsets for a structure variable when you register the structure data type.

To communicate with a CJ-series CPU Unit, you can set the offset type to *CJ* to automatically use the CJ-series memory configuration.

You can set the offset type to *User* to freely set your own offsets.

Version Information

The following table gives the unit version of the CPU Units and the Sysmac Studio version that are required to specify member offsets.

Unit version of CPU Unit	Sysmac Studio version		
	Ver.1.01 or lower	Ver.1.02	Ver.1.03 or higher
Ver.1.01 or later	Not possible.	Possible.*1	Possible.
Ver.1.00	Not possible.	Not possible.	Not possible.

*1. You cannot select the memory offset type. You can set member offsets.

If you change the memory configuration of a structure variable by setting offsets, you must make the same changes for the same structure variable in other NJ/NX-series CPU Units on the network.

Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* 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 NJ/NX-series CPU Unit is changed to match the memory configuration of the structure variable in the CJ-series CPU Unit.

Data Type Definitions

Name	Data type
Structure *Y*	STRUCT
a	DINT
b	INT
c	DINT

Variable Table

Name	Data type
Variable *NJ_X*	Structure *Y*

NJ/NX-series Structure Variable *NJ_X*

	Bytes
First byte	a
First byte + 4	b
First byte + 6	Not used.
First byte + 8	c

Communications is not possible because the memory configuration is not the same.

Data Type Definitions

Name	Data type
Structure *Y*	STRUCT
a	DINT
b	INT
c	DINT

Variable Table

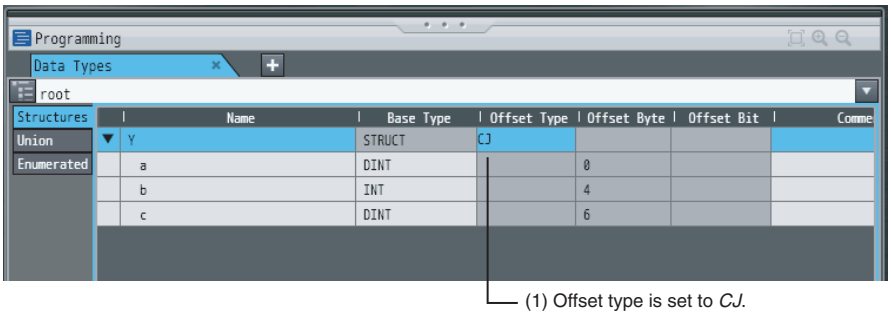
Name	Data type
Variable *CJ_X*	Structure *Y*

CJ-series Structure Variable *CJ_X*

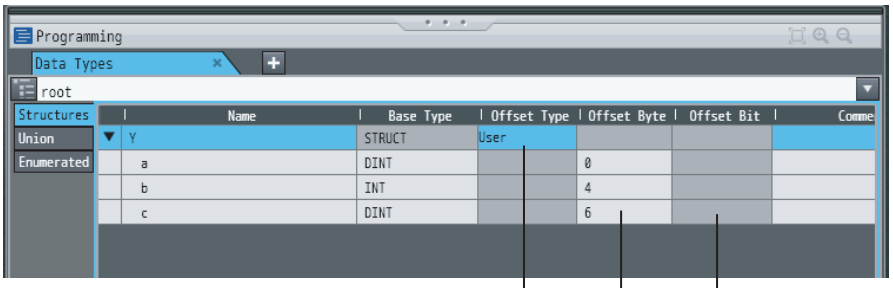
	Bytes
First byte	a
First byte + 4	b
First byte + 6	c

To align the memory configurations in the NJ-series and CJ-series CPU Units, offsets are set in the Sysmac Studio.

Here, the following offsets are set for member c of data type Y of the structure variable NJ_X.



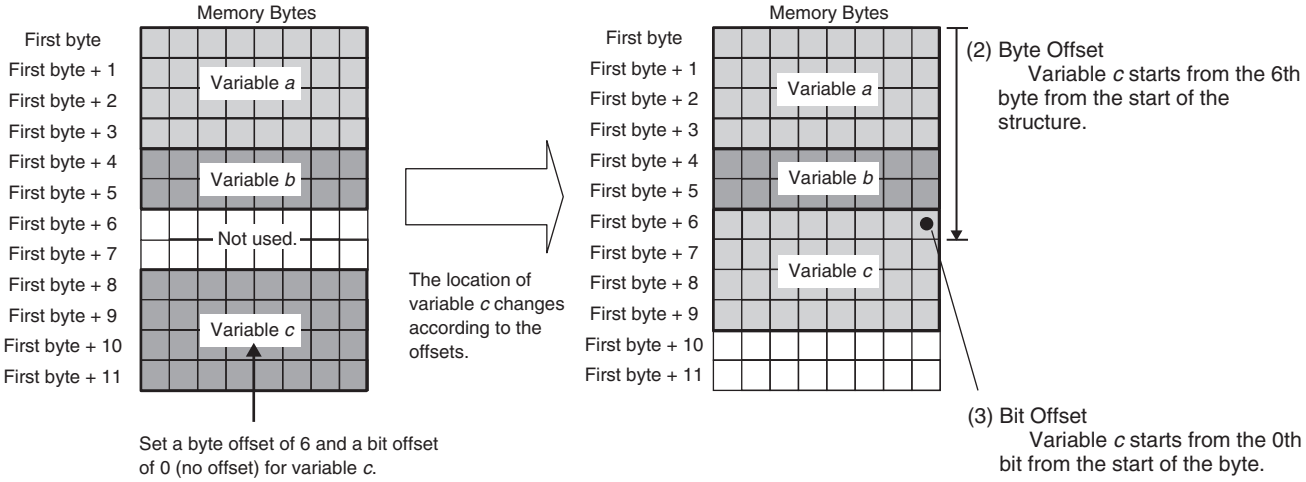
(1) Offset type is set to *CJ*.



(3) Bit Offset
Set the location of the first bit of the member variable.

(2) Byte Offset
Set the location of the first byte of the member from the beginning of the structure variable.

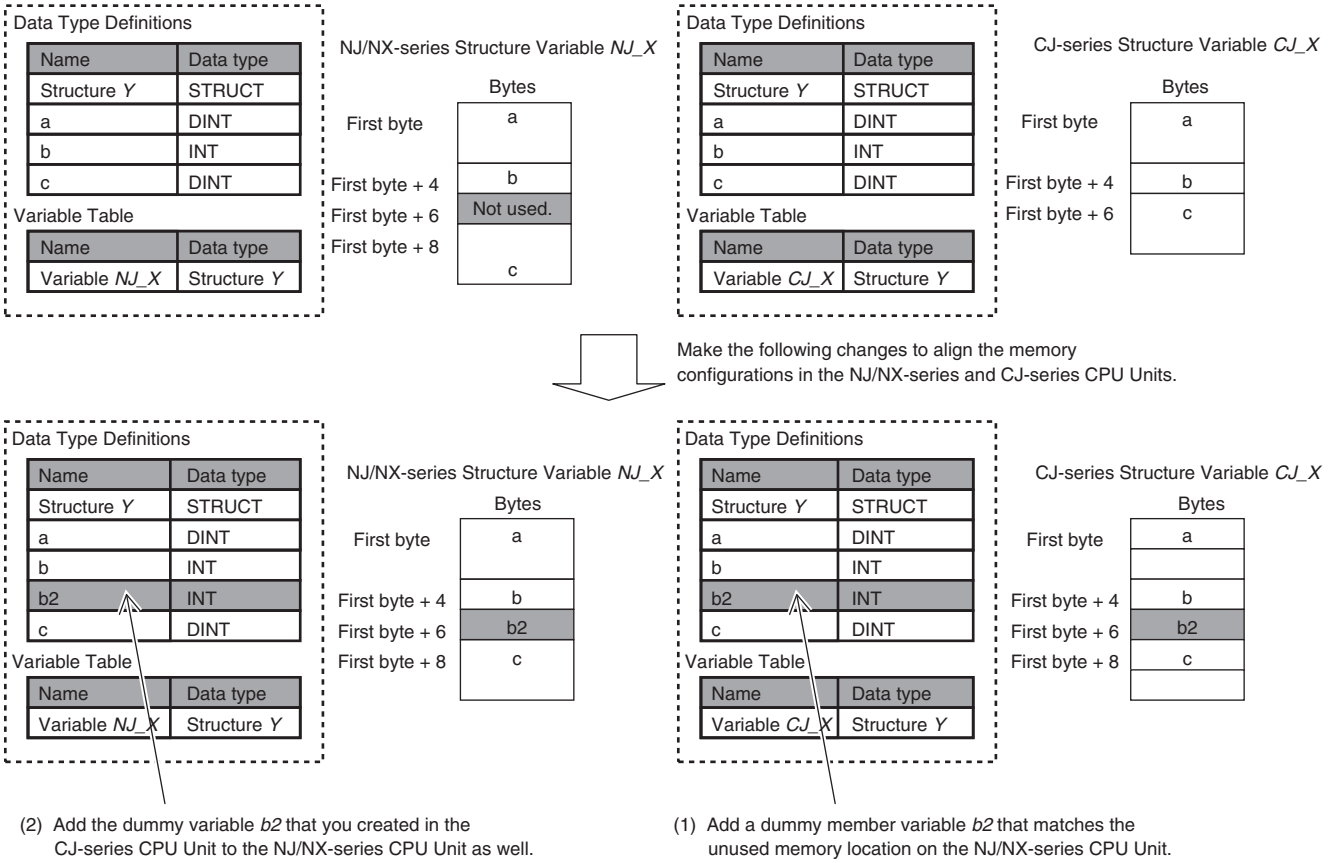
(1) Offset Type
Specify *User*.



● Method 2: Changing the Memory Configuration of the Structure Variable in the Remote Device

You can insert a member into the structure variable of the remote device to change it to match the memory configuration of the structure variable in the NJ/NX-series CPU Unit.
Both the memory configuration and the data types must be the same between the two structure variables. You therefore need to create the same members in both the remote device and the NJ/NX-series CPU Unit.

Example: The following example shows how the memory configuration of the structure variable in the CJ-series CPU Unit is changed to match the memory configuration of the structure variable in the NJ/NX-series CPU Unit.



A-12 Registering a Symbol Table on the CX-Designer

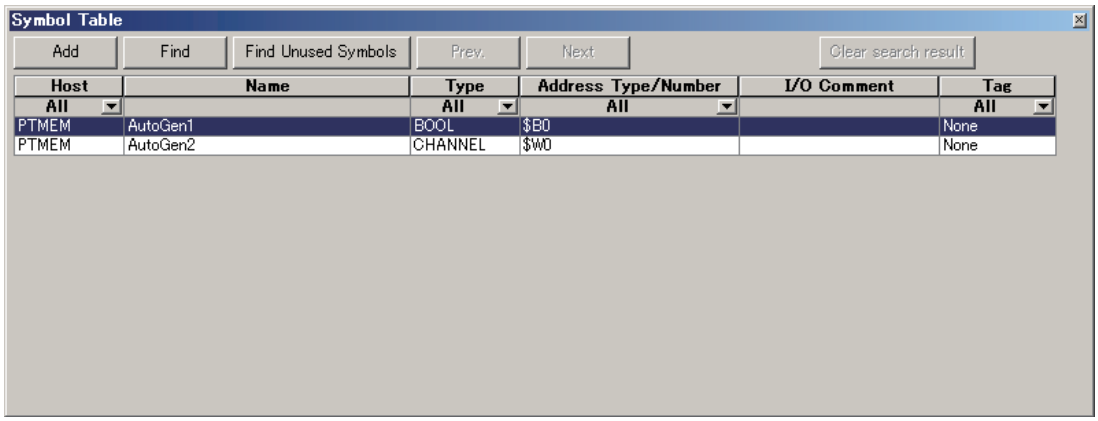
When you connect the NJ/NX-series Controller to an NS-series PT, you can use variables on the CX-Designer to set addresses for the functional objects. The variables are managed in a **Symbol Table**. This section shows how to copy a table of variables from a Microsoft Excel spreadsheet to register them all at the same time in a symbol table. Refer to the *CX-Designer User's Manual* (Cat. No. V099) for detailed information on the CX-Designer.

1 Use the following format to create a table of variables in a Microsoft Excel spreadsheet. You must use the same number and arrangement of columns as in the following format. Do not omit any columns even if they are empty, like the **Address Type/Number** and **I/O Comment** that are shown below.

Host	Name	Type	Address Type/Number	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
HOST3	_Card1PrgTransferCmd.Exec	BOOL			TRUE
HOST3	_Card1PrgTransferCmd.DirName	STRING[64]			TRUE
HOST3	_Card1PrgTransferCmd.Password	STRING[33]			TRUE
HOST3	_Card1PrgTransferCmd.TargetUserProgram	BOOL			TRUE
HOST3	_Card1PrgTransferCmd.TargetIPAdr	BOOL			TRUE
HOST3	_Card1PrgTransferCmd.TargetVariable	BOOL			TRUE
HOST3	_Card1PrgTransferCmd.TargetMemory	BOOL			TRUE
HOST3	_Card1PrgTransferSta.Done	BOOL			TRUE
HOST3	_Card1PrgTransferSta.Active	BOOL			TRUE
HOST3	_Card1PrgTransferSta.Err	BOOL			TRUE
HOST3	_Card1RestoreCmd.Exec	BOOL			TRUE
HOST3	_Card1RestoreCmd.DirName	STRING[64]			TRUE
HOST3	_Card1RestoreCmd.Password	STRING[33]			TRUE
HOST3	_Card1RestoreSta.Done	BOOL			TRUE
HOST3	_Card1RestoreSta.Active	BOOL			TRUE
HOST3	_Card1RestoreSta.Err	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetUserProgram	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetIPAdr	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetVariable	BOOL			TRUE

HOST3	_Card1RestoreCmdTargetMemory	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetUnitConfig	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetAbsEncoder	BOOL			TRUE

2 Start the CX-Designer and open the Symbol Table Dialog Box.

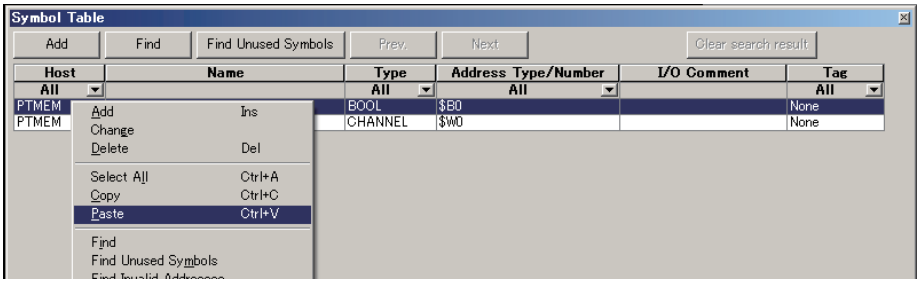


3 Copy the shaded portion of the Microsoft Excel spreadsheet.
Always copy all of the columns that are shown below.

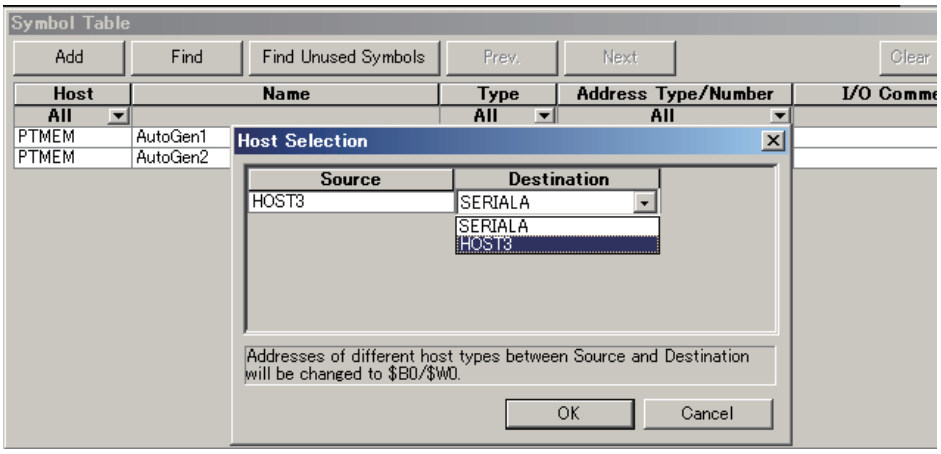
Host	Name	Type	Address Type/Number	I/O Comment	Tag
HOST3	_Card1BkupCmd.ExecBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.ExecVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.DirName	STRING[64]			TRUE
HOST3	_Card1BkupSta.Done	BOOL			TRUE
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
HOST3	_Card1PrgTransferCmd.Exec	BOOL			TRUE
HOST3	_Card1PrgTransferCmd.DirName	STRING[64]			TRUE
HOST3	_Card1PrgTransferCmd.Password	STRING[33]			TRUE
HOST3	_Card1PrgTransferCmd.TargetUserProgram	BOOL			TRUE
HOST3	_Card1PrgTransferCmd.TargetIPAdr	BOOL			TRUE
HOST3	_Card1PrgTransferCmd.TargetVariable	BOOL			TRUE
HOST3	_Card1PrgTransferCmd.TargetMemory	BOOL			TRUE
HOST3	_Card1PrgTransferSta.Done	BOOL			TRUE
HOST3	_Card1PrgTransferSta.Active	BOOL			TRUE
HOST3	_Card1PrgTransferSta.Err	BOOL			TRUE
HOST3	_Card1RestoreCmd.Exec	BOOL			TRUE
HOST3	_Card1RestoreCmd.DirName	STRING[64]			TRUE
HOST3	_Card1RestoreCmd.Password	STRING[33]			TRUE
HOST3	_Card1RestoreSta.Done	BOOL			TRUE

HOST3	_Card1RestoreSta.Active	BOOL			TRUE
HOST3	_Card1RestoreSta.Err	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetUserProgram	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetIPAdr	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetVariable	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetMemory	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetUnitConfig	BOOL			TRUE
HOST3	_Card1RestoreCmdTargetAbsEncoder	BOOL			TRUE

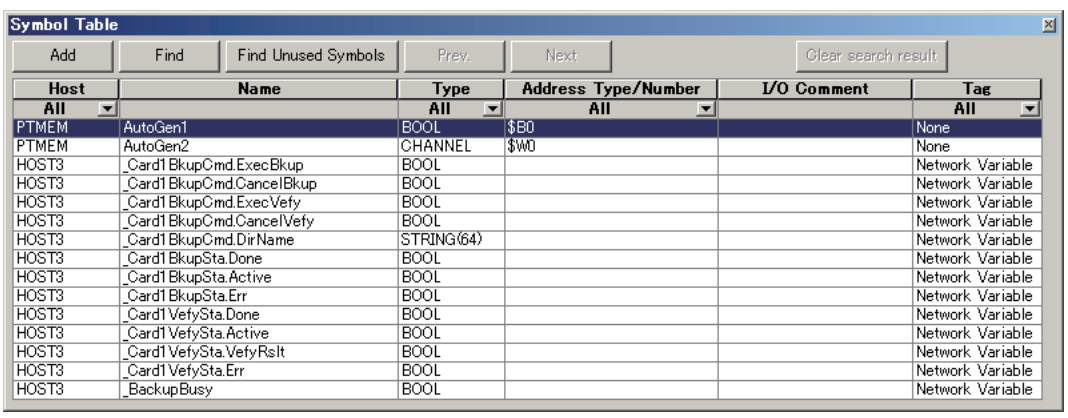
4 Right-click in the Symbol Table Dialog Box in the CX-Designer and select **Paste** from the menu.



5 In the Host Selection Dialog Box on the CX-Designer, select the NJ/NX-series Controller host and then click the **OK** Button.



The variables are registered in the Symbol Table Dialog Box of the CX-Designer.



A-13 Enable/Disable EtherCAT Slave and Axes

You can enable and disable EtherCAT slaves and axes using programming instructions. You can use this for the following types of applications.

- Managing more than one machine with different EtherCAT slave configurations and axis compositions with one project on the Sysmac Studio.
- Leaving one production line running while you change the EtherCAT slave configuration or axis composition of another line.

This section describes the instructions and system-defined variables that are used and provides some application examples.

Version Information

A CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher are required to use the instructions to enable and disable EtherCAT slaves and axes.

A-13-1 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-272.
- You can disable an EtherCAT slave to enable removing it or installing it on the EtherCAT network.

A-13-2 Using Instructions to Enable/Disable EtherCAT Slaves and Axes

You can use instructions in the user program to enable and disable EtherCAT slaves and axes. Separate instructions are used to enable and disable EtherCAT slaves and to enable and disable axes. Both instructions are given in the following table.

Item changed	Instruction
EtherCAT slaves	EC_ChangeEnableSetting (Enable/Disable EtherCAT Slave) instruction
Axes	MC_ChangeAxisUse (Change Axis Use) instruction

Version Information

A CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher are required to use the EC_ChangeEnableSetting and MC_ChangeAxisUse instructions.

EC_ChangeEnableSetting Instruction

The EC_ChangeEnableSetting (Enable/Disable EtherCAT Slave) instruction is used to enable and disable EtherCAT slaves. You can use the EC_ChangeEnableSetting instruction to enable or disable the EtherCAT slave with the specified node address. If you cycle the power supply to the Controller after this instruction is executed, the settings will return to the settings from before instruction execution. Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for the detailed specifications of the EC_ChangeEnableSetting instruction.

MC_ChangeAxisUse Instruction

The MC_ChangeAxisUse (Change Axis Use) instruction is used to enable and disable axes. The MC_ChangeAxisUse instruction changes the setting of the Axis Use axis parameter of the specified axis between *Used Axis* and *Unused Axis*. If you cycle the power supply to the Controller after this instruction is executed, the settings will return to the settings from before instruction execution. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for the detailed specifications of the MC_ChangeAxisUse instruction.

A-13-3 System-defined Variables That Indicate EtherCAT Slave or Axis Status

You can check the values of system-defined variables to get the current status of EtherCAT slaves and axes. The system-defined variables for these are given below.

Accessed status	System-defined variable name
EtherCAT slaves	*_EC_DisableSlavTbl[]* (Disabled Slave Table)
Axes	*_MC_AX[*].Cfg.AxEnable* (Axis Use)*1

*1. You can also use *_MC1_AX[].Cfg.AxEnable* and *_MC2_AX[].Cfg.AxEnable* with the NX701 CPU Unit.

_EC_DisableSlavTbl[] (Disabled Slave Table)

The *_EC_DisableSlavTbl[]* (Disabled Slave Table) system-defined variable tells whether each EtherCAT slave is currently disabled. The node address is specified for the array subscript. The meanings of the values in *_EC_DisableSlavTbl[]* (Disabled Slave Table) are given below.

Value	Meaning
TRUE	The EtherCAT slave with the specified node address is disabled.
FALSE	The EtherCAT slave with the specified node address is enabled.

_MC_AX[].Cfg.AxEnable (Axis Use)

The *_MC_AX[ ].Cfg.AxEnable* (Axis Use) system-defined variable tells whether each axis is defined and whether each axis is used. The axis number is specified for the array subscript. The meanings of the values in *_MC_AX[].Cfg.AxEnable* (Axis Use) are given below.

Value	Meaning
0:_mcNoneAxis	The specified axis is an undefined axis.
1: _mcUnusedAxis	The specified axis is an unused axis.
2: _mcUsedAxis	The specified axis is a used axis.

A-13-4 Enabling/Disabling Execution of Program

There are certain programs associated with the EtherCAT slaves and axes, which are enabled or disabled. These associated programs must be enabled or disabled as the EtherCAT slaves and axes are enabled or disabled. To enable or disable the program, use the following instructions in the user program.

Function	Instruction
Enable program	PrgStart instruction
Disable program	PrgStop instruction

Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for the detailed specifications of the PrgStart instruction and PrgStop instruction.

Precautions for Correct Use

When you want to disable the program, first disable the EtherCAT slave and axis which the program is associated with, and then disable the program.

Version Information

A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the PrgStart and PrgStop instructions.

A-13-5 Checking Enabled/Disabled Program

You can use the PrgStatus instruction to check the program is enabled or disabled that is associated with the EtherCAT slave and axis that are enabled and disabled. Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for the detailed specifications of the PrgStatus instruction.

Version Information

A CPU Unit with unit version 1.08 or later and the Sysmac Studio version 1.09 or higher are required to use the PrgStatus instruction.

A-13-6 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.

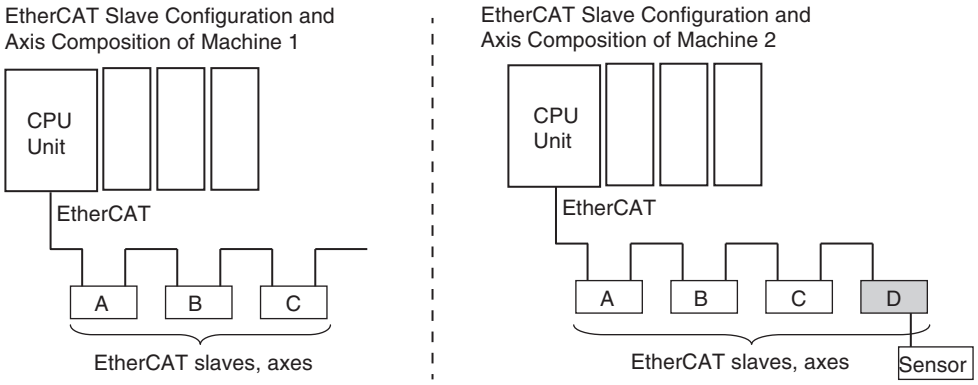
3 Set **Initial Status** of the program to *Run* on the Program Assignment Settings Display.

A-13-7 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 slave configuration and axis composition for the NJ-series Controllers are different for machines 1 and 2 as shown below. These two machines are centrally managed using one Sysmac Studio project.



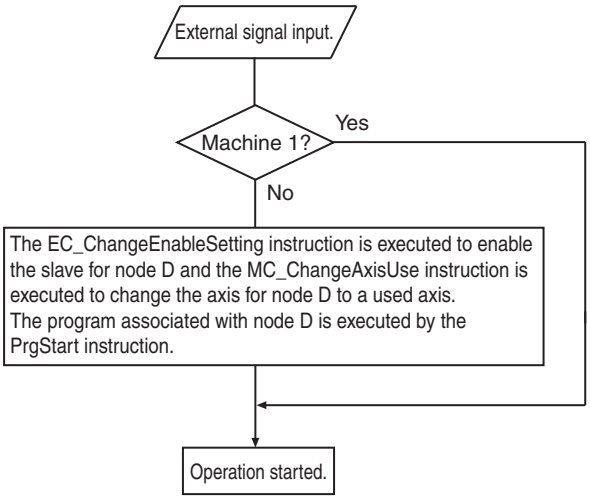
In the Sysmac Studio project, an EtherCAT Slave Configuration is created for all four EtherCAT slaves and axes in A, B, C, and D in the figure. Then, on the Sysmac Studio, you set the EtherCAT slave enable/disable settings, Axis Use parameter settings, and the associated program run/stop status at the start of operation according to machine 1, as shown in the following table.

EtherCAT slave	Installed in EtherCAT network	Enable/ Disable setting	Axis Use parameter setting	Associated programs
A, B, and C	Installed.	Enabled.	Used Axis	Run at the start of operation.
D	Not installed.	Disabled.	Unused Axis	Stop at the start of operation.

To make changes for machine 2, you use instructions to change the EtherCAT slave enable/disable settings, Axis Use parameter settings, and the associated program enable/disable settings as shown in the following table.

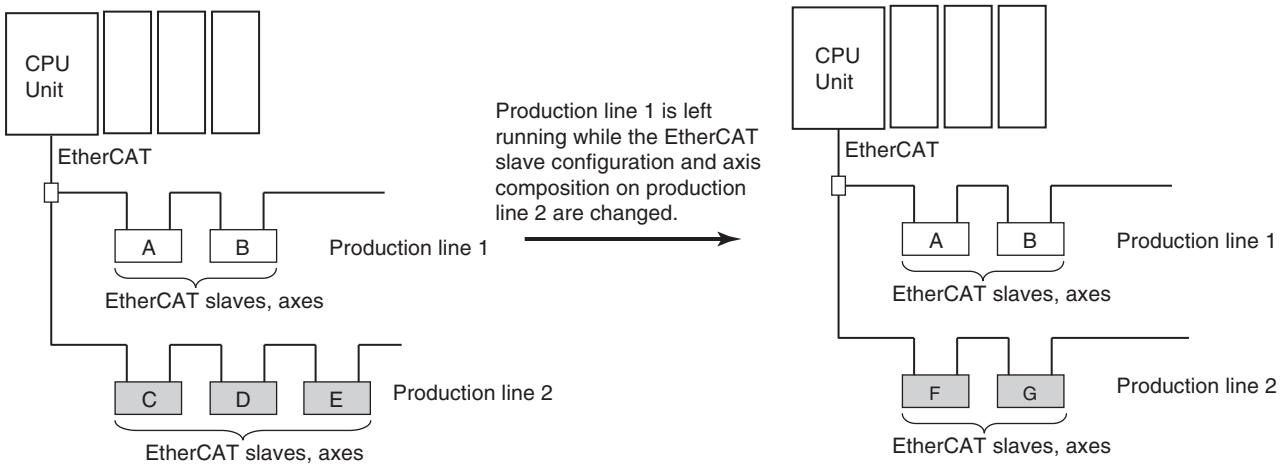
EtherCAT slave	Installed in EtherCAT network	Enable/ Disable setting	Axis Use parameter setting	Associated programs
A, B, and C	Installed.	Enabled.	Used Axis	Enabled.
D	Installed.	Enabled.	Used Axis	Enabled.

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 slave configuration 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.

EtherCAT slave	Installed in EtherCAT network	Enable/ Disable setting	Axis Use parameter setting	Associated programs
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 2.
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 2.
6. Install EtherCAT slaves F and G on production line 2.
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 2 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-14 Size Restrictions for the User Program

There are size restrictions for the user program due to the limitations of the memory capacity in the CPU Unit and other factors. If you exceed these restrictions, errors will occur during operation. This section describes each of the size restrictions of a user program that is created in a CPU Unit.
You can check the approximate sizes of the user program and variables with the memory display functions of the Sysmac Studio.
Be careful not to exceed these restrictions when you create the user program. The restrictions that are given in this section, however, are only reference values for use as guidelines. We recommend that you ensure ample leeway for the restrictions to allow for the possibility of future user program expansion as well as for other reasons.

Precautions for Correct Use

Errors can occur during online editing even if the user program size restrictions are not exceeded. This is because even if you change the user program with online editing, other data that is allocated in the memory of the CPU Unit may remain. If errors occur, change the Controller to PROGRAM mode and transfer the user program to the Controller again to reset the errors.

A-14-1 User Program Object Restrictions

This section describes the restrictions to user program objects. There are restrictions for the following objects.

- POU
- Variables
- Data type definitions
- Constants (literals)

POU Restrictions

There are restrictions both on POU definitions and POU instances.

● POU Definition Restrictions

POU definitions are subject to the following restrictions.

Restriction	CPU Unit model						
	NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum number of programs	1,000	1,000	1,000	500	500	500	500
Maximum value of the following: Number of function block definitions + Number of function definitions + Number of ladder diagram sections	6,000	6,000	3,000	450	3,000	750	450

Restriction	CPU Unit model						
	NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum total number of input, output, and in-out variables in function block and function definitions	64	64	64	64	64	64	64

● POU Instance Restrictions

POU instances are subject to the following restrictions. The maximum number of POU instances depends on the model and unit version of the CPU Unit and the version of the Sysmac Studio.

Maximum Number of POU Instances for the NX701-□□□□

Restriction	CPU Unit model
	NX701-□□□□
Maximum number of POU instances	48,000

Maximum Number of POU Instances for the NX502-□□□□

Restriction	CPU Unit model
	NX502-□□□□
Maximum number of POU instances	48,000

Maximum Number of POU Instances for the NX102-□□□□

Restriction	CPU Unit model
	NX102-□□□□
Maximum number of POU instances	9,000

Maximum Number of POU Instances for the NX1P2-□□□□

Restriction	CPU Unit model
	NX1P2-□□□□
Maximum number of POU instances	1,800

Maximum Number of POU Instances for the NJ501-□□□□

Restriction	Version	CPU Unit model
	Sysmac Studio	NJ501-□□□□
Maximum number of POU instances	1.05 or lower	6,000
	1.06 or higher	9,000

Maximum Number of POU Instances for the NJ301-□□□□

Restriction	Version		CPU Unit model
	CPU Unit	Sysmac Studio	NJ301-□□□□
Maximum number of POU instances	---	1.04 or lower	1,500
	1.03 or lower	1.05 or higher	2,400
	1.04 or later		3,000

Maximum Number of POU Instances for the NJ101-□□□□

Restriction	CPU Unit model
	NJ101-□□□□
Maximum number of POU instances	1,800

Refer to *Number of POU Instances* on page A-279 for information on counting POU instances.

Restrictions to Variables

There are restrictions to both variable usage and variable definitions.

● Restrictions to Variable Usage

The usage of variables is subject to the following restrictions.

Restriction	CPU Unit model						
	NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum total size*1 in Mbytes of variables without a Retain attribute	256	256 *2	32 *2	2*2	4	2	2
Maximum total size*1 in Mbytes of variables with a Retain attribute	4	4	1.5	0.032*2	2	0.5	0.5
Maximum number of variables*3 without a Retain attribute	360,000	360,000	90,000	90,000	*4	*5	22,500
Maximum number of variables*6 with a Retain attribute	40,000	40,000	10,000	5,000	10,000	*7	5,000
Maximum number of network variables	40,000	40,000	40,000	27,500	40,000	27,500	27,500

*1. The data size of each variable depends on its data type. Refer to *6-3-5 Data Types* on page 6-32 for the sizes of the data types.
*2. Memory for CJ-series Units is included.
*3. Refer to *Number of Variables without a Retain Attribute* on page A-279 for information on counting variables.
*4. The restriction depends on the Sysmac Studio version and the unit version of the CPU Unit as follows:
Sysmac Studio version 1.26 or higher and CPU Unit with unit version 1.20 or later: 180,000
Other combinations: 90,000
*5. The restriction depends on the Sysmac Studio version and the unit version of the CPU Unit as follows:
Sysmac Studio version 1.26 or higher and CPU Unit with unit version 1.20 or later: 90,000
Other combinations: 22,500
*6. Refer to *Number of Variables with a Retain Attribute* on page A-280 for information on counting variables.
*7. The restriction depends on the Sysmac Studio version and the unit version of the CPU Unit as follows:
Sysmac Studio version 1.05 or higher and CPU Unit with unit version 1.04 or later: 5,000
Other combinations: 2500

● Restrictions to Variable Definitions

Variable definitions are subject to the following restrictions.

Restriction	CPU Unit model						
	NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum number of elements per array	65,535	65,535	65,535	65,535	65,535	65,535	65,535
Maximum number of dimensions in an array	3	3	3	3	3	3	3
Maximum size in MB of an array	8	8	4	1	4	2	2
Maximum value of a subscript (element number) for an array	65,535	65,535	65,535	65,535	65,535	65,535	65,535
Maximum size in bytes*1 of a string variable	1,986	1,986	1,986	1,986	1,986	1,986	1,986

*1. The NULL character at the end must be counted. Therefore, there are 1,985 single-byte characters in a string that has a size of 1,986 bytes.

Restrictions to Data Type Definitions

Data type definitions are subject to the following restrictions.

Restriction	CPU Unit model						
	NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum number of data type definitions*1	8,000	8,000	1,000	1,000	2,000	1,000	1,000
Maximum number of levels in a structure definition	8	8	8	8	8	8	8
Maximum number of members in a structure definition	2,048	2,048	2,048	2,048	2,048	2,048	2,048
Maximum size in MB of a structure variable	8	8	4	1	4	2	2
Maximum number of members in a union definition	4	4	4	4	4	4	4
Maximum number of enumerators in an enumeration definition	2,048	2,048	2,048	2,048	2,048	2,048	2,048

*1. Refer to *Number of Data Type Definitions* on page A-280 for information on counting data types.

Restrictions to Constants (Literals)

The constants (literals) are subject to the following restrictions.

Restriction	CPU Unit model						
	NX701-□□□□	NX502-□□□□	NX102-□□□□	NX1P2-□□□□	NJ501-□□□□	NJ301-□□□□	NJ101-□□□□
Maximum size in bytes of a constant (literal)	1,985	1,985	1,985	1,985	1,985	1,985	1,985

A-14-2 Counting User Program Objects

This section describes how to count POU instances, variables with a Retain attribute, variables without a Retain attribute, and data type definitions. The information in this section is provided only as guidelines. The methods for counting objects sometimes varies with the unit version of the CPU Unit. Always use the Sysmac Studio to confirm that user program object sizes are suitable.

Number of POU Instances

POU instances are counted as described below.

● Objects Counted as POU Instances

The following objects are counted as POU instances.

- Programs
- Function block instances (both user-created instances and instructions are included)
- Functions (both user-created instances and instructions are included)

● Precautions in Counting POU Instances

Observe the following precautions when you count POU instances.

- If n instances of a function block are used for the same function block definition, count them as n instances.
- If the same function is used more than once in the same task, count them as one instance regardless of the actual number of functions.
- If the same function is used in different tasks, count them as one instance for each task.

Number of Variables without a Retain Attribute

Variables without a Retain attribute are counted as described below.

● Objects Counted as Variables without a Retain Attribute

The following objects are counted as variables without a Retain attribute.

- Global variables without a Retain attribute
- Local variables without a Retain attribute in programs and function block instances (both user-created instances and instructions are included)

● Precautions in Counting Variables without a Retain Attribute

Observe the following precautions when you count variables without a Retain attribute.

- Count arrays as one variable each regardless of the number of elements.
- Count function block instances as one variable. Both user-created instances and instructions are included for function block instances.
- Count arrays of function block instances as one variable each regardless of the number of elements. However, count one variable for each element of the array for the number of variables without a Retain attribute that are used in the function block.

Number of Variables with a Retain Attribute

Variables with a Retain attribute are counted as described below.

● Objects Counted as Variables with a Retain Attribute

The following objects are counted as variables with a Retain attribute.

- Global variables with a Retain attribute
- Local variables with a Retain attribute in programs and function block instances (both user-created instances and instructions are included)

● Precautions in Counting Variables with a Retain Attribute

Observe the following precautions when you count variables with a Retain attribute.

- Count arrays as one variable each regardless of the number of elements.
- Do not count arrays of function block instances. However, count one variable for each element of the array for the number of variables with a Retain attribute that are used in the function blocks.

Number of Data Type Definitions

Data type definitions are counted as described below.

● Objects Counted as Data Type Definitions

The following objects are counted as data type definitions.

- User-created structure definitions
- User-created union definitions
- User-created enumeration definitions

A-15 Replacing CPU Units with Unit Version 1.02 or Earlier

An NJ-series CPU Unit with a unit version of 1.02 or earlier does not support the SD Memory Card backup functions and Sysmac Studio Controller backup functions. Therefore, the following work is required to replace a CPU Unit when it fails or to change to a newer version.

Work	Description
Upload the data from the CPU Unit.	Upload the following three types of data from the old CPU Unit. Each of these must be uploaded separately. • Project • Present values of variables and memory • Tag data link tables
Connect the new CPU Unit.	Remove the old CPU Unit from the Controller and connect the new CPU Unit.
Download the data to the CPU Unit.	Download the three types of data that you stored in the computer to the new CPU Unit.

Details on the above work is provided in the following sections.

A-15-1 Uploading the Data from the CPU Unit

Upload the following three types of data from the old CPU Unit. Each of these must be uploaded separately. Use the Sysmac Studio and the Network Configurator.

Data to upload	Contents of data	Support Software
Project	• Unit Configuration and Unit Setup • I/O Map • Controller Setup (Operation Settings and Built-in EtherNet/IP Port Settings) • Motion Control Setup • Cam Data Settings • Event Setup • Task Settings • POUs • Data type definitions and global variable definitions	Sysmac Studio
Present values of variables and memory	• Values of variables with a Retain attribute • Values of the Holding, DM, and EM Areas in the memory for CJ-series Units • Absolute encoder home offsets	Sysmac Studio
Tag data link tables*1	• Tag data link settings for EtherNet/IP	Sysmac Studio Network Configurator*2

*1. You need to upload tag data link tables only when tag data links are set.

*2. Use the Network Configurator with the Sysmac Studio version 1.09 or lower.

Precautions for Correct Use

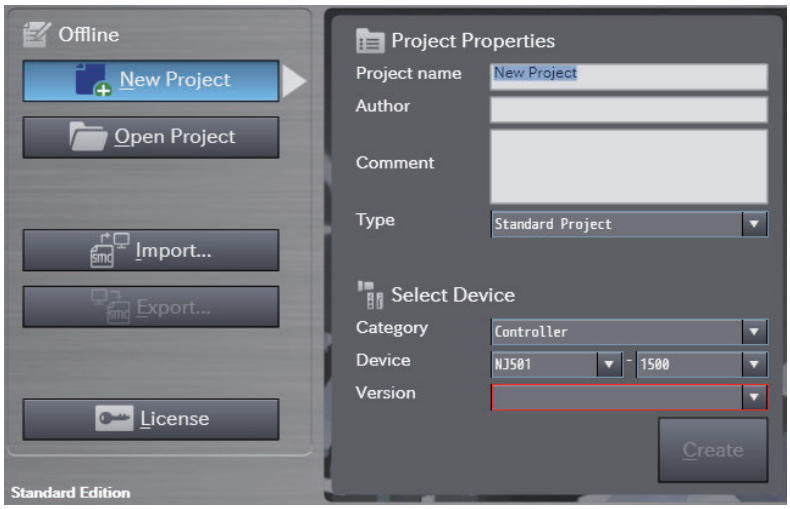
The following data in the CPU Unit is not included in the project, present values of variables and memory, or tag data link tables. Therefore, you must set them again after you replace the CPU Unit.

- Data Trace Settings
- Controller name
- Operation authority verification
- User authentication
- Time zone setting for the built-in clock

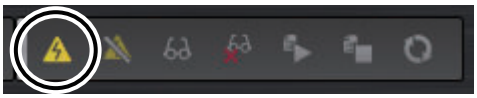
Uploading the Project

Use the following procedure to upload the project.

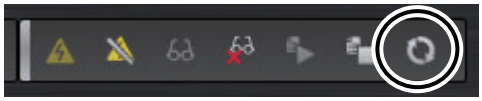
1. Start the Sysmac Studio on the computer that is connected to the NJ-series Controller.
2. Click the **New Project** Button and create a new project.
3. Select the **Device** and **Version** of the CPU Unit to replace.



4. Click the **Online** Button in the toolbar.



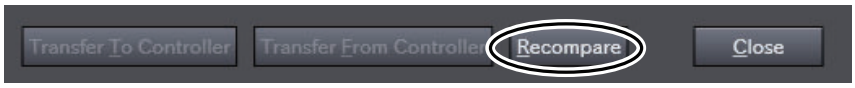
5. Click the **Synchronize** Button in the toolbar.



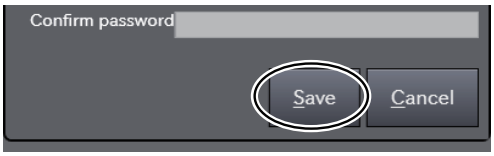
6. Clear the **Do not transfer the EtherNet/IP connection settings (built-in port and Unit)** Check Box and click the **Transfer From Controller** Button.

The project in the Controller is uploaded to the computer.

7 Click the **Recompare** Button.
The uploaded project is compared to the project in the Controller.



8 Click the **Save** Button.
The project is saved in the computer.

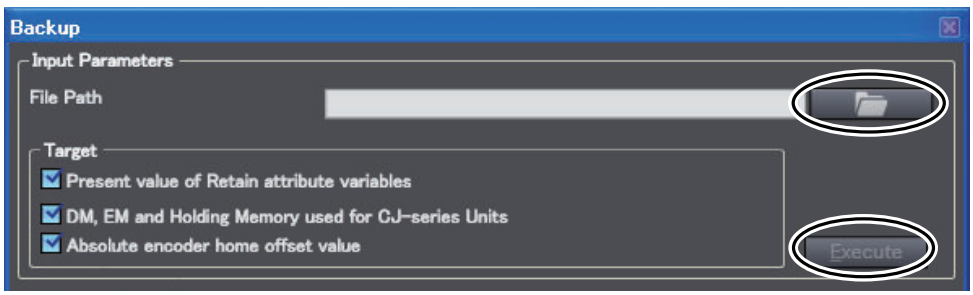


Uploading the Present Values of Variables and Memory

Use the Sysmac Studio variable and memory backup functions to upload the present values of variables and memory. Refer to *9-8 Sysmac Studio Variable and Memory Backup Functions* on page 9-53 for details on the Sysmac Studio variable and memory backup functions.
Use the following procedure.

1 Select **Backup - Backup Variables and Memory** from the Tools Menu on the Sysmac Studio.

2 Select the *Present value of Retain attribute variables*, *DM, EM and Holding Memory used for CJ -series Units*, and *Absolute encoder home offset value* Check Boxes and click the **Execute** Button.
The variable and memory data is uploaded to the computer.



Uploading Tag Data Link Tables

The Network Configurator is used with the Sysmac Studio version 1.09 or lower.

Use the following procedure to upload the tag data link tables.

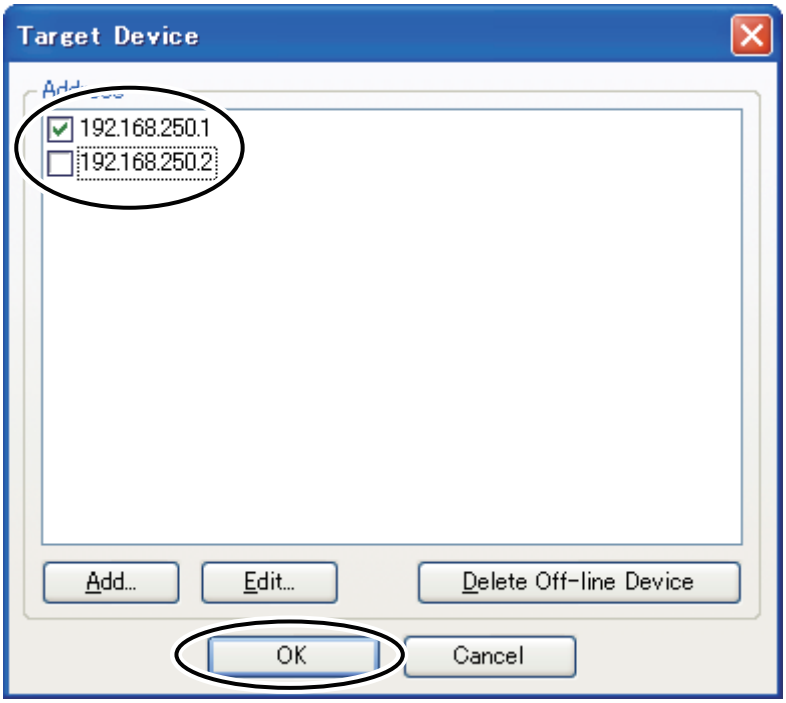
1 Start the Network Configurator on the computer that is connected to the NJ-series Controller.

2 Select **Network** - **Connect** from the toolbar.

3 Select **Network** - **Upload** from the toolbar.
The following message is displayed: **Uploading all devices parameters from network will start based on the current document, OK?**

4 Click the **Yes** Button.

5 Select only the IP address of the connected CPU Unit as the device and click the **OK** Button.



6 Select **File** - **Save As**.
The tag data link tables are uploaded to the computer.

A-15-2 Connecting the New CPU Unit

Remove the old CPU Unit and connect the new CPU Unit. Refer to the *NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)* for details on the connection methods.

Perform the following actions for the new CPU Unit.

- Insert the SD Memory Card that was in the old CPU Unit into the new CPU Unit.
- Set the DIP switch to the same settings as the old CPU Unit.

A-15-3 Downloading the Data to the CPU Unit

Download the project, present values of variables and memory, and tag data link tables to the new CPU Unit.

Precautions for Safe Use

Check the operation of the downloaded project for proper execution before you use it for actual operation.

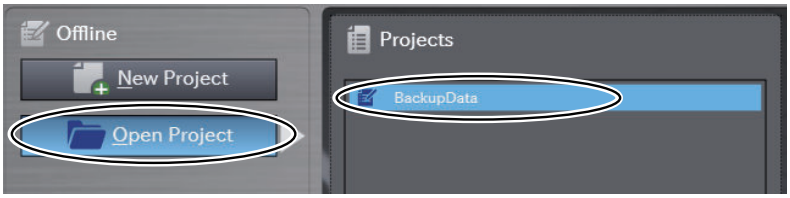
Downloading the Project

Use the following procedure to download the project.

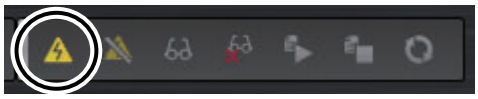
1 Start the Sysmac Studio on the computer that is connected to the NJ-series Controller.

2 Click the **Open Project** Button.

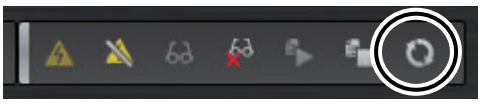
3 From the project list, select the project that you uploaded from the old CPU Unit.
In the following example, the name of the project that you uploaded from the previous CPU Unit is *BackupData*.



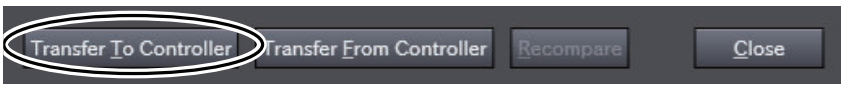
4 Click the **Online** Button in the toolbar.



5 Click the **Synchronize** Button in the toolbar.



6 Clear the **Do not transfer the EtherNet/IP connection settings (built-in port and Unit)** Check Box and click the **Transfer To Controller** Button.
The project in the computer is downloaded to the Controller.

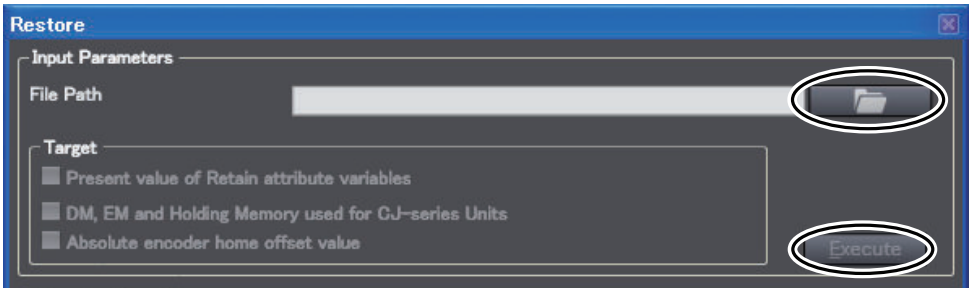


Downloading the Present Values of Variables and Memory

Use the Sysmac Studio variable and memory backup functions to download the present values of variables and memory. Refer to *9-8 Sysmac Studio Variable and Memory Backup Functions* on page 9-53 for details on the Sysmac Studio variable and memory backup functions.
Use the following procedure.

1 Select **Backup** - **Restore Variables and Memory** from the Tools Menu on the Sysmac Studio.

2 Select the data file that you uploaded from the old CPU Unit and click the **Execute** Button.
The variable and memory data is downloaded to the Controller.



Downloading Tag Data Link Tables

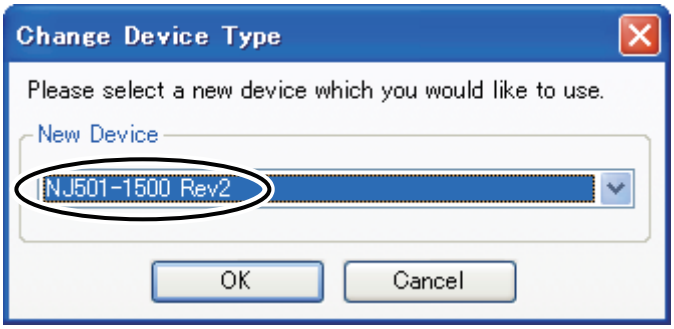
The Network Configurator is used with the Sysmac Studio version 1.09 or lower.
Use the following procedure to download the tag data link tables.

1 Start the Network Configurator on the computer that is connected to the NJ-series Controller.

2 Select **Network** - **Connect** from the toolbar.

3 Select **File** - **Open** and open the tag data link table file that you uploaded from the old CPU Unit.

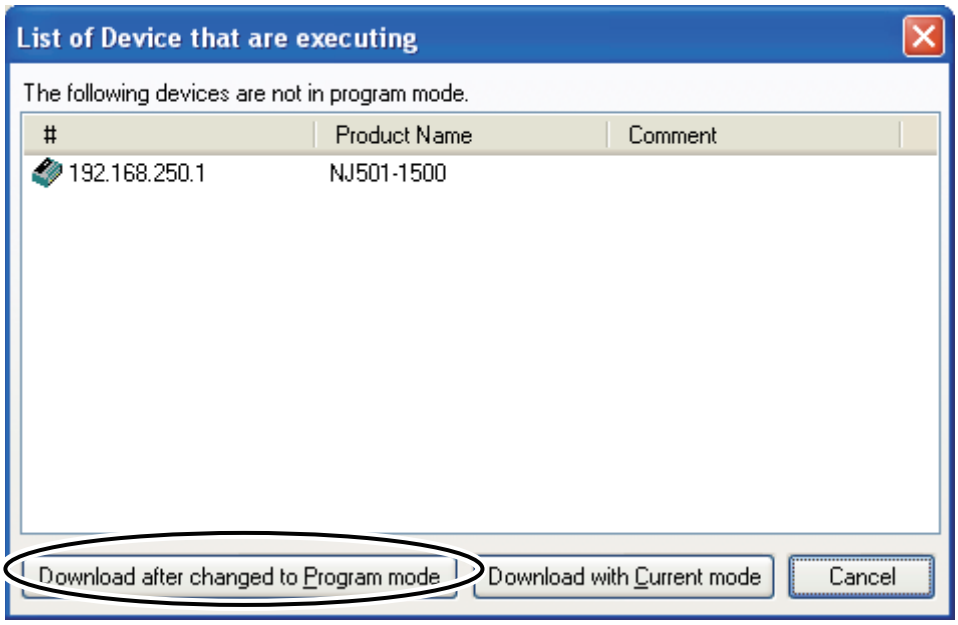
4 To download tag data link tables to a CPU Unit with unit version 1.03 or later when the tag data link tables were uploaded from a CPU Unit with unit version 1.02 or earlier, select **Device**–**Change Device** and select *NJ□□□-1□□□ Rev2* as the *New Device*.



5 Select **Network** − **Download** from the toolbar.
The following message is displayed: **In order to enable new configuration, downloading parameters to all devices will start, OK?**

6 Click the **Yes** Button.
A list of the currently active devices is displayed.

7 Click the **Download after changed to Program mode** Button.
The tag data link tables are downloaded to the Controller.



Precautions for Correct Use

If *NJ□□□-1□□□ Rev2* is not displayed as a *new device* selection in step 4, above, use the newest version of the Network Configurator.

A-16 Possible Combination of Windows Versions and CPU Unit Versions

Depending on the unit version of the CPU Unit, the Windows version on which support software can be used differs.

The target support software is as follows.

Support Software	Description
Sysmac Studio	
Sysmac Controller Log Upload Tool	Included with Sysmac Studio.
CX-ConfiguratorFDT	Included with Sysmac Studio.
Event Log Utility	Included with SYSMAC Gateway.

The Windows versions on which support software can be used for each CPU Unit version are shown below. Use a computer with the required version of Windows according to the unit version of the CPU Unit.

CPU Unit	Unit version	Windows version		Reason why they cannot be used in combination
		10 version 1803 or higher	7, 8, 10 version 1709 or lower	
NX502-□□□□ NX102-□□□□ NX1P2-□□□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Ver. 1.49 or later	Can be used.	Cannot be used.	The support software and the CPU Unit cannot communicate.
	Version earlier than 1.49	Can be used.	Can be used.	---
NX701-□□□□	Ver.1.29 or later	Can be used.	Cannot be used.	The support software and the CPU Unit cannot communicate.
	Version earlier than 1.29	Can be used.	Can be used.	---

A-17 Device Variables

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 device 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 device variable specifications table is given.

A-17-1 X Bus Unit Common Function Module

Variable name	Meaning	Function	Data type	Range of values	Reference
PowerOnHour	Total Power ON Time	Contains the total time that the X Bus Unit power has been ON. Contains the total time that the X Bus Unit has been ON in 1-hour increments. The value is not updated after it reaches 4294967295. This variable is not initialized at startup.	UDINT	0 to 4294967295	page A-291
PowerOnCount	Power Interruption Count	Contains the number of times that the power supply to the X Bus Unit has been interrupted. The value is incremented by 1 each time the power supply is interrupted after the first time that the power to the X Bus Unit was turned ON. The value is not updated after it reaches 4294967295. This variable is not initialized at startup.	UDINT	0 to 4294967295	page A-291
UnitVersion	Unit Version	Contains the unit version of the X Bus Unit. 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) If the unit version is 1.08, “1” is stored in element number 0 and “8” is stored in element number 1. Example) 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-292
HardwareRevision	Hardware Revision	Contains the hardware revision of the X Bus Unit. Contains - if the hardware revision is in blank, and A to Z for other cases.	STRING[2]	- or A to Z	page A-292

Variable name	Meaning	Function	Data type	Range of values	Reference
ProjectUnitVersion	Project Unit Version	Contains the project unit version of the project stored in the X Bus Unit. The integer part of the project unit version is stored in element number 0. The fractional part of the project unit version is stored in element number 1. Example) If the project unit version is 1.09, “1” is stored in element number 0 and “9” is stored in element number 1. Example) If the project unit version is 1.40, “1” is stored in element number 0 and “40” is stored in element number 1.	ARRAY [0..1] OF USINT	0 to 99	page A-292
StorageDeteriorated	Non-volatile Memory Life Warning Flag	TRUE when the number of deletions for non-volatile memory or the number of bad blocks in memory exceeds the warning value.	BOOL	TRUE or FALSE	page A-292
XBU_ErrSta	X Bus Unit Common Function Module Error Status	Gives the status of errors that are detected in the X Bus Unit Common Function Module and the X Bus Function Module.	WORD	16#0000 to 16#00F0	page A-292

A-18 Specifications for Individual Device Variables

The specifications for each device variable are given as described below.

Variable name	This is the device variable name. The prefix gives the category name.			**Members (not all)**	The member names are given for structure variables only.
Meaning	This is the meaning of the variable.			**Global/local**	Global: Global variable, Local: Local variable
Function	The function of the variable is described.				
Data type	The data type of the variable is given.			**Range of values**	The range of values that the variable can take is given.
R/W access	R: Read only, RW: Read/write	**Retained**	The Retain attribute of the variable is given.	**Network Publish**	The Network Publish attribute of the variable is given.
Usage in user program	Whether you can use the variable directly in the user program is specified.	**Related instructions**	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.		

A-18-1 X Bus Unit Common Function Module

Variable name	PowerOnHour				
Meaning	Total Power ON Time			**Global/local**	Global
Function	Contains the total time that the X Bus Unit power has been ON. Contains the total time that the X Bus Unit has been ON in 1-hour increments. 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	**Retained**	Retained.	**Network Publish**	Published.
Usage in user program	Possible.	**Related instructions**	---		

Variable name	PowerOnCount				
Meaning	Power Interruption Count			**Global/local**	Global
Function	Contains the number of times that the power supply to the X Bus Unit has been interrupted. The value is incremented by 1 each time the power supply is interrupted after the first time that the power to the X Bus Unit was turned ON. 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	**Retained**	Retained.	**Network Publish**	Published.
Usage in user program	Possible.	**Related instructions**	---		

Variable name	UnitVersion				
Meaning	Unit Version			Global/local	Global
Function	Contains the unit version of the X Bus Unit. 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) If the unit version is 1.08, “1” is stored in element number 0 and “8” is stored in element number 1. Example) 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	Retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	HardwareRevision				
Meaning	Hardware Revision			Global/local	Global
Function	Contains the hardware revision of the X Bus Unit. 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	Retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	ProjectUnitVersion				
Meaning	Project Unit Version			Global/local	Global
Function	Contains the project unit version of the project stored in the X Bus Unit. The integer part of the project unit version is stored in element number 0. The fractional part of the project unit version is stored in element number 1. Example) If the project unit version is 1.09, “1” is stored in element number 0 and “9” is stored in element number 1. Example) If the project unit version is 1.40, “1” is stored in element number 0 and “40” is stored in element number 1.				
Data type	ARRAY [0..1] OF USINT			Range of values	0 to 99
R/W access	R	Retained	Retained.	Network Publish	Published.
Usage in user program	Possible.	Related instructions	---		

Variable name	StorageDeteriorated				
Meaning	Non-volatile Memory Life Warning Flag			Global/local	Global
Function	TRUE when the number of deletions for non-volatile memory or the number of bad blocks in memory exceeds the warning value.				
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	XBU_ErrSta				
Meaning	X Bus Unit Common Function Module Error Status			Global/local	Global
Function	Gives the status of errors that are detected in the X Bus Unit Common Function Module and the X Bus Function Module.				
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	---		

A-19 Version Information for NX-series Controllers

This section describes the relationship between the unit versions of NX-series CPU Units and the Sysmac Studio versions, and the functions that are supported for each unit version.

A-19-1 Relationship between Unit Versions of CPU Units and Sysmac Studio Versions

This section also describes how the unit versions of NX-series CPU Units correspond to Sysmac Studio versions. Normally use the corresponding versions.

Precautions for Correct Use

For the following CPU Unit versions, use Sysmac Studio on Windows 10 version 1803 or higher OS on the computer. Sysmac Studio on Windows 10 version 1709 or lower cannot connect online to the Controller.

- NX102, NX1P2 CPU Unit: Version 1.49 or later
- NX701 CPU Unit: Version 1.29 or later
- NX502 CPU Unit: Version 1.60 or later

Unit Versions and Corresponding Sysmac Studio Versions

The following table gives the relationship between unit versions of NX-series CPU Units and the corresponding Sysmac Studio versions.

Unit version of CPU Unit	Corresponding version of Sysmac Studio
Ver.1.63*1	Ver.1.55
Ver.1.60*2	Ver.1.54
Ver.1.60*3	Ver.1.53
Ver.1.32*4	
Ver.1.50*3	Ver.1.52
Ver.1.49*3	Ver.1.50
Ver.1.29*4	
Ver.1.48*5	Ver.1.49
Ver.1.28*4	
Ver.1.47*6	Ver.1.47
Ver.1.38*7	
Ver.1.27*8	
Ver.1.46*6	Ver.1.46
Ver.1.37*7	
Ver.1.26*8	
Ver.1.43*9	Ver.1.44
Ver.1.36*7	

Unit version of CPU Unit	Corresponding version of Sysmac Studio
Ver.1.24*8	
Ver.1.42*6	Ver.1.43
Ver.1.35*7	Ver.1.41
Ver.1.23*8	
Ver.1.41*10	Ver.1.40
Ver.1.34*7	
Ver.1.22*8	
Ver.1.40*11	Ver.1.30*12
	Ver.1.29
Ver.1.21*13	
Ver.1.32*5	Ver.1.28
Ver.1.21*14	
Ver.1.31*15	Ver.1.24
Ver.1.30*16	Ver.1.23
Ver.1.18*17	Ver.1.22
Ver.1.16*18	Ver.1.20
Ver.1.14	Ver.1.18
Ver.1.13*19	Ver.1.17*20
Ver.1.12	Ver.1.16
Ver.1.11	Ver.1.15
Ver.1.10	Ver.1.13

*1. This is the unit version of NX502-□□□□ CPU Units.
*2. This is the unit version of NX502-□□□□ CPU Units.
*3. This is the unit version of NX102-□□□□ CPU Units and NX1P2-□□□□□□□□ CPU Units.
*4. This is the unit version of NX701-□□□□ CPU Units.
*5. This is the unit version of NX102-□□□□ CPU Units.
*6. This is the unit version of NX102-□□00 CPU Units and NX1P2-□□□□□□□□ CPU Units.
*7. This is the unit version of NX102-□□20 CPU Units.
*8. This is the unit version of NX701-□□□□ CPU Units.
*9. This is the unit version of NX102-□□00 CPU Units.
*10. This is the unit version of NX102-□□00 CPU Units and NX1P2-□□□□□□□□ CPU Units.
*11. This is the unit version of NX102-□□00 CPU Units and NX1P2-□□□□□□□□ CPU Units.
There is no NX1P2-9B□□□□□ CPU Unit with unit version earlier than 1.40.
*12. Use an NX1P2-9B□□□□□ CPU Unit with Sysmac Studio version 1.30 or higher.
You cannot use an NX1P2-9B□□□□□ CPU Unit with Sysmac Studio version 1.29 or lower.
*13. This is the unit version of NX701-□□□□ CPU Units.
*14. This is the unit version of NX1P2-□□□□□□□□ CPU Units.
*15. This is the unit version of NX102-□□□□ CPU Units. There is no NX701-□□□□ CPU Unit and NX1P2-□□□□□□□ CPU Unit with unit version 1.31.
*16. This is the unit version of NX102-□□□□ CPU Units. There is no NX102-□□□□ CPU Unit with unit version 1.29 or earlier. There are no NX701-□□□□ CPU Unit and NX1P2-□□□□□□□□ CPU Unit with unit version 1.30.
*17. There are no NX701-□□□□ CPU Unit and NX1P2-□□□□□□□□ CPU Unit with unit version 1.17.
*18. There are no NX701-□□□□ and NX1P2-□□□□□□□□ CPU Units with unit version 1.15.
*19. There is no NX1P2-□□□□□□□□ CPU Unit with unit version 1.12 or earlier.
*20. Use an NX1P2-□□□□□□□□ CPU Unit with Sysmac Studio version 1.17 or higher. You cannot use an NX1P2-□□□□□□□□ CPU Unit with Sysmac Studio version 1.16 or lower.

Specifications When Not Using the Sysmac Studio Version That Corresponds to the Unit Version of the CPU Unit

The specifications when you do not use the Sysmac Studio version that corresponds to the unit version of the NX-series CPU Unit are given in this section.

● Using Sysmac Studio Version 1.12 or Lower

You cannot use the NX-series CPU Unit with Sysmac Studio version 1.12 or lower.

● Using a Lower Version of Sysmac Studio

If you use a lower version of the Sysmac Studio, you can use only the functions of the unit version of the CPU Unit that corresponds to the Sysmac Studio version.

Example: Unit version of CPU Unit: 1.11

Sysmac Studio version: 1.13

Unit version 1.10 of the CPU Unit corresponds to Sysmac Studio version 1.13. Therefore, you can use only the functions that are supported by unit version 1.10 of the CPU Unit. You cannot use functionality that was added for unit version 1.11 or later of the CPU Unit.

● Using a CPU Unit with an Earlier Unit Version

If you use an NX-series CPU Unit with an earlier version, select the unit version of the used CPU Unit or an earlier unit version in the Select Device Area of the Project Properties Dialog Box on the Sysmac Studio. The unit version that you selected is the project unit version of the project. You can use only the functions that are supported by the project unit version.

Example: Unit version of CPU Unit: 1.11

Sysmac Studio version: 1.16

Unit version 1.12 of the CPU Unit corresponds to Sysmac Studio version 1.16. However, the used CPU Unit is unit version 1.11, so select version 1.11 or earlier as the version in the Select Device Area of the Project Properties Dialog Box.

If you select version 1.11 as the version in the Select Device Area of the Project Properties Dialog Box, you can use only the functions that are supported by project unit version 1.11. You cannot use functionality that was added for unit version 1.12 or later of the CPU Unit.

Additional Information

Project Unit Version for Projects

- With Sysmac Studio version 1.02 or higher, you can select the unit version in the Select Device Area of the relevant dialog boxes.
- You can select any unit version that is the same as or earlier than the unit version of the CPU Unit. For example, if the unit version of the CPU Unit is 1.11, select either *1.10* or *1.11*.
- The Sysmac Studio will treat the project unit version as the unit version for the CPU Unit. For example, if the project unit version is 1.10, you can use the functionality for unit version 1.10 on the Sysmac Studio.
- You can transfer a project to the Sysmac Studio if the project unit version is the same as or earlier than the unit version of the destination CPU Unit.
- Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on the Select Device Area of the relevant dialog boxes.

A-19-2 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 NX-series CPU Unit.

● Additions and Changes to Functional Specifications

The following table gives the unit version of the CPU Units and the Sysmac Studio version for each addition or change to the functional specifications.

Function				Addition/ change	Unit version	Sysmac Studio version	Reference
Pro-gram-ming	Variable	Arrays	Variable-length array	Addition	Ver.1.18	Ver.1.22	*6-3-7 Array Specifications and Range Specifications for Data Types* on page 6-51
Motion control	Single axis	Auxiliary functions for single-axis con-trol	Cam Monitor	Addition	Ver.1.32/ Ver.1.21	Ver.1.28/ Ver.1.29	*NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)*
Com-muni-cations	Built-in EtherNet/ IP port	TCP/IP function	Packet Filter (Simple) *1	Addition	Ver.1.30	Ver.1.23	*NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)*
			Packet Filter	Addition	*2	Ver.1.50	
			DHCP client	Addition	Ver1.60*5	Ver.1.54	
		CIP Safety routing*1		Addition	Ver.1.31	Ver.1.24	*NX-series Safety Control Unit User's Manual (Cat. No. Z930-E1-12 or later)*
		Secure socket service*3		Addition	Ver.1.37/ Ver.1.46	Ver.1.46	*NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)*
		OPC UA	Server func-tion	Addition	Refer to the *NJ/NX-series CPU Unit OPC UA User's Manual (Cat. No. W588)* for information on the models and unit versions of the CPU Units that support OPC UA, and the corresponding Sysmac Studio versions.		
		Others	Modbus/ TCP(Client) *1	Addition	Ver.1.30	Ver.1.23	*NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)*
			FINS commu-nications service*1	Addition	Ver.1.30	Ver.1.23	*NX-series CPU Unit FINS Function User's Manual (Cat. No. W596)*
	EtherCAT port	Packet monitoring		Change*4	Ver.1.40	Ver.1.29	*NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual (Cat. No. W505)*
	Communications instructions			Change	Ver.1.11 Ver.1.30 *1	Ver.1.15 Ver.1.23	*NJ/NX-series Instructions Reference Manual (Cat. No. W502)*

Function				Addition/ change	Unit version	Sysmac Studio version	Reference
Security	Asset protection and preventing incorrect operation	User authentication	Number of groups	Addition	*2	Ver.1.50	*8-6-11 User Authentication* on page 8-53
SD Memory Cards	Application	Program transfer from SD Memory Card		Addition	Ver.1.11	Ver.1.15	*9-5 Program Transfer from SD Memory Card* on page 9-39
Backing up Data	Safety unit restore*1			Addition	Ver.1.31	Ver.1.24	*9-1-4 Types of Backup Functions* on page 9-8
PLC	Firmware update prohibition			Addition	Ver.1.60/ Ver.1.32	Ver.1.53	*8-6-13 Firmware Update Prohibition* on page 8-58
Debugging	Automation playback*5			Addition	Ver.1.63	Ver.1.55	*8-7-6 Automation Playback* on page 8-80

*1. This addition applies only to an NX102 CPU Unit.
*2. For information on the CPU Unit models and unit versions supporting the function, refer to the listed references.
*3. This addition applies only to an NX102 CPU Unit and NX1P2 CPU Unit.
*4. Packet monitoring can be used with project unit version earlier than 1.40. It cannot be used with project unit version 1.40 or later.
*5. This addition applies only to an NX502 CPU Unit.

Note Refer to the manuals for the function modules for additions and changes to function module functions for each unit version of the CPU Units.

● Additions and Changes to Basic Instructions and Motion Control Instructions

The basic instructions and motion control instructions that you can use have added or changed for the new unit version of the CPU Unit.
For details, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* and *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

● Additions and Changes to Controller Events

The events that can occur have added or changed for the new unit version of the CPU Unit. There are also changes in the recovery methods to use when some errors occur.
For details, refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)*.

● Additions and Changes to System-defined Variables

The system-defined variables that you can use have added or changed for the new unit version of the CPU Unit.
Refer to *A-7 System-defined Variables* on page A-91 for details.

A-20 Version Information for NJ-series Controllers

This section describes the relationship between the unit versions of NJ-series CPU Units and the Sysmac Studio versions, and the functions that are supported for each unit version.

A-20-1 Relationship between Unit Versions of CPU Units and Sysmac Studio Versions

This section also describes how the unit versions of NJ-series CPU Units correspond to Sysmac Studio versions. Normally use the corresponding versions.

Precautions for Correct Use

For the following CPU Unit versions, use Sysmac Studio on Windows 10 version 1803 or higher OS on the computer. Sysmac Studio on Windows 10 version 1709 or lower cannot connect online to the Controller.

- NJ-series CPU Unit: Ver. 1.49 or later

Unit Versions and Corresponding Sysmac Studio Versions

The following table gives the relationship between unit versions of NJ-series CPU Units and the corresponding Sysmac Studio versions.

Unit version of CPU Unit	Corresponding version of Sysmac Studio
Ver.1.60	Ver.1.53
Ver.1.50	Ver.1.52
Ver.1.49	Ver.1.50
Ver.1.48*1	Ver.1.49
Ver.1.47*2	
Ver.1.47*3	Ver.1.47
Ver.1.27*4	
Ver.1.46*3	Ver.1.46
Ver.1.26*4	
Ver.1.43*5	Ver.1.44
Ver.1.42*6	Ver.1.43
Ver.1.41*7	Ver.1.42
Ver.1.23*8	Ver.1.41
Ver.1.41*9	Ver.1.40
Ver.1.40*10	Ver.1.29
Ver.1.21*11	
Ver.1.21*12	Ver.1.28
Ver.1.20	Ver.1.26
Ver.1.19	Ver.1.24

Unit version of CPU Unit	Corresponding version of Sysmac Studio
Ver.1.18	Ver.1.22
Ver.1.17	Ver.1.21
Ver.1.16	Ver.1.20
Ver.1.15	Ver.1.19
Ver.1.14	Ver.1.18
Ver.1.13	Ver.1.17
Ver.1.12	Ver.1.16
Ver.1.11	Ver.1.15
Ver.1.10*13	Ver.1.13*14
	Ver.1.12
Ver.1.09	Ver.1.10
Ver.1.08	Ver.1.09
Ver.1.07	Ver.1.08
Ver.1.06	Ver.1.07
Ver.1.05	Ver.1.06
Ver.1.04	Ver.1.05
Ver.1.03	Ver.1.04
Ver.1.02	Ver.1.03
Ver.1.01	Ver.1.02
Ver.1.00*15	Ver.1.01
	Ver.1.00

*1. This is the unit version of NJ501-1□00 CPU Units.
*2. This is the unit version of NJ501-□□20, NJ501-1340, NJ501-5300, NJ501-4□□□, and NJ101-□□20 CPU Units.
*3. This is the unit version of NJ501-1□00, NJ501-R□□□, NJ301-□□□□, and NJ101-□□00 CPU Units.
*4. This is the unit version of NJ501-□□20, NJ501-4□□□, NJ501-1340, NJ501-5300, and NJ101-1□20 CPU Units.
*5. This is the unit version of NJ501-1□00 CPU Units.
*6. This is the unit version of NJ501-1□00, NJ301-1□00, and NJ101-□□00 CPU Units.
*7. This is the unit version of NJ501-R□00 CPU Units.
*8. This is the unit version of NJ501-1□20, NJ501-4320, and NJ101-□□20 CPU Units.
*9. This is the unit version of NJ501-1□00, NJ301-1□00, and NJ101-□□00 CPU Units.
*10. This is the unit version of NJ501-1□00, NJ301-1□□□, and NJ101-□□00 CPU Units.
*11. This is the unit version of NJ501-4□00, NJ501-4□10, NJ501-1340, and NJ501-5300 CPU Units.
*12. This is the unit version of NJ501-1□00, NJ301-1□00, and NJ101-□□00 CPU Units.
*13. There is no NJ101-□□□□ CPU Unit with unit version 1.09 or earlier.
*14. Use an NJ101-□□□□ CPU Unit with Sysmac Studio version 1.13 or higher. You cannot use an NJ101-□□□□ CPU Unit with Sysmac Studio version 1.12 or lower.
*15. There is no NJ301-□□□□ CPU Unit with unit version 1.00. Therefore, you cannot use an NJ301-□□□□ CPU Unit with Sysmac Studio version 1.01 or lower.

Specifications When Not Using the Sysmac Studio Version That Corresponds to the Unit Version of the CPU Unit

The specifications when you do not use the Sysmac Studio version that corresponds to the unit version of the NJ-series CPU Unit are given in this section.

● Using a Lower Version of Sysmac Studio

If you use a lower version of the Sysmac Studio, you can use only the functions of the unit version of the CPU Unit that corresponds to the Sysmac Studio version.

Example: Unit version of CPU Unit: 1.04

Sysmac Studio version: 1.04

Unit version 1.03 of the CPU Unit corresponds to Sysmac Studio version 1.04. Therefore, you can use only the functions that are supported by unit version 1.03 of the CPU Unit. You cannot use functionality that was added for unit version 1.04 or later of the CPU Unit.

● Using a CPU Unit with an Earlier Unit Version

If you use an NJ-series CPU Unit with an earlier version, select the unit version of the used CPU Unit or an earlier unit version in the Select Device Area of the Project Properties Dialog Box on the Sysmac Studio. The unit version that you selected is the project unit version of the project. You can use only the functions that are supported by the project unit version.

Example: Unit version of CPU Unit: 1.03

Sysmac Studio version: 1.05

Unit version 1.04 of the CPU Unit corresponds to Sysmac Studio version 1.05. However, the used CPU Unit is unit version 1.03, so select version 1.03 or earlier as the version in the Select Device Area of the Project Properties Dialog Box.

If you select version 1.03 as the version in the Select Device Area of the Project Properties Dialog Box, you can use only the functions that are supported by project unit version 1.03. You cannot use functionality that was added for unit version 1.04 or later of the CPU Unit. You cannot use functionality that was added for unit version 1.04 or later of the CPU Unit.

Precautions for Correct Use

An error will occur if you perform the following type of operation. Use it with caution.

- Create a project on Sysmac Studio version 1.02 or higher with unit version 1.01 or later selected as the version in the Select Device Area of the Project Properties Dialog Box.
- Upload the project to Sysmac Studio version 1.01.

Additional Information

Project Unit Version for Projects

- With Sysmac Studio version 1.02 or higher, you can select the unit version in the Select Device Area of the relevant dialog boxes.
- You can select any unit version that is the same as or earlier than the unit version of the CPU Unit. For example, if the unit version of the CPU Unit is 1.01, select either *1.00* or *1.01*.
- The Sysmac Studio will treat the project unit version as the unit version for the CPU Unit. For example, if the project unit version is 1.00, you can use the functionality for unit version 1.00 on the Sysmac Studio.
- You can transfer a project to the Sysmac Studio if the project unit version is the same as or earlier than the unit version of the destination CPU Unit.
- Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on the Select Device Area of the relevant dialog boxes.

A-20-2 Functions That Were Added or Changed for Each Unit Version

This section gives the functions that were added or changed for each unit version of NJ-series CPU Unit.

● Additions and Changes to Functional Specifications

The following table gives the unit version of the CPU Units and the Sysmac Studio version for each addition or change to the functional specifications.

Function				Addition/ Change	Unit version	Sysmac Studio version	Reference
Tasks	Function	Conditionally executed tasks		Addition	1.03	1.04	*5-2 Overview of Tasks* on page 5-6
Programming	Namespaces			Addition	1.01	1.02	*6-7 Namespaces* on page 6-142
	Data types	Structure data types	Specifying member offsets	Addition	1.01	1.02	*Specifying Structure Member Offsets* on page 6-44
				Change		1.03	
	Variables	Arrays	Variable-length array	Addition	1.18	1.22	*6-3-7 Array Specifications and Range Specifications for Data Types* on page 6-51
	Libraries			Addition	1.01	1.02	*6-8 Libraries* on page 6-147
Motion control	Single axis	Single-axis position control	Cyclic synchronous absolute positioning	Addition	1.03	1.04	*NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)*
		Auxiliary functions for single-axis control	Homing with specified parameters	Addition	1.03	1.04	
			Enabling digital cam switches	Addition	1.06	1.07	
			Slave axis position compensation	Addition	1.10	1.12	
			Cam monitor	Addition	1.21	1.28/1.29	
			Start velocity	Addition	1.05	1.06	
	Axes groups	Multi-axes coordinated control	Axes group cyclic synchronous absolute positioning	Addition	1.01	1.02	
		Auxiliary functions for multi-axes coordinated control	Reading axes group positions	Addition	1.01	1.02	
			Changing the axes in a group	Addition	1.01	1.02	
	Common items	Cams	Generating cam tables	Addition	1.08	1.09	
		Parameters	Changing axis parameters	Addition	1.08	1.09	
	Auxiliary functions	Input signal logic inversion		Addition	1.05	1.06	
Unit (I/O) management	NX Units			Addition	1.05	1.06	*NX-series EtherCAT Coupler Unit User's Manual (Cat. No. W519)*

Function				Addition/ Change	Unit version	Sysmac Studio version	Reference
Communications	Built-in EtherNet/IP port	TCP/IP function	Packet Filter	Addition	*1	1.50	*NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual (Cat. No. W506)*
		TCP/IP applications	FTP client	Addition	1.08	1.09	
		OPC UA	Server Function	Addition	Refer to the *NJ/NX-series CPU Unit OPC UA User's Manual (Cat. No. W588)* for models and unit versions of the CPU Units which support OPC UA function and corresponding Sysmac Studio versions.		
	Built-in EtherCAT port	Packet monitoring*2 (NJ301-□□□□)		Addition Change*3	1.10 1.40	1.12 1.29	*NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual (Cat. No. W505)*
	Communications instructions			Change	1.08 1.11	1.09 1.15	*NJ/NX-series Instructions Reference Manual (Cat. No. W502)*
Debugging function	Differential monitoring			Addition	1.03	1.04	*8-7-5 Differential Monitoring* on page 8-75
Reliability functions	Self diagnosis	Controller errors	Changing levels	Addition	1.03	1.04	*8-9 Changing Event Levels* on page 8-95
Security	Asset protection and preventing incorrect operation	Protection	Data protection	Addition	1.01	1.02	*8-6-4 Data Protection* on page 8-46
		Operation authority verification	Number of groups	Change	1.01	1.02	*8-6-10 Operation Authority Verification* on page 8-51
		User authentication	Number of groups	Addition	*1	1.50	*8-6-11 User Authentication* on page 8-53
SD Memory Cards	Application	Program Transfer from SD Memory Card		Addition	1.11	1.15	*9-5 Program Transfer from SD Memory Card* on page 9-39
Backing up Data	SD Memory Card backups	Operation methods	CPU Unit front-panel DIP switch	Addition	1.03	1.04	*9-2 SD Memory Card Backups* on page 9-15
			Specification with system-defined variables	Addition	1.03	1.04	
			SD Memory Card Window in Sysmac Studio	Addition	1.03	1.04	
			Special instruction	Addition	1.08	1.09	
		Protection	Disabling backups to SD Memory Cards	Addition	1.03	1.04	*9-3 Disabling Backups to SD Memory Cards* on page 9-35
	Sysmac Studio Controller backups			Addition	1.03	1.04	*9-6 Sysmac Studio Controller Backups* on page 9-46

Function		Addition/ Change	Unit ver- sion	Sysmac Studio version	Reference
PLC	Firmware update prohibition	Addition	1.60	1.53	*8-6-13 Firmware Update Prohibition* on page 8-58

*1. For information on the CPU Unit models and unit versions supporting the function, refer to the listed references.
*2. This addition applies only to an NJ301-□□□□ CPU Unit. The NJ501-□□□□ and NJ101-□□□□ CPU Units support packet monitoring with all versions.
*3. Packet monitoring can be used with project unit version earlier than 1.40. It cannot be used with project unit version 1.40 or later.

Note Refer to the manuals for the function modules for additions and changes to function module functions for each unit version of the CPU Units.

● Addition of Mountable CJ-series Units

The CJ-series Units that can be mounted have added for the new unit version of the CPU Unit.
For details, refer to the *NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)*.

● Additions and Changes to Basic Instructions and Motion Control Instructions

The basic instructions and motion control instructions that you can use have added or changed for the new unit version of the CPU Unit.
For details, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* and *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

● Additions and Changes to Controller Events

The events that can occur have added or changed for the new unit version of the CPU Unit. There are also changes in the recovery methods to use when some errors occur.
For details, refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)*.

● Additions and Changes to System-defined Variables

The system-defined variables that you can use have added or changed for the new unit version of the CPU Unit.
Refer to *A-7 System-defined Variables* on page A-91 for details.

A-20-3 Performance Improvements for Unit Version Upgrades

This section introduces the functions for which performance was improved for each unit version of NJ-series CPU Unit and for each Sysmac Studio version.

Function				Performance value	Unit version	Sysmac Studio version
Programming	Program capacity	Quantities	Number of POU instances (NJ501-□□□□)	9,000	---	1.06 or higher
				6,000	---	1.05 or lower
			Number of POU instances (NJ301-□□□□)	3,000	1.04 or later	1.05 or higher
				1,500		1.04 or lower
				2,400	1.03 or earlier	1.05 or higher
				1,500		1.04 or lower
	Memory capacity for variables	Variables with a Retain attribute	Number of variables*1 (NJ301-□□□□)	5,000	1.04 or later	1.05 or higher
				2,500		1.04 or lower
				2,500	1.03 or earlier	---
		Variables without a Retain attribute	Number of variables (NJ501-□□□□)	180,000	Ver.1.20 or later	Ver.1.26 or higher
				90,000	Other than the above combination	
			Number of variables (NJ301-□□□□)	90,000	Ver.1.20 or later	Ver.1.26 or higher
				22,500	Other than the above combination	
Motion control	Number of controlled axes	Maximum number of controlled axes*2*3*4 (NJ301-□□□□)		15 axes	1.06 or later	1.07 or higher
				8 axes (NJ301-1200) 4 axes (NJ301-1100)	Other than the above combination	
		Maximum number of axes for single-axis control*4*5 (NJ301-□□□□)		15 single axes	1.06 or later	1.07 or higher
				8 single axes (NJ301-1200) 4 single axes (NJ301-1100)	Other than the above combination	
Built-in EtherNet/IP port	CIP service: Tag data links (cyclic communications)	Packet interval		Can be set for each connection. 1 to 10,000 ms in 1-ms increments	1.03 or later	---
				Can be set for each connection. 10 to 10,000 ms in 1-ms increments	1.02 or earlier	
		Allowed communications bandwidth per Unit		3,000 pps*6 (including heartbeat)	1.03 or later	---
				1,000 pps (including heartbeat)	1.02 or earlier	
	Number of TCP sockets			30	1.03 or later	---
				16	1.02 or earlier	

Function		Performance value	Unit version	Sysmac Studio version
Built-in EtherCAT port	Communications cycle*7 (NJ301-□□□□)	500, 1000, 2000, or 4000 μs	1.03 or later	---
		1,000, 2,000, or 4,000 μs	1.02 or earlier	

*1. This performance improvement applies only to an NJ301-□□□□ CPU Unit. The maximum number of variables with a Retain attributes for the NJ501-□□□□ is 10,000.
*2. This is the total for all axis types.
*3. This performance improvement applies only to an NJ301-□□□□ CPU Unit. The maximum number of controlled axes for the NJ501-□□□□ are as follows:
NJ501-□5□□: 64 axes, NJ501-□4□□: 32 axes, and NJ501-□3□□: 16 axes
*4. There is no change in the maximum number of used real axes.
*5. This performance improvement applies only to an NJ301-□□□□ CPU Unit. The maximum number of axes for single-axis control for the NJ501-□□□□ are as follows:
NJ501-□5□□: 64 axes, NJ501-□4□□: 32 axes, and NJ501-□3□□: 16 axes
*6. Here, pps means “packets per second” and indicates the number of packets that can be processed in one second.
*7. This performance improvement applies only to an NJ301-□□□□ CPU Unit. You can use 500, 1,000, 2,000 or 4,000 μs communications cycle with an NJ501-□□□□ CPU Unit, and 1,000, 2,000 or 4,000 μs communications cycle with an NJ101-□□□□ CPU Unit.

I

Index

Index

Numerics

_<TaskName>_Exceeded (Task Period Exceeded Flag)..... 5-125
_AlarmFlag (User-defined Error Status)..... 8-93, A-93, A-166
_APB_LogStatus.SettingName (Variable Log Output Setting Name)..... A-111, A-195
_APB_LogStatus.SettingType (Variable Log Output Setting Type)..... A-111
_APB_LogStatus.SettingType(Variable Log Output Setting Type)..... A-195
_APB_LogStatus.Status (Variable Log Output Setting Status)..... A-111, A-195
_APB_Status (APB Service Status)..... A-110
_APB_Status.ActiveSamplingSettingName (Active Sampling Setting Name)..... A-110, A-194
_APB_Status.ActiveSamplingSettingNumber (Active Sampling Setting Number)..... A-110, A-194
_APB_Status.IsProjectBackupFinished (APB Project Backup Finished)..... A-110, A-194
_APB_Status.LatestErrorID (APB Instruction Error Code) A-110, A-194
_APB_Status.LatestEventCode (APB Service Event Code).. A-110, A-195
_APB_Status.LatestLogName (Latest Variable Log File Name)..... A-110, A-193
_APB_Status.LatestLogTime (Latest Variable Log Output Date and Time)..... A-110, A-194
_APB_Status.Status (APB Service Operating Status)..... A-110, A-193
_BackupBusy (Backup Function Busy Flag)..... A-99, A-178
_Card1Access (SD Memory Card Access Flag)..... 8-36, A-94, A-167
_Card1BkupCmd (SD Memory Card Backup Commands)..... A-94, A-168
_Card1BkupSta (SD Memory Card Backup Status)..... A-95, A-170
_Card1Capacity (SD Memory Card Storage Capacity)..... 8-36, A-94, A-168
_Card1Deteriorated (SD Memory Card Life Warning Flag).... 8-35, 8-36, A-94, A-167
_Card1Err (SD Memory Card Error Flag)..... 8-31, 8-36, A-94, A-167
_Card1PowerFail (SD Memory Card Power Interruption Flag)..... 8-36, A-94, A-168
_Card1PrgTransferCmd (SD Memory Card Program Transfer Command)..... A-96, A-171
_Card1PrgTransferSta (SD Memory Card Program Transfer Status)..... A-97, A-173
_Card1Protect (SD Memory Card Write Protected Flag)..... 8-36, A-94, A-167
_Card1Ready (SD Memory Card Ready Flag)..... 8-36, A-94, A-166
_Card1RestoreCmd (SD Memory Card Restore Command). A-97, A-174
_Card1RestoreCmdTargetAbsEncoder (Absolute Encoder Home Offset Transfer Flag)..... A-98, A-177
_Card1RestoreCmdTargetIPAdr (IP Address Transfer Flag).. A-98, A-176
_Card1RestoreCmdTargetMemory (Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag)..... A-98, A-176
_Card1RestoreCmdTargetOpcuaSPF (OPCUA Security Profile Transfer Flag)..... A-98, A-177
_Card1RestoreCmdTargetUnitConfig (Unit and Slave Parameters Transfer Flag)..... A-98, A-176
_Card1RestoreCmdTargetUserProgram (User Program and Settings Transfer Flag)..... A-98, A-175
_Card1RestoreCmdTargetVariable (Present Values of Variables with the Retain Attribute Transfer Flag).... A-98, A-176
_Card1RestoreCmdTargetXBusUnitConfig (X Bus Unit Settings Transfer Flag)..... A-98, A-177
_Card1RestoreCmdTargetXBusUnitIPAdr (IP Address Settings Transfer Flag of EtherNet/IP Port on X Bus Unit)..... A-98, A-177
_Card1RestoreSta (SD Memory Card Restore Status)..... A-98, A-175
_Card1Used (SD Memory Card Storage Usage)..... 8-37, A-94, A-168
_Card1VefySta (SD Memory Card Verify Status) A-96, A-170
_CJB_CBU□□InitSta (CPU Bus Unit Initializing Flags)..... 8-29, A-109, A-191
_CJB_CBU□□Restart (CPU Bus Unit Restart Bits)..... 8-29, A-109, A-192
_CJB_ErrSta (I/O Bus Error Status)..... A-108, A-190
_CJB_InRespTm (Basic Input Unit Input Response Times)... 8-28, A-108, A-191
_CJB_IOUnitInfo (Basic I/O Unit Information)... A-109, A-191
_CJB_MaxRackNo (Largest Rack Number)..... A-107, A-189
_CJB_MaxSlotNo (Largest Slot Number)..... A-107, A-189
_CJB_MstrErrSta (I/O Bus Master Error Status)..... A-108, A-190
_CJB_SCU00P□ChgSta (Serial Communications Unit 0, Port 1/2 Settings Changing Flags)..... A-110, A-193
_CJB_SCU15P□ChgSta (Serial Communications Units 1 to 15, Port 1/2 Settings Changing Flags)..... A-110, A-193
_CJB_SIO□□InitSta (Special I/O Unit Initializing Flags)..... 8-29, A-109, A-192
_CJB_SIO□□Restart (Special I/O Unit Restart Bits)..... 8-29, A-109, A-192
_CJB_UnitErrSta (I/O Bus Unit Error Status).... A-108, A-190
_CurrentTime (System Time)..... 8-5, A-92, A-163
_DeviceOutHoldCfg (Device Output Hold Configuration)..... A-102, A-183
_DeviceOutHoldStatus (Device Output Hold Status)..... A-102, A-183
_EC_CommErrTbl (Communications Error Slave Table)..... A-122, A-209
_EC_CycleExceeded (EtherCAT Communications Cycle Exceeded)..... A-122, A-209

_EC_DisableSlavTbl (Disabled Slave Table)..... A-124, A-211
_EC_DisconnSlavTbl (Disconnected Slave Table)..... A-124, A-211
_EC_EntrySlavTbl (Network Connected Slave Table)..... A-124, A-210
_EC_ErrSta (EtherCAT Error)..... A-120, A-205
_EC_InData1Invalid (Input Data1 Invalid)..... A-125, A-212
_EC_InData2Invalid (Input Data2 Invalid)..... A-125, A-213
_EC_InDataInvalid (Input Data Invalid)..... A-125, A-212
_EC_IndataInvalidErr (Input Process Data Invalid Error)..... A-122, A-209
_EC_LanHwErr (Communications Controller Error)..... A-120, A-206
_EC_LinkOffErr (Link OFF Error)..... A-120, A-206
_EC_LinkStatus (Link Status)..... A-124, A-212
_EC_MacAdrErr (MAC Address Error)..... A-120, A-206
_EC_MBXSlavTbl (Message Communications Enabled Slave Table)..... A-124, A-210
_EC_MsgErr (EtherCAT Message Error)..... A-121, A-208
_EC_MstrErr (Master Error)..... A-120, A-205
_EC_NetCfgCmpErr (Network Configuration Verification Error)..... A-121, A-207
_EC_NetCfgErr (Network Configuration Information Error).... A-120, A-206
_EC_NetTopologyErr (Network Configuration Error)..... A-121, A-207
_EC_PDActive (Process Data Communications Status)..... A-124, A-211
_EC_PDCommErr (Process Data Communications Error)..... A-121, A-207
_EC_PDSendErr (Process Data Transmission Error)..... A-121, A-207
_EC_PDSlavTbl (Process Data Communicating Slave Table)..... A-124, A-210
_EC_PDTimeoutErr (Process Data Reception Timeout Error)..... A-121, A-207
_EC_PktMonStop (Packet Monitoring Stopped) A-124, A-211
_EC_PktSaving (Saving Packet Data File)..... A-125, A-212
_EC_PortErr (Communications Port Error)..... A-120, A-205
_EC_RegSlavTbl (Registered Slave Table)..... A-124, A-209
_EC_Ring Breaking (Ring Disconnection)..... A-125
_EC_RingBreaking (Ring Disconnection)..... A-213
_EC_RingBreakNodeAdr (Slave Node Address Before Ring Disconnection)..... A-125, A-213
_EC_SlavAdrDupErr (Slave Node Address Duplicated Error) A-121, A-208
_EC_SlavAppErr (Slave Application Error)..... A-121, A-208
_EC_SlavEmergErr (Emergency Message Detected)..... A-121, A-208
_EC_SlavErr (Slave Error)..... A-120, A-205
_EC_SlavErrTbl (Slave Error Table)..... A-120, A-206
_EC_SlavInitErr (Slave Initialization Error)..... A-121, A-208
_EC_StatisticsLogBusy (Diagnosis/Statistics Log Busy)..... A-126, A-214
_EC_StatisticsLogCycleSec (Diagnosis/Statistics Log Cycle) A-126, A-214
_EC_StatisticsLogEnable (Diagnosis/Statistics Log Enable). A-125, A-213
_EC_StatisticsLogErr (Diagnosis/Statistics Log Error)..... A-126, A-214
_EIP_BootpErr (BOOTP Server Error)..... A-137, A-223
_EIP_CipErr (CIP Communications Error)..... A-131, A-217
_EIP_DhcpErr (DHCP Server Error)..... A-138, A-223
_EIP_DNSCfgErr (DNS Setting Error)..... A-137, A-222
_EIP_DNSSrvErr (DNS Server Connection Error)..... A-144, A-230
_EIP_ErrSta (EtherNet/IP Error)..... A-127, A-215
_EIP_EstbTargetSta (Normal Target Node Information)..... A-150, A-234
_EIP_EtnCfgErr (Basic Ethernet Setting Error). A-135, A-220
_EIP_EtnOnlineSta (Online)..... A-147, A-231
_EIP_IdentityErr (Identity Error)..... A-138, A-224
_EIP_IPAdrCfgErr (IP Address Setting Error).... A-135, A-221
_EIP_IPAdrDupErr (IP Address Duplication Error)..... A-136, A-222
_EIP_IPRTblErr (IP Route Table Error)..... A-138, A-224
_EIP_LanHwErr (Communications Controller Error)..... A-134, A-219
_EIP_MacAdrErr (MAC Address Error)..... A-133, A-218
_EIP_MultiSwONErr (Multiple Switches ON Error)..... A-143, A-229
_EIP_NTPResult (NTP Operation Information). A-157, A-238
_EIP_NTPResult.ExecNormal (NTP Operation Result)..... A-158, A-239
_EIP_NTPResult.ExecTime (NTP Last Operation Time)..... A-158, A-239
_EIP_NTPSrvErr (NTP Server Connection Error)..... A-144, A-230
_EIP_PortErr (Communications Port Error)..... A-128, A-215
_EIP_RegTargetSta (Registered Target Node Information).... A-149, A-233
_EIP_TagAdrErr (Tag Name Resolution Error).. A-142, A-228
_EIP_TargetNodeErr (Target Node Error Information)..... A-155, A-237
_EIP_TargetPLCErr (Target PLC Error Information)..... A-153, A-236
_EIP_TargetPLCModeSta (Target PLC Operating Mode)..... A-151, A-235
_EIP_TcpAppCfgErr (TCP Application Setting Error)..... A-144, A-230
_EIP_TcpAppErr (TCP Application Communications Error)... A-133, A-218
_EIP_TDLinkAllRunSta (All Tag Data Link Communications Status)..... A-148, A-232
_EIP_TDLinkCfgErr (Tag Data Link Setting Error)..... A-139, A-225
_EIP_TDLinkErr (Tag Data Link Communications Error)..... A-141, A-227
_EIP_TDLinkOpnErr (Tag Data Link Connection Failed)..... A-140, A-226
_EIP_TDLinkRunSta (Tag Data Link Communications Status)..... A-148, A-232
_EIP_TDLinkStartCmd (Tag Data Link Communications Start Switch)..... A-160, A-239
_EIP_TDLinkStopCmd (Tag Data Link Communications Stop Switch)..... A-160, A-240
_EIP1_BootpErr (Port1 BOOTP Server Error).. A-137, A-223

_EIP1_CipErr (CIP Communications1 Error).... A-132, A-217
_EIP1_DhcpErr (Port1 DHCP Server Error)..... A-138, A-224
_EIP1_EstbTargetSta (CIP Communications1 Normal Target Node Information)..... A-150, A-234
_EIP1_EtnCfgErr (Port1 Basic Ethernet Setting Error)..... A-135, A-220
_EIP1_EtnOnlineSta (Port1 Online)..... A-147, A-231
_EIP1_IdentityErr (CIP Communications1 Identity Error)..... A-139, A-225
_EIP1_IPAdrCfgErr (Port1 IP Address Setting Error)..... A-136, A-221
_EIP1_IPAdrDupErr (Port1 IP Address Duplication Error)..... A-136, A-222
_EIP1_LanHwErr (Port1 Communications Controller Error)... A-134, A-219
_EIP1_MacAdrErr (Port1 MAC Address Error). A-134, A-219
_EIP1_MultiSwONErr (CIP Communications1 Multiple Switches ON Error)..... A-143, A-229
_EIP1_PortErr (Communications Port1 Error)...A-129, A-216
_EIP1_RegTargetSta (CIP Communications1 Registered Target Node Information)..... A-149, A-233
_EIP1_TagAdrErr (CIP Communications1 Tag Name Resolution Error)..... A-142, A-228
_EIP1_TargetNodeErr (CIP Communications1 Target Node Error Information)..... A-156, A-238
_EIP1_TargetPLCErr (CIP Communications1 Target PLC Error Information)..... A-153, A-236
_EIP1_TargetPLCModeSta (CIP Communications1 Target PLC Operating Mode)..... A-152, A-235
_EIP1_TDLinkAllRunSta (CIP Communications1 All Tag Data Link Communications Status)..... A-148, A-233
_EIP1_TDLinkCfgErr (CIP Communications1 Tag Data Link Setting Error)..... A-139, A-225
_EIP1_TDLinkErr (CIP Communications1 Tag Data Link Communications Error)..... A-141, A-227
_EIP1_TDLinkOpnErr (CIP Communications1 Tag Data Link Connection Failed)..... A-140, A-226
_EIP1_TDLinkRunSta (CIP Communications1 Tag Data Link Communications Status)..... A-148, A-232
_EIP1_TDLinkStartCmd (CIP Communications1 Tag Data Link Communications Start Switch)..... A-160, A-239
_EIP1_TDLinkStopCmd (CIP Communications1 Tag Data Link Communications Stop Switch)..... A-161, A-240
_EIP2_BootpErr (Port2 BOOTP Server Error).. A-137, A-223
_EIP2_CipErr (CIP Communications2 Error).... A-133, A-218
_EIP2_DhcpErr (Port2 DHCP Server Error)..... A-138, A-224
_EIP2_EstbTargetSta (CIP Communications2 Normal Target Node Information)..... A-151, A-235
_EIP2_EtnCfgErr (Port2 Basic Ethernet Setting Error)..... A-135, A-220
_EIP2_EtnOnlineSta (Port2 Online)..... A-147, A-231
_EIP2_IdentityErr (CIP Communications2 Identity Error)..... A-139, A-225
_EIP2_IPAdrCfgErr (Port2 IP Address Setting Error)..... A-136, A-221
_EIP2_IPAdrDupErr (Port2 IP Address Duplication Error)..... A-136, A-222
_EIP2_LanHwErr (Port2 Communications Controller Error)... A-134, A-220
_EIP2_MacAdrErr (Port2 MAC Address Error). A-134, A-219
_EIP2_MultiSwONErr (CIP Communications2 Multiple Switches ON Error)..... A-144, A-230
_EIP2_PortErr (Communications Port2 Error)...A-130, A-216
_EIP2_RegTargetSta (CIP Communications2 Registered Target Node Information)..... A-150, A-234
_EIP2_TagAdrErr (CIP Communications2 Tag Name Resolution Error)..... A-143, A-229
_EIP2_TargetNodeErr (CIP Communications2 Target Node Error Information)..... A-157, A-238
_EIP2_TargetPLCErr (CIP Communications2 Target PLC Error Information)..... A-154, A-237
_EIP2_TargetPLCModeSta (CIP Communications2 Target PLC Operating Mode)..... A-152, A-236
_EIP2_TDLinkAllRunSta (CIP Communications2 All Tag Data Link Communications Status)..... A-149, A-233
_EIP2_TDLinkCfgErr (CIP Communications2 Tag Data Link Setting Error)..... A-139, A-226
_EIP2_TDLinkErr (CIP Communications2 Tag Data Link Communications Error)..... A-141, A-227
_EIP2_TDLinkOpnErr (CIP Communications2 Tag Data Link Connection Failed)..... A-141, A-227
_EIP2_TDLinkRunSta (CIP Communications2 Tag Data Link Communications Status)..... A-148, A-232
_EIP2_TDLinkStartCmd (CIP Communications2 Tag Data Link Communications Start Switch)..... A-160, A-240
_EIP2_TDLinkStopCmd (CIP Communications2 Tag Data Link Communications Stop Switch)..... A-161, A-240
_ErrSta (Controller Error Status)..... 5-125, A-93, A-166
_FINSTCPConnSta (FINS/TCP Connection Status)..... A-100, A-181
_HardwareRevision (Hardware Revision)..... A-101, A-181
_MC_AX (Axis Variables)..... A-118, A-204
_MC_AX_ErrSta (Axis Error Status)..... A-117, A-202
_MC_COM (Common Variable)..... A-117, A-202
_MC_ComErrSta (Common Error Status)..... A-117, A-201
_MC_ErrSta (Motion Control Function Module Error Status). A-117, A-201
_MC_GRP (Axes Group Variables)..... A-117, A-203
_MC_GRP_ErrSta (Axes Group Error Status)...A-117, A-202
_MC1_AX (Axis Variables)..... A-119, A-204
_MC1_GRP (Axes Group Variables)..... A-118, A-203
_MC2_AX (Axis Variables)..... A-119, A-204
_MC2_GRP (Axes Group Variables)..... A-118, A-203
_NXB_ErrSta (NX Bus Function Module Error Status)..... A-113, A-198
_NXB_MaxUnitNo (Largest Unit Number)..... A-111, A-196
_NXB_MstrErrSta (NX Bus Function Module Master Error Status)..... A-113, A-198
_NXB_UnitErrFlagTbl (NX Unit Error Status).... A-114, A-199
_NXB_UnitErrStaTbl (NX Bus Function Module Unit Error Status)..... A-114, A-199
_NXB_UnitIOActiveTbl (NX Unit I/O Data Active Status)..... A-111, A-196
_NXB_UnitMsgActiveTbl (NX Unit Message Enabled Status) A-112, A-197
_NXB_UnitRegTbl (NX Unit Registration Status)..... A-112, A-197

_PLC_ErrSta (PLC Function Module Error Status)..... A-103, A-184
_PLC_OptBoardSta (Option Board Status)..... A-103, A-185
_PLC_OptBoardSta.Error (Option Board Error) A-103, A-185
_PLC_OptBoardSta.isDetect (Option Board Mounted)..... A-103, A-185
_PLC_OptBoardSta.Run (Option Board Normal Operation).. A-103, A-185
_PLC_OptSerialErrSta (Serial Option Board Error Status)..... A-104, A-186
_PLC_OptSerialErrSta.Error (Transmission Error)..... A-104, A-186
_PLC_OptSerialErrSta.FramingErr (Framing Error)..... A-105, A-187
_PLC_OptSerialErrSta.OverRun (Overrun Error)..... A-106, A-188
_PLC_OptSerialErrSta.ParityErr (Parity Error).. A-105, A-187
_PLC_SFLogSta (Safety Data Logging Status) A-107, A-188
_PLC_SFLogSta.IsComplete (Safety Data Logging Completed Flag)..... A-107, A-189
_PLC_SFLogSta.IsOutput (Log File Output Completed Flag) A-107, A-189
_PLC_SFLogSta.IsStart (Safety Data Logging Busy Flag).... A-107, A-188
_PLC_TraceSta.IsComplete (Trace Completed Flag)..... 8-75, A-102, A-184
_PLC_TraceSta.IsStart (Trace Busy Flag)..... 8-75, A-102, A-183
_PLC_TraceSta.IsTrigger (Trace Trigger Monitor Flag)..... 8-75, A-102, A-184
_PLC_TraceSta.ParamErr (Trace Parameter Error Flag)..... 8-75, A-102, A-184
_Port_isAvailable (Network Communications Instruction Enabled Flag)..... A-100, A-180
_Port_numUsingPort (Number of Used Ports).. A-100, A-180
_PowerOnCount (Power Interruption Count).....A-99, A-178
_PowerOnHour (Total Power ON Time)..... A-99, A-178
_ProjectUnitVersion (Project Unit Version)..... A-101, A-181
_RetainFail (Retention Failure Flag).....A-99, A-179
_SelfTest_HighTemperature (CPU Unit High Temperature Flag).....A-101, A-182
_SelfTest_LowBattery (Low Battery Flag)..... A-101, A-182
_SelfTest_LowFanRevolution (Low FAN Revolution Flag)..... A-101, A-182
_StorageDeteriorated (Non-volatile Memory Life Warning Flag).....A-101, A-182
_TaskName_Active (Task Active Flag)..... 5-8, A-92, A-164
_TaskName_ExceedCount (Task Period Exceeded Count)... 5-9, A-93, A-165
_TaskName_Exceeded (Task Period Exceeded Flag)..... 5-9, 5-131, A-93, A-165
_TaskName_Exceeded Count (Task Period Exceeded Count)..... 5-125
_TaskName_ExecCount (Task Execution Count)..... 5-9, 5-131, A-93, A-165
_TaskName_LastExecTime (Last Task Execution Time)..... 5-8, A-92, A-164
_TaskName_MaxExecTime (Maximum Task Execution Time).....5-8, A-92, A-164
_TaskName_MinExecTime (Minimum Task Execution Time). 5-8, A-92, A-164
_UnitVersion (Unit Version)..... A-101, A-181
_XBU_ErrSta (X Bus Function Module Error Status)..... A-115, A-200
_XBU_MstrErr (X Bus Function Module Master Error Status) A-116, A-200
_XBU_RegTbl (X Bus Unit Registration Status) A-115, A-200
_XBU_UnitErr (X Bus Function Module Unit Error Status)..... A-116, A-200
_XBU_UnitErrTbl (X Bus Function Module Unit Error Status Table).....A-116, A-201

A

Absolute Encoder Home Offset Transfer Flag.....A-98, A-177
accessing tasks.....5-114
Active Sampling Setting Name..... A-110, A-194
Active Sampling Setting Number.....A-110, A-194
All Tag Data Link Communications Status.....A-148, A-232
Always FALSE Flag.....A-99, A-179
Always TRUE Flag..... A-99, A-179
APB Instruction Error Code.....A-110, A-194
APB Project Backup Finished.....A-110, A-194
APB Service Event Code.....A-110, A-195
APB Service Operating Status.....A-110, A-193
APB Service Status..... A-110
APS Log Output Status..... A-110
Axes Group Error Status.....A-117, A-202
Axes Group Variables.....A-117, A-118, A-203
Axis Error Status.....A-117, A-202
Axis Variables..... A-118, A-119, A-204

B

Backing up data.....9-3
Backup Function Busy Flag.....A-99, A-178
Basic Ethernet Setting Error.....A-135, A-220
Basic I/O Unit Information.....A-109, A-191
Basic Input Unit Input Response Times... 8-28, A-108, A-191
Bit strings.....6-33, 6-45
BOOL..... 6-33, 6-45
Boolean..... 6-33, 6-45
BOOTP Server Error..... A-137, A-223
BYTE.....6-33, 6-45

C

Carry Flag.....6-8, A-99, A-179
CIP Communications Error.....A-131, A-217
CIP Communications1 All Tag Data Link Communications Status.....A-148, A-233
CIP Communications1 Error.....A-132, A-217
CIP Communications1 Identity Error..... A-139, A-225
CIP Communications1 Multiple Switches ON Error..... A-143, A-229
CIP Communications1 Normal Target Node Information..... A-150, A-234
CIP Communications1 Registered Target Node Information.. A-149, A-233

CIP Communications1 Tag Data Link Communications Error A-141, A-227
CIP Communications1 Tag Data Link Communications Start Switch..... A-160, A-239
CIP Communications1 Tag Data Link Communications Status..... A-148, A-232
CIP Communications1 Tag Data Link Communications Stop Switch..... A-161, A-240
CIP Communications1 Tag Data Link Connection Failed..... A-140, A-226
CIP Communications1 Tag Data Link Setting Error..... A-139, A-225
CIP Communications1 Tag Name Resolution Error..... A-142, A-228
CIP Communications1 Target Node Error Information..... A-156, A-238
CIP Communications1 Target PLC Error Information..... A-153, A-236
CIP Communications1 Target PLC Operating Mode..... A-152, A-235
CIP Communications2 All Tag Data Link Communications Status..... A-149, A-233
CIP Communications2 Error..... A-133, A-218
CIP Communications2 Identity Error..... A-139, A-225
CIP Communications2 Multiple Switches ON Error..... A-144, A-230
CIP Communications2 Normal Target Node Information..... A-151, A-235
CIP Communications2 Registered Target Node Information.. A-150, A-234
CIP Communications2 Tag Data Link Communications Error A-141, A-227
CIP Communications2 Tag Data Link Communications Start Switch..... A-160, A-240
CIP Communications2 Tag Data Link Communications Status..... A-148, A-232
CIP Communications2 Tag Data Link Communications Stop Switch..... A-161, A-240
CIP Communications2 Tag Data Link Connection Failed..... A-141, A-227
CIP Communications2 Tag Data Link Setting Error..... A-139, A-226
CIP Communications2 Tag Name Resolution Error..... A-143, A-229
CIP Communications2 Target Node Error Information..... A-157, A-238
CIP Communications2 Target PLC Error Information..... A-154, A-237
CIP Communications2 Target PLC Operating Mode..... A-152, A-236
CIP Safety communications..... 2-10
CIP Safety routing..... A-296
CJ-series Unit configuration..... 1-15
Common Error Status..... A-117, A-201
Common Variable..... A-117, A-202
Communications Controller Error..... A-120, A-134, A-206, A-219
Communications Error Slave Table..... A-122, A-209
Communications Port Error..... A-120, A-128, A-205, A-215
Communications Port1 Error..... A-129, A-216
Communications Port2 Error..... A-130, A-216
Controller Error Status..... A-93, A-166
CPU Bus Unit Initializing Flags..... 8-29, A-109, A-191
CPU Bus Unit Restart Bits..... 8-29, A-109, A-192
CPU Unit High Temperature Flag..... A-101, A-182

D

Date..... 6-34, 6-46
DATE..... 6-34, 6-46
Date and time..... 6-34, 6-46
DATE_ AND_ TIME..... 6-34, 6-46
Derivative data types..... 6-36
Device Output Hold Configuration..... A-102, A-183
Device Output Hold Status..... A-102, A-183
DHCP Server Error..... A-138, A-223
Diagnosis/Statistics Log Busy..... A-126, A-214
Diagnosis/Statistics Log Cycle..... A-126, A-214
Diagnosis/Statistics Log Enable..... A-125, A-213
Diagnosis/Statistics Log Error..... A-126, A-214
DINT..... 6-33
Disabled Slave Table..... A-124, A-211
Disconnected Slave Table..... A-124, A-211
DNS Server Connection Error..... A-144, A-230
DNS Setting Error..... A-137, A-222
Durations..... 6-34, 6-46
DWORD..... 6-33

E

Emergency Message Detected..... A-121, A-208
EN..... 6-20
ENO..... 6-20
EtherCAT Communications Cycle Exceeded.... A-122, A-209
EtherCAT Error..... A-120, A-205
EtherCAT Message Error..... A-121, A-208
EtherCAT network configuration..... 1-5, 1-7, 1-10, 1-12, 1-14
EtherNet/IP Error..... A-127, A-215
Event Log Settings..... 4-6
Event task..... 5-12, 5-69
External variables..... 6-20

F

FINS/TCP Connection Status..... A-100, A-181
First Program Period Flag..... 6-8, A-100, A-180
First RUN Period Flag..... 6-8, A-100, A-179
Framing Error..... A-105, A-187
Function module: EtherCAT Master Function Module..... 2-4
Function module: EtherNet/IP Function Module..... 2-4
Function module: Motion Control Function Module..... 2-4
Function module: NX Bus Function Module..... 2-4
Function module: PLC Function Module..... 2-4
Function module: X Bus Function Module..... 2-4

H

Hardware Revision..... A-101, A-181, A-289, A-292
HardwareRevision (Hardware Revision)..... A-289, A-292

I

I/O Bus Error Status..... A-108, A-190
I/O Bus Master Error Status..... A-108, A-190
I/O Bus Unit Error Status..... A-108, A-190
Identity Error..... A-138, A-224
In-out variables..... 6-20, 6-94
Initial Status of Program..... 4-11
Input Data Invalid..... A-125, A-212
Input Data1 Invalid..... A-125, A-212
Input Data2 Invalid..... A-125, A-213
Input Process Data Invalid Error..... A-122, A-209
Input variables..... 6-19
Instruction Error Flag..... 6-8, 6-140, A-100, A-180
INT..... 6-33
Integers..... 6-33, 6-45
Internal variables..... 6-20, 6-36
IP Address Duplication Error..... A-136, A-222
IP Address Setting Error..... A-135, A-221
IP Address Settings Transfer Flag of EtherNet/IP Port on X Bus Unit..... A-98, A-177
IP Address Transfer Flag..... A-98, A-176
IP Route Table Error..... A-138, A-224

L

Largest Rack Number..... A-107, A-189
Largest Slot Number..... A-107, A-189
Largest Unit Number..... A-111, A-196
Last Task Execution Time..... 5-8, A-92, A-164
Latest Variable Log File Name..... A-110, A-193
Latest Variable Log Output Date and Time..... A-110, A-194
Link OFF Error..... A-120, A-206
Link Status..... A-124, A-212
LINT..... 6-33
Log File Output Completed Flag..... A-107, A-189
Low Battery Flag..... A-101, A-182
Low FAN Revolution Flag..... A-101, A-182
LREAL..... 6-33
LWORD..... 6-33

M

MAC Address Error..... A-120, A-133, A-206, A-218
Master Error..... A-120, A-205
Maximum Task Execution Time..... 5-8, A-92, A-164
Memory settings for CJ-series Units..... 4-35, 4-37
Message Communications Enabled Slave Table..... A-124, A-210
Minimum Task Execution Time..... 5-8, A-92, A-164
Motion Control Function Module Error Status..... A-117, A-201
Multiple Switches ON Error..... A-143, A-229

N

Network Communications Instruction Enabled Flag..... A-100, A-180
Network Configuration Error..... A-121, A-207
Network Configuration Information Error..... A-120, A-206
Network Configuration Verification Error..... A-121, A-207
Network Connected Slave Table..... A-124, A-210
Non-volatile Memory Life Warning Flag..... A-101, A-182, A-290, A-292
Normal Target Node Information..... A-150, A-234
NTP Last Operation Time..... A-158, A-239
NTP Operation Information..... A-157, A-238
NTP Operation Result..... A-158, A-239
NTP Server Connection Error..... A-144, A-230
Number of Used Ports..... A-100, A-180
NX Bus Function Module Error Status..... A-113, A-198
NX Bus Function Module Master Error Status..... A-113, A-198
NX Bus Function Module Unit Error Status..... A-114, A-199
NX Unit configuration..... 1-8, 1-10, 1-13
NX Unit Error Status..... A-114, A-199
NX Unit I/O Data Active Status..... A-111, A-196
NX Unit Message Enabled Status..... A-112, A-197
NX Unit Registration Status..... A-112, A-197

O

Online..... A-147, A-231
OPCUA Security Profile Transfer Flag..... A-98, A-177
Operation Settings..... 4-5
Option Board Error..... A-103, A-185
Option Board Mounted..... A-103, A-185
Option Board Normal Operation..... A-103, A-185
Option Board Status..... A-103, A-185
Output variables..... 6-19
Overrun Error..... A-106, A-188

P

P_CY (Carry Flag)..... 6-8, 6-136, A-99, A-179
P_First_Run (First Program Period Flag)..... 6-8, A-100, A-180
P_First_RunMode (First RUN Period Flag) 6-8, A-100, A-179
P_Off (Always FALSE Flag)..... A-99, A-179
P_On (Always TRUE Flag)..... A-99, A-179
P_PRGER (Instruction Error Flag).. 6-8, 6-140, A-100, A-180
Packet Monitoring Stopped..... A-124, A-211
Parity Error..... A-105, A-187
Period/Execution Conditions..... 4-9
PLC Function Module Error Status..... A-103, A-184
Port1 Basic Ethernet Setting Error..... A-135, A-220
Port1 BOOTP Server Error..... A-137, A-223
Port1 Communications Controller Error..... A-134, A-219
Port1 DHCP Server Error..... A-138, A-224
Port1 IP Address Duplication Error..... A-136, A-222
Port1 IP Address Setting Error..... A-136, A-221
Port1 MAC Address Error..... A-134, A-219
Port1 Online..... A-147, A-231
Port2 Basic Ethernet Setting Error..... A-135, A-220
Port2 BOOTP Server Error..... A-137, A-223
Port2 Communications Controller Error..... A-134, A-220
Port2 DHCP Server Error..... A-138, A-224
Port2 IP Address Duplication Error..... A-136, A-222
Port2 IP Address Setting Error..... A-136, A-221
Port2 MAC Address Error..... A-134, A-219
Port2 Online..... A-147, A-231
Power Interruption Count..... A-99, A-178, A-289, A-291
PowerOnCount (Power Interruption Count)..... A-289, A-291

PowerOnHour (Total Power ON Time)..... A-289, A-291
Present Values of Memory Used for CJ-series Units with the Retain Attribute Transfer Flag..... A-98, A-176
Present Values of Variables with the Retain Attribute Transfer Flag..... A-98, A-176
Process Data Communicating Slave Table..... A-124, A-210
Process Data Communications Error..... A-121, A-207
Process Data Communications Status..... A-124, A-211
Process Data Reception Timeout Error..... A-121, A-207
Process Data Transmission Error..... A-121, A-207
Program Execution Order..... 4-11
PROGRAM mode..... 2-43, 2-45
Project Unit Version..... A-101, A-181, A-290, A-292
ProjectUnitVersion (Project Unit Version)..... A-290, A-292

R

REAL..... 6-33, 6-46
Real numbers..... 6-33, 6-46
real processing time of task..... 5-132
refreshing tasks..... 5-114
Registered Slave Table..... A-124, A-209
Registered Target Node Information..... A-149, A-233
Restoring data..... 9-3
Retained..... 6-22
Retention Failure Flag..... A-99, A-179
Return value..... 6-20
Ring Disconnection..... A-125, A-213
RUN mode..... 2-43, 2-45
RUN output..... 8-6

S

Safety Data Logging Busy Flag..... A-107, A-188
Safety Data Logging Completed Flag..... A-107, A-189
Safety Data Logging Status..... A-107, A-188
Saving Packet Data File..... A-125, A-212
SD Memory Card Access Flag..... 8-36, A-94, A-167
SD Memory Card Backup Commands..... A-94, A-168
SD Memory Card Backup Status..... A-95, A-170
SD Memory Card Error Flag..... 8-31, 8-36, A-94, A-167
SD Memory Card Life Warning Flag. 8-35, 8-36, A-94, A-167
SD Memory Card Power Interruption Flag. 8-36, A-94, A-168
SD Memory Card Program Transfer Command.. A-96, A-171
SD Memory Card Program Transfer Setting..... 4-5
SD Memory Card Program Transfer Status..... A-97, A-173
SD Memory Card Ready Flag..... 8-36, A-94, A-166
SD Memory Card Restore Command..... A-97, A-174
SD Memory Card Restore Setting..... 4-6
SD Memory Card Restore Status..... A-98
SD Memory Card Settings..... 4-5
SD Memory Card Storage Capacity..... 8-36, A-94, A-168
SD Memory Card Storage Usage..... 8-37, A-94, A-168
SD Memory Card Verify Status..... A-96, A-170
SD Memory Card Write Protected Flag..... 8-36, A-94, A-167
Secure Communication..... 10-12
Security Settings..... 4-6
Semi-user-defined variables..... 2-13
Sequence control and motion control..... 2-2
Serial Communications Unit 0, Port 1/2 Settings Changing Flags..... A-110, A-193
Serial Communications Units 1 to 15, Port 1/2 Settings Changing Flags..... A-110, A-193
Serial Option Board Error Status..... A-104, A-186
Setting Change during RUN Mode..... 4-7
Setting debug programs..... 7-6
Setting simulation programs..... 7-6
simulation..... 7-3
SINT..... 6-33, 6-45
Slave Application Error..... A-121, A-208
Slave Error..... A-120, A-205
Slave Error Table..... A-120, A-206
Slave Initialization Error..... A-121, A-208
Slave Node Address Before Ring Disconnection..... A-125, A-213
Slave Node Address Duplicated Error..... A-121, A-208
Special I/O Unit Initializing Flags..... 8-29, A-109, A-192
Special I/O Unit Restart Bits..... 8-29, A-109, A-192
Start and stop the Simulator..... 7-3
StorageDeteriorated (Non-volatile Memory Life Warning Flag)..... A-290, A-292
STRING..... 6-34, 6-46
Support Software..... 1-5, 1-8, 1-10, 1-13, 1-15
System Service Monitoring Settings..... 4-5
System Time..... A-92, A-163
System-defined variables..... 2-13
system-defined variables for motion control..... 2-16

T

Tag Data Link Communications Error..... A-141, A-227
Tag Data Link Communications Start Switch..... A-160, A-239
Tag Data Link Communications Status..... A-148, A-232
Tag Data Link Communications Stop Switch..... A-160, A-240
Tag Data Link Connection Failed..... A-140, A-226
Tag Data Link Setting Error..... A-139, A-225
Tag Name Resolution Error..... A-142, A-228
Target Node Error Information..... A-155, A-237
Target PLC Error Information..... A-153, A-236
Target PLC Operating Mode..... A-151, A-235
Task Active Flag..... 5-8, A-92, A-164
Task Execution Count..... 5-9, 5-131, A-93, A-165
task execution time..... 5-132
Task Name..... 4-8, 4-10
Task Period Exceeded Count..... 5-9, 5-125, A-93, A-165
Task Period Exceeded Detection..... 4-9
Task Period Exceeded Flag... 5-9, 5-125, 5-131, A-93, A-165
Task Timeout Detection Time..... 4-10
Task Type..... 4-8
TCP Application Communications Error..... A-133, A-218
TCP Application Setting Error..... A-144, A-230
Text strings..... 6-34, 6-46
TIME..... 6-34, 6-46
Time of day..... 6-34, 6-46
TIME_ OF_DAY..... 6-34, 6-46
Total Power ON Time..... A-99, A-178, A-289, A-291
Trace Busy Flag..... 8-75, A-102, A-183
Trace Completed Flag..... 8-75, A-102, A-184

Trace Parameter Error Flag.....8-75, A-102, A-184
Trace Trigger Monitor Flag..... 8-75, A-102, A-184
Transmission Error..... A-104, A-186

U

UDINT.....6-33
UINT..... 6-33
ULINT..... 6-33
Unions..... 6-36
Unit and Slave Parameters Transfer Flag..... A-98, A-176
Unit Version.....A-101, A-181, A-289, A-292
UnitVersion (Unit Version)..... A-289, A-292
User Program and Settings Transfer Flag.....A-98, A-175
User-defined Error Status.....8-93, A-93, A-166
User-defined variables.....2-13
USINT.....6-33

V

Variable Access Time..... 4-10
Variable Log Output Setting Name.....A-111, A-195
Variable Log Output Setting Status..... A-111, A-195
Variable Log Output Setting Type..... A-111, A-195
Verifying data.....9-3
version..... 25

W

WORD..... 6-33
Write Protection Key.....8-33

X

X Bus Function Module Error Status.....A-115, A-200
X Bus Function Module Master Error Status.....A-116, A-200
X Bus Function Module Unit Error Status..... A-116, A-200
X Bus Function Module Unit Error Status Table A-116, A-201
X Bus Unit Common Function Module Error Status.....
.....A-290, A-292
X Bus Unit configuration..... 1-8
X Bus Unit Registration Status.....A-115, A-200
X Bus Unit Settings Transfer Flag..... A-98, A-177
XBU_ErrSta (X Bus Unit Common Function Module Error Status).....A-290, A-292

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.
438B Alexandra Road, #08-01/02 Alexandra
Technopark, Singapore 119968
Tel: (65) 6835-3011 Fax: (65) 6835-3011

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-6023-0333 Fax: (86) 21-5037-2388

Authorized Distributor:

©OMRON Corporation 2011-2023 All Rights Reserved.
In the interest of product improvement,
specifications are subject to change without notice.

Cat. No. W501-E1-44 0723